Max-Planck-Institute for Gravitational Physics (Albert-Einstein-Institute)

Università degli Studi di Trento

LTPDA Unit Test Report S2-AEI-TN-XXXX

prepared by	M Hewitson
checked by	
Issue	1
Revision	1
Status	Draft
Number of pages	1094
date of issue	April 28, 2011
approved by	

Distribution List

Name	Company/ Institute
	AEI Hannover

Contents

List of Tables

1 Introduction

This document captures the results of the unit test run on LTPDA. The development cycle of LTPDA uses an automated test-rig which runs every night using a fresh build of the toolbox. The test-rig runs a series of different tests, including the unit tests presented in this report.

The LTPDA unit tests are gathered according to the object class and method within that class. For each unit test, there is a syntax test (to check that the method can be run) and an algorithm test (to check that the method does the correct thing).

The test tables presented in this document are structured as follows:

- Column 1 contains the test name or number, together with any sub-name or number
- Column 2 contains the general description of the test.
- Column 3, row 1 contains the description of the syntax test.
- Column 3, row 2 contains the description of the algorithm test.
- Column 4, row 1 contains the result of the syntax test (pass or fail).
- Column 4, row 2 contains the result of the algorithm test (pass or fail).

1.1 Version

This report refers to version 2.4 of the LTPDA toolbox.

2 Results

ao/abs			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/abs] method works with a vector of objects as input.	Test that the [ao/abs] method works for a vector of objects as input.	pass

ao/abs			
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/abs] method works with a matrix of objects as input.	Test that the [ao/abs] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/abs] method works with a list of objects as input.	Test that the [ao/abs] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/abs] method works with a mix of different arrays of objects as input.	Tests that the [ao/abs] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/abs] method properly applies history.	Test that the result of applying the [ao/abs] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/abs]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/abs] method can modify the input AO.	Test that the [ao/abs] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/abs] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/abs] method uses the plist to get the axis.	Test that the [ao/abs] method uses the plist to get the axis.	pass

ao/abs			
		1) Check that the [ao/abs] method applies to the x-axis 2) Check that the [ao/abs] method applies to the y-axis 3) Check that the [ao/abs] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/abs] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/abs] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/abs] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 1: Unit tests for ao/abs.

ao/acos			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/acos] method works with a vector of objects as input.	Test that the [ao/acos] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/acos] method works with a matrix of objects as input.	Test that the [ao/acos] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/acos] method works with a list of objects as input.	Test that the [ao/acos] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/acos] method works with a mix of different arrays of objects as input.	Tests that the [ao/acos] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/acos] method properly applies history.	Test that the result of applying the [ao/acos] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/acos]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/acos] method can modify the input AO.	Test that the [ao/acos] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/acos			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/acos] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/acos] method uses the plist to get the axis.	Test that the [ao/acos] method uses the plist to get the axis.	pass
		1) Check that the [ao/acos] method applies to the x-axis 2) Check that the [ao/acos] method applies to the y-axis 3) Check that the [ao/acos] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/acos] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/acos] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/acos] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 2: Unit tests for ao/acos.

ao/and			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
rule1 (tsdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass

ao/and			
rule1 (cdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single tsdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single fsdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector cdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector cdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single tsdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single fsdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP cdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ tsdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ fsdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP cdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/and			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
and			pass
			pass
tests (fsdata and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (AO no data and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and AO no data)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different fs in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x values in fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

ao/and			

Table 3: Unit tests for ao/and.

ao/angle			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/angle] method works with a vector of objects as input.	Test that the [ao/angle] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/angle] method works with a matrix of objects as input.	Test that the [ao/angle] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/angle] method works with a list of objects as input.	Test that the [ao/angle] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/angle] method works with a mix of different arrays of objects as input.	Tests that the [ao/angle] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/angle] method properly applies history.	Test that the result of applying the [ao/angle] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/angle]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/angle] method can modify the input AO.	Test that the [ao/angle] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/angle			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/angle] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/angle] method uses the plist to get the axis.	Test that the [ao/angle] method uses the plist to get the axis.	pass
		1) Check that the [ao/angle] method applies to the x-axis 2) Check that the [ao/angle] method applies to the y-axis 3) Check that the [ao/angle] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/angle] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/angle] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/angle] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 4: Unit tests for ao/angle.

ao/ao			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/ao] method works with a vector of objects as input.	Test that the [ao/ao] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/ao] method works with a matrix of objects as input.	Test that the [ao/ao] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/ao] method works with a list of objects as input.	Test that the [ao/ao] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/ao] method works with a mix of different arrays of objects as input.	Tests that the [ao/ao] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/ao] method properly applies history.	Test that the result of applying the [ao/ao] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/ao]'. 2) Check that the rebuilt object is the same object as the input.	pass
08	Tests that the ao method properly applies history to the ASCII-file constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt objects are the same as 'out1' and 'ouit2'	pass

ao/ao			
09	Tests that the ao method properly applies history to the complex ASCII-file constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the read data are correct 3) Check that the rebuilt object is the same as in 'out1..5'	pass
13	Tests that the ao method properly applies history to the vals constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
15	Tests that the ao method properly applies history to the plist(fcn) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
16	Tests that the ao method properly applies history to the plist(vals) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
17	Tests that the ao method properly applies history to the plist(x/y-vals) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
18	Tests that the ao method properly applies history to the plist(tsfcn) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
19	Tests that the ao method properly applies history to the plist(fsfcn) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt objects are the same as 'out1' and 'out2'	pass
20	Tests that the ao method properly applies history to the plist(win) constructor.	Test that the output can be processed back to an m-file.	pass

ao/ao			
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
21	Tests that the ao method properly applies history to the plist(waveform) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt objects are the same as 'out_...'.	pass
23	Tests that the ao method properly applies history to the plist(polynomial) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
25	Tests that the ao method properly applies history to the plist(pzmodel) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
26	Tests that the ao method properly applies history to the data-object constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'.	pass
28	Tests that the ao method properly applies history to the x-vector, y-vector constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
29	Tests that the ao method properly applies history to the filename + plist constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
30	Tests that the ao method properly applies history to the filename + plist constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check the read data 3) Check that the rebuilt object is the same as 'out'	pass
31	Tests that the ao method properly applies history to the filename + plist constructor.	Test that the output can be processed back to an m-file.	pass

ao/ao			
		1) Check that the last entry in the history of 'out' corresponds to 'ao'. 2) Check that the rebuilt object is the same as 'out'.	pass
60	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
61	Tests that the constructor properly applies history to the read XML-file constructor.	Tests that the constructor properly applies history to the read XML-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
62	Tests that the constructor properly applies history in the struct constructor.	Tests that the constructor properly applies history in the struct constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
64	Tests that the contructor properly applies history to the plist(filename) constructor.	Tests that the contructor properly applies history to the plist(filename) constructor.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
65	Tests that the contructed object can be submitted and retrieved.	Tests that the contructed object can be submitted and retrieved.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
66	Tests that the construuctor properly works with the plist(pzmodel) constructor.	Tests that the constructor properly works with the plist(pzmodel) constructor.	pass

ao/ao			
		1) Check that the last entry in the history of 'out' corresponds to 'ssm'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
67	Tests that the constructor properly applies history to the pole/zero model + plist object constructor.	Tests that the constructor properly applies history to the pole/zero model + plist object constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
68	Tests that the constructor properly applies history to the conn+Id constructor.	Tests that the constructor properly applies history to the conn+Id constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 5: Unit tests for ao/ao.

ao/asin			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/asin] method works with a vector of objects as input.	Test that the [ao/asin] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/asin] method works with a matrix of objects as input.	Test that the [ao/asin] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/asin] method works with a list of objects as input.	Test that the [ao/asin] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/asin] method works with a mix of different arrays of objects as input.	Tests that the [ao/asin] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/asin] method properly applies history.	Test that the result of applying the [ao/asin] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/asin]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/asin] method can modify the input AO.	Test that the [ao/asin] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/asin			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/asin] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/asin] method uses the plist to get the axis.	Test that the [ao/asin] method uses the plist to get the axis.	pass
		1) Check that the [ao/asin] method applies to the x-axis 2) Check that the [ao/asin] method applies to the y-axis 3) Check that the [ao/asin] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/asin] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/asin] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/asin] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 6: Unit tests for ao/asin.

ao/atan			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/atan] method works with a vector of objects as input.	Test that the [ao/atan] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/atan] method works with a matrix of objects as input.	Test that the [ao/atan] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/atan] method works with a list of objects as input.	Test that the [ao/atan] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/atan] method works with a mix of different arrays of objects as input.	Tests that the [ao/atan] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/atan] method properly applies history.	Test that the result of applying the [ao/atan] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/atan]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/atan] method can modify the input AO.	Test that the [ao/atan] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/atan			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/atan] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/atan] method uses the plist to get the axis.	Test that the [ao/atan] method uses the plist to get the axis.	pass
		1) Check that the [ao/atan] method applies to the x-axis 2) Check that the [ao/atan] method applies to the y-axis 3) Check that the [ao/atan] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/atan] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/atan] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/atan] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 7: Unit tests for ao/atan.

ao/atan2			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the atan2 method works only with two AOs as input..	Tests that the atan2 method works only with two AOs as input..	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the atan2 method properly applies history.	Test that the result of applying the atan2 method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'atan2'. 2) Check that the rebuilt object is the same object as the input.	pass
04	Tests that the atan2 method can not be used as a modifier method.	Tests that the atan2 method can not be used as a modifier method. The command should fail.	pass
		1) Nothing to test.	pass

Table 8: Unit tests for ao/atan2.

ao/average			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the average method works with a vector of AOs as input.	Tests that the average method works with a vector of AOs as input.	pass
		1) Check that the outputs have exactly one AO 2) Check that the outputs have the correct data. 3) Check the rebuilt objects	pass
03	Tests that the average method works with a matrix of AOs as input.	Tests that the average method works with a matrix of AOs as input.	pass
		1) Check that the outputs have exactly one AO 2) Check that the outputs have the correct data. 3) Check the rebuilt objects	pass
04	Tests that the average method works with a list of AOs as input.	Tests that the average method works with a list of AOs as input.	pass
		1) Check that the outputs have exactly one AO 2) Check that the outputs have the correct data. 3) Check the rebuilt objects	pass
05	Tests that the average method works with a mix of different shaped AOs as input.	Tests that the average method works with a mix of different shaped AOs as input.	pass
		1) Check that the output is exact one AO 2) Check that the output have the correct data.	pass
06	Tests that the [ao/average] method properly applies history.	Test that the result of applying the [ao/average] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/average]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the average method cannot modify the input AO. The method must throw an error for the modifier call.	Test that the average method cannot modify the input AO by calling with no output	pass
		1) Nothing to check.	pass

ao/average			
08	Test the shape of the output.	Test that the average method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
18	Tests that the average method works with a single AO as input.	Tests that the average method works with a single AO as input.	pass
		1) Check that the outputs have exactly one AO 2) Check that the outputs have the correct data. 3) Check the rebuilt objects	pass

Table 9: Unit tests for ao/average.

bin/data			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [bin/data] method works with a vector of objects as input.	Test that the [bin/data] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [bin/data] method works with a matrix of objects as input.	Test that the [bin/data] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [bin/data] method works with a list of objects as input.	Test that the [bin/data] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [bin/data] method works with a mix of different arrays of objects as input.	Tests that the [bin/data] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [bin/data] method properly applies history.	Test that the result of applying the [bin/data] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[bin/data]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [bin/data] method can modify the input AO.	Test that the [bin/data] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

bin/data			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [bin/data] value of the copy 4) Check that out and amodi are the same	pass
10	Check that the [bin/data] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [bin/data] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
17	Tests handling of units: 1) rebinning of the PSD data 2) rebinning of the LPSD data 3) compares the units of the input and output	1) Rebinning the data	pass
		1) Check that (rebinned PSD yunits) equals (input PSD yunits) 2) Check that (rebinned PSD xunits) equals (input PSD xunits) 3) Check that (rebinned LPSD yunits) equals (input LPSD yunits) 4) Check that (rebinned LPSD xunits) equals (input LPSD xunits)	pass
18	Tests handling of units: 1) rebinning of the PSD data 2) rebinning of the LPSD data 3) compares the units of the input and output	1) Rebinning the data	pass
		1) Check that (rebinned PSD yunits) equals (input PSD yunits) 2) Check that (rebinned PSD xunits) equals (input PSD xunits) 3) Check that (rebinned LPSD yunits) equals (input LPSD yunits) 4) Check that (rebinned LPSD xunits) equals (input LPSD xunits)	pass
19	Tests handling of units: 1) rebinning of the PSD data 2) rebinning of the LPSD data 3) compares the units of the input and output	1) Rebinning the data	pass

bin/data			
		1) Check that (rebinned PSD yunits) equals (input PSD yunits) 2) Check that (rebinned PSD xunits) equals (input PSD xunits) 3) Check that (rebinned LPSD yunits) equals (input LPSD yunits) 4) Check that (rebinned LPSD xunits) equals (input LPSD xunits)	pass
20	Tests handling of units: 1) rebinning of the PSD data 2) rebinning of the LPSD data	1) Rebinning the data	pass
		Nothing to check	pass
21	Tests handling of units: 1) rebinning of the PSD data 2) rebinning of the LPSD data	1) Rebinning the data	pass
		Nothing to check	pass

Table 10: Unit tests for bin/data.

ao/cat			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/cat] method works with a vector of objects as input.	Test that the [ao/cat] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/cat] method works with a matrix of objects as input.	Test that the [ao/cat] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/cat] method works with a list of objects as input.	Test that the [ao/cat] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/cat] method works with a mix of different arrays of objects as input.	Tests that the [ao/cat] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	The cat method doesn't apply history.	The cat method doesn't apply history.	pass
		1) Nothing to test	pass
07	The cat method can not be used as a modifer method.	The cat method can not be used as a modifer method. In this case throws the method an error.	pass
		1) Nothing to test.	pass

Table 11: Unit tests for ao/cat.

ao/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of AOs as input.	Test that the char method works for a vector of AOs as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the char method works with a matrix of AOs as input.	Test that the char method works for a matrix of AOs as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the char method works with a list of AOs as input.	Test that the char method works for a list of AOs as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the char method works with a mix of different shaped AOs as input.	Test that the char method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the char method properly applies history.	The method char doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 12: Unit tests for ao/char.

ao/cohere			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the cohere method works with a vector of AOs as input.	Test that the cohere method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1 2) Check that each output AO contains the correct data.	pass
03	Test that the cohere method doesn't work for a matrix of AOs as input.	Test that the cohere method doesn't work for a matrix of AOs as input.	pass
		1) Nothing to check	pass
04	Tests that the cohere method works with a list of AOs as input (only two objects).	Test that the cohere method works for a list of AOs as input. (only two objects)	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that each output AO contains the correct data.	pass
05	Test that the cohere method doesn't work with an input of matrices and vectors and single AOs.	Test that the cohere method doesn't work with an input of matrices and vectors and single AOs.	pass
		1) Nothing to check	pass
06	Tests that the cohere method properly applies history.	Test that the result of applying the cohere method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'cohere'. 2) Check that the rebuilt object is the same as 'out'.	pass
07	Tests that the cohere method can not modify the input AO.	Test that the cohere method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
08	Test the shape of the output.	Test that the plus method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the output data doesn't change.	pass

ao/cohere			
09	Check that the cohere method pass back the output objects to a list of output variables or to a single variable.	This test is not longer necessary because the cohere method pass back always only one object.	pass
		1) Nothing to check.	pass
10	Tests that the cohere method agrees with MATLAB's mscohere when configured to use the same parameters.	Test that the applying cohere works on two AOs.	pass
		1) Check that output agrees with the output of MATLAB's mscohere. 2) Check that the shape of the output data is equal to the input data	pass
11	Check that the [ao/cohere] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests symmetry properties of complex-coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) complex coherence of the white noise series 4) compare C(x,y) with	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) complex coherence of the white noise	pass
		1) Check that C(x,y) equals conj(C(y,x)) 2) Check that C(x,x) equals 1 2) Check that C(y,y) equals 1	pass
13	Tests symmetry properties of magnitude-squared coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) magnitude-squared coherence of the white noise series 4) compare C(x,y) with	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) magnitude-squared coherence of the white noise	pass
		1) Check that C(x,y) equals C(y,x) 1) Check that C(x,x) equals 1 1) Check that C(y,y) equals 1	pass
14	Tests symmetry properties of complex-coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) complex coherence of the combination of white noise series 4) compare C(x,y) with 1	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) complex coherence of the combination of noise	pass

ao/cohere			
		1) Check that the complex coherence equals 1	pass
15	Tests symmetry properties of complex-coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) magnitude-squared	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) magnitude-squared coherence of the combination of noise	pass
		1) Check that the magnitude-squared coherence equals 1	pass
16	Tests symmetry properties of complex-coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) magnitude-squared coherence M of the combination of white noise series 4) complex	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) magnitude-squared coherence of the combination of noise	pass
		1) Check that the magnitude-squared coherence equals the square modulus of the complex coherence	pass
17	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) complex coherence of the white noise series 4) compares the units of the input	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) complex cohere of the white noise	pass
		1) Check that (complex coherence yunits) equals [1] 2) Check that (complex coherence xunits) equals [Hz]	pass
18	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) magnitude-squared coherence of the white noise series 4) compares the units of the input and output	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) magnitude-squared cohere of the white noise	pass
		1) Check that (magnitude-squared coherence yunits) equals [1] 2) Check that (magnitude-squared coherence xunits) equals [Hz]	pass
19	Tests that differently sized data sets are treated properly	Test that applying cohere works on two AOs.	pass

ao/cohere			
		1) Check that cohere used the length of the shortest ao.	pass
20	Tests that applying a single window the coherence is 1	Test that applying cohere works on two AOs.	pass
		1) Check that the calculated cohere is 1	pass
21	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cohere of the	1) Prepare the test tsdata: white noise from normal distribution + offset 2) cohere of the noise, without detrending, random window, set number of averages	pass
		1) Check that calculated navs are identical to those requested	pass
22	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from uniform pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cohere of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run cohere	1) white noise produced from uniform pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cohere of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run cohere again, with the nfft used	pass
		1) Check that calculated objects C1 and C2 are identical	pass
23	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cohere of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run cohere again, with the nfft used 6) compare navs, nfft, coheres	1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cohere of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run cohere again, with the nfft used 6) run cohere again, with conflicting parameters, and verify it uses nfft rather than navs	pass
		1) Check that calculated objects C1 and C2 are identical 2) Check that C3 used different values	pass
24	Tests that the cohere method agrees with MATLAB's mscohere when configured to use the same parameters.	Test that the applying cohere works on two AOs.	pass
		1) Check that output agrees with the output of MATLAB's mscohere. 2) Check that the shape of the output data is equal to the input data	pass

ao/cohere			
25	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) complex coherence of the white noise series 4) compares the units of the input	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) complex cohere of the white noise	pass
		1) Check that (complex coherence yunits) equals [1] 2) Check that (complex coherence xunits) equals [Hz]	pass
26	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) complex coherence of the white noise series 4) compares the units of the input and output	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) complex cohere of the white noise	pass
		1) Check that (complex coherence yunits) equals [1] 2) Check that (complex coherence xunits) equals [Hz]	pass
30	Tests handling of special cases: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) the same noise series 3) cohere of the white noise series 4) compares the output to unity	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: the same data as 1) and 2) 4) cohere of the series	pass
		1) Check that calculated cohere equals 1	pass

Table 13: Unit tests for ao/cohere.

ao/complex			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the complex method works with a vector of AOs as input.	Test that the complex method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that the output AO contains the correct data.	pass
03	Tests that the complex method works with a list of AOs as input.	Test that the complex method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that the output AO contains the correct data.	pass
05	Tests that the complex method can not modify the input AO.	Test that the complex method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
06	Tests that the [ao/complex] method properly applies history.	Test that the result of applying the [ao/complex] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/complex]'. 2) Check that the re-built object is the same object as the input.	pass
07	Control the method with a plist.	Test that the complex method keeps the data shape of the first input object. the input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
11	Check that the [ao/complex] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 14: Unit tests for ao/complex.

ao/compute			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the complex method works with a vector of AOs as input.	Test that the complex method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that each output AO contains the correct data.	pass
03	Tests that the compute method works with a matrix of AOs as input.	Test that the compute method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is 2. 2) Check that each output AO contains the correct data.	pass
04	Tests that the compute method works with a list of AOs as input.	Test that the compute method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is 2. 2) Check that each output AO contains the correct data.	pass
05	Tests that the compute method works with a mix of different shaped AOs as input.	Test that the compute method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is 2. 2) Check that each output AO contains the correct data.	pass
06	Tests that the compute method applies no history.	Test that the result of applying the compute method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' is not 'compute'. 2) Check that the rebuilt object is the same as 'out'.	pass
11	Check that the [ao/compute] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 15: Unit tests for ao/compute.

ao/conj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/conj] method works with a vector of objects as input.	Test that the [ao/conj] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/conj] method works with a matrix of objects as input.	Test that the [ao/conj] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/conj] method works with a list of objects as input.	Test that the [ao/conj] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/conj] method works with a mix of different arrays of objects as input.	Tests that the [ao/conj] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/conj] method properly applies history.	Test that the result of applying the [ao/conj] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/conj]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/conj] method can modify the input AO.	Test that the [ao/conj] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/conj			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/conj] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/conj] method uses the plist to get the axis.	Test that the [ao/conj] method uses the plist to get the axis.	pass
		1) Check that the [ao/conj] method applies to the x-axis 2) Check that the [ao/conj] method applies to the y-axis 3) Check that the [ao/conj] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/conj] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/conj] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/conj] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 16: Unit tests for ao/conj.

ao/conv			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the conv method works with a vector of AOs as input.	Test that the conv method works for a vector of AOs as input.	pass
		1) Check that the output is exact one Ao with cdata. 2) Check that each output AO contains the correct data.	pass
03	Tests that the conv method works with a matrix of AOs as input.	Test that the conv method works for a matrix of AOs as input.	pass
		1) Check that the output is exact one Ao with cdata. 2) Check that each output AO contains the correct data.	pass
04	Tests that the conv method works with a list of AOs as input.	Test that the conv method works for a list of AOs as input.	pass
		1) Check that the output is exact one Ao with cdata. 2) Check that each output AO contains the correct data.	pass
05	Tests that the conv method works with a mix of different shaped AOs as input.	Test that the conv method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the output is exact one Ao with cdata. 2) Check that each output AO contains the correct data.	pass
06	Tests that the conv method properly applies history.	Test that the result of applying the conv method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'conv'. 2) Check that the rebuilt object is the same object as the input.	pass
07	The conv method can not modify the input AO.	The conv method throws an error if it is uesed as a modifier.	pass
		1) Nothing to test.	pass
11	Check that the [ao/conv] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

ao/conv			

Table 17: Unit tests for ao/conv.

ao/convert			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the convert method works with a vector of AOs as input.	Test that the convert method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the convert method works with a matrix of AOs as input.	Test that the convert method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the convert method works with a list of AOs as input.	Tests that the convert method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
05	Tests that the convert method works with a mix of different shaped AOs as input.	Tests that the convert method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
06	Tests that the convert method properly applies history.	Test that the result of applying the convert method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'convert'. 2) Check that the rebuilt object is the same object as the input.	pass

ao/convert			
07	Tests that the convert method can modify the input AO.	Test that the convert method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the convert value of the copy 4) Check that out and amodi are the same	pass
08	Control the method with a plist.	Test that the convert method can modify the input object depending to the plist.	pass
		1) Check that the convert method aplpiesthe different actions 4) Check that the rebuilt objects are the same as 'out[1..6]'.	pass
09	Test the shape of the output.	Test that the convert method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the convert method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/convert] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 18: Unit tests for ao/convert.

ao/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 19: Unit tests for ao/copy.

ao/cos			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/cos] method works with a vector of objects as input.	Test that the [ao/cos] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/cos] method works with a matrix of objects as input.	Test that the [ao/cos] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/cos] method works with a list of objects as input.	Test that the [ao/cos] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/cos] method works with a mix of different arrays of objects as input.	Tests that the [ao/cos] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/cos] method properly applies history.	Test that the result of applying the [ao/cos] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/cos]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/cos] method can modify the input AO.	Test that the [ao/cos] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/cos			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/cos] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/cos] method uses the plist to get the axis.	Test that the [ao/cos] method uses the plist to get the axis.	pass
		1) Check that the [ao/cos] method applies to the x-axis 2) Check that the [ao/cos] method applies to the y-axis 3) Check that the [ao/cos] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/cos] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/cos] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/cos] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 20: Unit tests for ao/cos.

ao/cov			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the cov method works with a vector of AOs as input.	Test that the cov method works for a vector of AOs as input.	pass
		1) Check that the output is exact one Ao with cdata. 2) Check that each output AO contains the correct data.	pass
03	Tests that the cov method works with a matrix of AOs as input.	Test that the cov method works for a matrix of AOs as input.	pass
		1) Check that the output is exact one Ao with cdata. 2) Check that each output AO contains the correct data.	pass
04	Tests that the cov method works with a list of AOs as input.	Tests that the cov method works with a list of AOs as input.	pass
		1) Check that the output is exact one Ao with cdata. 2) Check that each output AO contains the correct data.	pass
05	Tests that the cov method works with a mix of different shaped AOs as input.	Tests that the cov method works with a mix of different shaped AOs as input.	pass
		1) Check that the output is exact one Ao with cdata. 2) Check that each output AO contains the correct data.	pass
06	Tests that the cov method properly applies history.	Test that the result of applying the cov method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'cov'. 2) Check that the rebuilt object is the same object as the input.	pass
07	The cov method can not modify the input AO.	The cov method can not modify the input AO.	pass
		1) Nothing to test.	pass
11	Check that the [ao/cov] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

ao/cov			

Table 21: Unit tests for ao/cov.

ao/cpsd			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the cpsd method works with a vector of AOs as input. (only with two objects in the vector)	Test that the cpsd method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input.	pass
03	Tests that the cpsd method doesn't work with a matrix of AOs as input.	Test that the cpsd method doesn't work for a matrix of AOs as input.	pass
		1) Nothing to check.	pass
04	Tests that the cpsd method works with a list of AOs as input.	Test that the cpsd method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the cpsd method doesn't work with a mix of different shaped AOs as input.	Test that the cpsd method doesn't work with an input of matrices and vectors and single AOs.	pass
		1) Nothing to check	pass
06	Tests that the cpsd method properly applies history.	Test that the result of applying the cpsd method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'cpsd'. 2) Check that the rebuilt object is the same as 'out'.	pass
07	Tests that the cpsd method can not modify the input AO.	Test that the cpsd method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
08	Test the shape of the output.	Test that the cpsd method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the output data doesn't change.	pass
09	Check that the cpsd method pass back the output objects to a list of output variables or to a single variable.	This test is not longer necessary because the cpsd method pass back always only one object.	pass

ao/cpsd			
		1) Nothing to check.	pass
10	Tests that the cpsd method agrees with MATLAB's cpsd when configured to use the same parameters.	Test that the applying cpsd works on two AOs.	pass
		1) Check that output agrees with the output of MATLAB's cpsd.	pass
11	Check that the [ao/cpsd] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
17	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) CPSD of the white noise series 4) compares the	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) CPSD of the white noise	pass
		1) Check that (calculated CPSD yunits) equals input_1 units*input_2 units/Hz	pass
18	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) CPSD of the white noise series Comparison with PSD: 4) compares the off-diagonal terms to check they are complex-conjugated 5)	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) CPSD of the white noise 6) PSD of the white noise	pass
		1) Check that CPSD(x,y) equals conj(CPSD(y,x)) 2) Check that CPSD(x,x) equals PSD(x) 3) Check that CPSD(y,y) equals PSD(y)	pass
24	Test that different sized data sets are treated properly.	Test that applying cpsd works on two AOs.	pass
		1) Check that cpsd used the length of the shortest ao.	pass
25	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) CPSD of the white noise series 4) compares the units of the input and output	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) CPSD of the white noise	pass
		1) Check that (calculated CPSD yunits) equals input_1 units*input_2 units/Hz	pass

ao/cpsd			
51	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cpsd of the noise,	1) Prepare the test tsdata: white noise from normal distribution + offset 2) cpsd of the noise, without detrending, random window, set number of averages	pass
		1) Check that calculated navs are identical to those requested	pass
52	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from uniform pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cpsd of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run cpsd again,	1) white noise produced from uniform pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cpsd of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run cpsd again, with the nfft used	pass
		1) Check that calculated objects C1 and C2 are identical	pass
53	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cpsd of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run cpsd again, with the nfft used 6) compare navs, nfft, psds	1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) cpsd of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run cpsd again, with the nfft used 6) run cpsd again, with conflicting parameters, and verify it uses nfft rather than navs	pass
		1) Check that calculated objects C1 and C2 are identical 2) Check that C3 used different values	pass

Table 22: Unit tests for ao/cpsd.

ao/created			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the created method works with a vector of AOs as input.	Test that the created method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output contains the correct data.	pass
03	Tests that the created method works with a matrix of AOs as input.	Test that the created method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output contains the correct data.	pass
04	Tests that the created method works with a list of AOs as input.	Test that the created method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the created method works with a mix of different shaped AOs as input.	Test that the created method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the created method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the created method can be used with the modify command.	Tests that the created method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the created method retruns always a well defined time object even for an empty input object.	Test that the created method with an empty AO	pass
		1) Check that the output is a time object with a ell defined time.	pass

Table 23: Unit tests for ao/created.

ao/creator			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the creator method works with a vector of AOs as input.	Test that the creator method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output contains the correct data.	pass
03	Tests that the creator method works with a matrix of AOs as input.	Test that the creator method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output contains the correct data.	pass
04	Tests that the creator method works with a list of AOs as input.	The creator method doesn't work for a list of AOs as input.	pass
		1) Nothing to test.	pass
05	Tests that the creator method works with a mix of different shaped AOs as input.	The creator method doesn't work with different shaped input objects.	pass
		1) Nothing to test	pass
06	Tests that the creator method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the creator method can be used with the modify command.	Tests that the creator method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the creator method retruns all creator(s)/modifier(s) which are in the history.	Test that the creator method uses the option 'all' direct or in a plist. The test file must have the modifier 'first', 'second' and 'third'	pass
		1) Check that out1 contains only one creator 2) Check that out2 contain more creator/modifier	pass
09	Tests the negative case for the option 'all'.	Test that the creator method throws an error if the option 'all' is used in connection with a matrix/vector of AOs.	pass
		1) Nothing to test.	pass

ao/creator			

Table 24: Unit tests for ao/creator.

ao/ctranspose			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/ctranspose] method works with a vector of objects as input.	Test that the [ao/ctranspose] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/ctranspose] method works with a matrix of objects as input.	Test that the [ao/ctranspose] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/ctranspose] method works with a list of objects as input.	Test that the [ao/ctranspose] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/ctranspose] method works with a mix of different arrays of objects as input.	Tests that the [ao/ctranspose] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/ctranspose] method properly applies history.	Test that the result of applying the [ao/ctranspose] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/ctranspose]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the ctranspose method can modify the input AO.	Test that the ctranspose method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/ctranspose			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is ctranspose(at1).	pass
09	Control the method with a plist.	Test that the abs method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the [ao/ctranspose] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/ctranspose] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 25: Unit tests for ao/ctranspose.

ao/delay			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/delay] method works with a vector of objects as input.	Test that the [ao/delay] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/delay] method works with a matrix of objects as input.	Test that the [ao/delay] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/delay] method works with a list of objects as input.	Test that the [ao/delay] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/delay] method works with a mix of different arrays of objects as input.	Tests that the [ao/delay] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/delay] method properly applies history.	Test that the result of applying the [ao/delay] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/delay]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/delay] method can modify the input AO.	Test that the [ao/delay] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/delay			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/delay] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/delay] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/delay] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/delay] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 26: Unit tests for ao/delay.

ao/demux			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the demux method works with a mix of different shaped AOs as input.	Tests that the demux method works with a mix of different shaped AOs as input.	pass
		1) Check the output objects.	pass
03	Negative test. Check that the demux method throwns an error for too few output variables.	Check that the demux method throwns an error for too few output variables.	pass
		1) Nothing to check.	pass
04	Negative test. Check that the demux method throwns an error for too many output variables.	Check that the demux method throwns an error for too few output variables.	pass
		1) Nothing to check.	pass

Table 27: Unit tests for ao/demux.

ao/det			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the det method works with a vector of AOs as input.	Test that the det method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the det method works with a matrix of AOs as input.	Test that the det method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the det method works with a list of AOs as input.	Test that the det method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the det method works with a mix of different shaped AOs as input.	Test that the det method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the det method properly applies history.	Test that the result of applying the det method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'det'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the det method can modify the input AO.	Test that the det method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at4' and 'ain' are now different. 2) Check that 'ain' is det(at4).	pass

ao/det			
08	Control the method with a plist.	Test that the det method can modify the single axis controlled by the plist and the resuld can be processed back to an m-file.	pass
		1) Check that the det method applies to the x-axis 2) Check that the det method applies to the y-axis 3) Check that the det method applies to both axes 4) Check that the re-built objects are the same object as 'out[1..3]'.	pass
09	Control the method with a plist.	Test that the det method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the det method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/det] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 28: Unit tests for ao/det.

ao/detrend			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/detrend] method works with a vector of objects as input.	Test that the [ao/detrend] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/detrend] method works with a matrix of objects as input.	Test that the [ao/detrend] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/detrend] method works with a list of objects as input.	Test that the [ao/detrend] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/detrend] method works with a mix of different arrays of objects as input.	Tests that the [ao/detrend] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/detrend] method properly applies history.	Test that the result of applying the [ao/detrend] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/detrend]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/detrend] method can modify the input AO.	Test that the [ao/detrend] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/detrend			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/detrend] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/detrend] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/detrend] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/detrend] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 29: Unit tests for ao/detrend.

ao/dft			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/dft] method works with a vector of objects as input.	Test that the [ao/dft] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/dft] method works with a matrix of objects as input.	Test that the [ao/dft] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/dft] method works with a list of objects as input.	Test that the [ao/dft] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/dft] method works with a mix of different arrays of objects as input.	Tests that the [ao/dft] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/dft] method properly applies history.	Test that the result of applying the [ao/dft] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/dft]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/dft] method can modify the input AO.	Test that the [ao/dft] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/dft			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/dft] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/dft] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/dft] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/dft] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 30: Unit tests for ao/dft.

ao/diag			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the diag method works with a vector of AOs as input.	Test that the diag method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the diag method works with a matrix of AOs as input.	Test that the diag method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the diag method works with a list of AOs as input.	Test that the diag method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the diag method works with a mix of different shaped AOs as input.	Test that the diag method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the diag method properly applies history.	Test that the result of applying the diag method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'diag'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the diag method can modify the input AO.	Test that the diag method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at4' and 'ain' are now different. 2) Check that 'ain' is diag(at4).	pass

ao/diag			
08	Control the method with a plist.	Test that the diag method can modify the single axis controlled by the plist and the resuld can be processed back to an m-file.	pass
		1) Check that the svd method applies with different options. 2) Check that the re-built object is the same object as 'out'.	pass
09	Control the method with a plist.	Test that the diag method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the diag method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/diag] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 31: Unit tests for ao/diag.

ao/diff			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/diff] method works with a vector of objects as input.	Test that the [ao/diff] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/diff] method works with a matrix of objects as input.	Test that the [ao/diff] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/diff] method works with a list of objects as input.	Test that the [ao/diff] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/diff] method works with a mix of different arrays of objects as input.	Tests that the [ao/diff] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/diff] method properly applies history.	Test that the result of applying the [ao/diff] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/diff]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/diff] method can modify the input AO.	Test that the [ao/diff] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/diff			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/diff] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/diff] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/diff] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/diff] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass
13	Control the method with a plist.	Test the computation of derivative using a 2nd order	pass
		1) Check that the diff method uses the 2nd order derivative. 2) Check that the re-built object is the same object as 'out'.	pass
14	Control the method with a plist.	Test the computation of derivative using a 2nd order with a parabolic fit	pass

ao/diff			
		1) Check that the diff method uses the 2nd order derivative with a parabolic fit 2) Check that the re-built object is the same object as 'out'.	pass
15	Control the method with a plist.	Test the 5 point derivative.	pass
		1) Check that the diff method uses the 5 point derivative. 2) Check that the re-built object is the same object as 'out'.	pass

Table 32: Unit tests for ao/diff.

ao/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of AOs as input.	Test that the display method works for a vector of AOs as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of AOs as input.	Test that the display method works for a matrix of AOs as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of AOs as input.	Test that the display method works for a list of AOs as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped AOs as input.	Test that the display method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the display method properly applies history.	The method display doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 33: Unit tests for ao/display.

ao/dopplercorr			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the dopplercorr method works with a vector of AOs as input.	Test that the dopplercorr method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the dopplercorr method works with a matrix of AOs as input.	Test that the dopplercorr method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the dopplercorr method works with a list of AOs as input.	Test that the dopplercorr method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the dopplercorr method works with a mix of different shaped AOs as input.	Test that the dopplercorr method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the dopplercorr method properly applies history.	Test that the result of applying the dopplercorr method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'dopplercorr'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/dopplercorr			
07	Check that the dopplercorr method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/dopplercorr] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 34: Unit tests for ao/dopplercorr.

ao/downsample			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the downsample method works with a vector of AOs as input.	Test that the downsample method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the downsample method works with a matrix of AOs as input.	Test that the downsample method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the downsample method works with a list of AOs as input.	Test that the downsample method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the downsample method works with a mix of different shaped AOs as input.	Test that the downsample method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the downsample method properly applies history.	Test that the result of applying the downsample method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'downsample'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the downsample method can modify the input AO.	Test that the downsample method can modify the input AO by calling with no output.	pass

ao/downsample			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is downsample(at1).	pass
08	Tests that the downsample method keeps the data shape of the input object.	Test that the downsample method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the downsample method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Control the method with a plist.	Test the downsample method with a factor and an offset.	pass
		1) Check that the downsample method with an offset and a factor 2) Check that the re-built object is the same object as 'out'.	pass
11	Check that the [ao/downsample] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 35: Unit tests for ao/downsample.

ao/dropduplicates			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the dropduplicates method works with a vector of AOs as input.	Test that the dropduplicates method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the dropduplicates method works with a matrix of AOs as input.	Test that the dropduplicates method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the dropduplicates method works with a list of AOs as input.	Test that the dropduplicates method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the dropduplicates method works with a mix of different shaped AOs as input.	Test that the dropduplicates method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the dropduplicates method properly applies history.	Test that the result of applying the dropduplicates method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'dropduplicates'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the dropduplicates method can modify the input AO.	Test that the dropduplicates method can modify the input AO by calling with no output.	pass

ao/dropduplicates			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is dropduplicates(at1).	pass
08	Tests that the dropduplicates method keeps the data shape of the input object.	Test that the dropduplicates method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the dropduplicates method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Control the method with a plist.	Test the dropduplicates method with different tolerances.	pass
		1) Check that the different tolerances 2) Check that the re-built object is the same object as 'out'.	pass
11	Check that the [ao/dropduplicates] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 36: Unit tests for ao/dropduplicates.

ao/dsmean			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the dsmean method works with a vector of AOs as input.	Test that the dsmean method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the dsmean method works with a matrix of AOs as input.	Test that the dsmean method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the dsmean method works with a list of AOs as input.	Test that the dsmean method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the dsmean method works with a mix of different shaped AOs as input.	Test that the dsmean method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the dsmean method properly applies history.	Test that the result of applying the dsmean method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'dsmean'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the dsmean method can modify the input AO.	Test that the dsmean method can modify the input AO by calling with no output.	pass

ao/dsmean			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is dsmean(at1).	pass
08	Tests that the dsmean method keeps the data shape of the input object.	Test that the dsmean method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the dsmean method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/dsmean] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 37: Unit tests for ao/dsmean.

ao/eig			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eig method works with a vector of AOs as input.	Test that the eig method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the eig method works with a matrix of AOs as input.	Test that the eig method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the eig method works with a list of AOs as input.	Test that the eig method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the eig method works with a mix of different shaped AOs as input.	Test that the eig method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the eig method properly applies history.	Test that the result of applying the eig method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'eig'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the eig method can modify the input AO.	Test that the eig method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at4' and 'ain' are now different. 2) Check that 'ain' is eig(at4).	pass

ao/eig			
08	Control the method with a plist.	Test that the eig method can modify the single axis controlled by the plist and the resuld can be processed back to an m-file.	pass
		1) Check that the svd method applies with different options. 2) Check that the re-built objects are the same object as 'out[1..2]'.	pass
09	Control the method with a plist.	Test that the eig method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the eig method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/eig] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 38: Unit tests for ao/eig.

ao/eq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eq method works with a vector of AOs as input.	Test that the eq method works for a vector of AOs as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
03	Tests that the eq method works with a matrix of AOs as input.	Test that the eq method works for a matrix of AOs as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
04	Tests that the eq method works with a list of AOs as input.	The eq method doesn't works for a list of AOs as input. Nothing to do.	pass
			pass
05	Tests that the eq method works with a mix of different shaped AOs as input.	The eq method doesn't works for a list of AOs as input. Nothing to do.	pass
			pass
06	Tests that the eq method properly applies history.	The eq method doesn't change the AO, thus will no history added. Nothing to do	pass
			pass
07	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass
08	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'plotinfo'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass

ao/eq			
09	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 't0'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass
10	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'x'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass
11	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'y'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass
12	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'xunits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass
13	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'yunits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass
14	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'fs'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass
15	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'description'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass

ao/eq			
16	Test the eq method with an exception list which is in a plist.	Test that the eq method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 39: Unit tests for ao/eq.

ao/exp			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/exp] method works with a vector of objects as input.	Test that the [ao/exp] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/exp] method works with a matrix of objects as input.	Test that the [ao/exp] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/exp] method works with a list of objects as input.	Test that the [ao/exp] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/exp] method works with a mix of different arrays of objects as input.	Tests that the [ao/exp] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/exp] method properly applies history.	Test that the result of applying the [ao/exp] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/exp]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/exp] method can modify the input AO.	Test that the [ao/exp] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/exp			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/exp] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/exp] method uses the plist to get the axis.	Test that the [ao/exp] method uses the plist to get the axis.	pass
		1) Check that the [ao/exp] method applies to the x-axis 2) Check that the [ao/exp] method applies to the y-axis 3) Check that the [ao/exp] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/exp] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/exp] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/exp] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 40: Unit tests for ao/exp.

ao/export			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the export method works with non complex data in the AO.	Tests that the export method works with non complex data in the AO.	pass
		1) Check that the file exist. 2) Check that the read data is the same as the saved data.	pass
03	Tests that the export method works with complex data in the AO.	Tests that the export method works with complex data in the AO.	pass
		1) Check that the file exist. 2) Check that the read data is the same as the saved data.	pass
04	Tests that the export method works with a plist which contains the filename.	Tests that the export method works with a plist which contains the filename.	pass
		1) Check that the file exist. 2) Check that the read data is the same as the saved data.	pass

Table 41: Unit tests for ao/export.

ao/fft			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/fft] method works with a vector of objects as input.	Test that the [ao/fft] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/fft] method works with a matrix of objects as input.	Test that the [ao/fft] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/fft] method works with a list of objects as input.	Test that the [ao/fft] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/fft] method works with a mix of different arrays of objects as input.	Tests that the [ao/fft] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/fft] method properly applies history.	Test that the result of applying the [ao/fft] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/fft]'. 2) Check that the rebuilt object is the same object as the input.	pass
07	Tests that the [ao/fft] method can modify the input AO.	Test that the [ao/fft] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/fft			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/fft] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/fft] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/fft] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/fft] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests that the fft method agrees with MATLAB's fft when configured to use the same parameters.	Test that the applying fft works on a single AO.	pass
		1) Check that output agrees with the output of MATLAB's fft.	pass
13	Tests that the fft method works with different data types. The testing of tsdata types are done before.	Test that the applying fft works on cdata and xydata.	pass
		1) Check that each output AO contains the correct data.	pass
14	Tests that the fft method works with a plist which constains the key/value pair 'type'/'two'.	Test that the applying fft works with a plist.	pass
		1) Check that each output AO contains the correct data. 2) Check that the re-built object is the same object as 'out'.	pass

Table 42: Unit tests for ao/fft.

ao/fftfilt			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the fftfilt method works with a vector of AOs as input.	Test that the fftfilt method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the fftfilt method works with a matrix of AOs as input.	Test that the fftfilt method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the fftfilt method works with a list of AOs as input.	Test that the fftfilt method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the fftfilt method works with a mix of different shaped AOs as input.	Test that the fftfilt method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the fftfilt method properly applies history.	Test that the result of applying the fftfilt method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'fftfilt'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the fftfilt method can modify the input AO.	Test that the fftfilt method can modify the input AO by calling with no output.	pass
		1) Check that 'at2' and 'ain' are now different. 2) Check that 'ain' is fftfilt(at2).	pass

ao/fftfilt			
08	Tests that the fftfilt method keeps the data shape of the input object.	Test that the fftfilt method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the fftfilt method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 43: Unit tests for ao/fftfilt.

ao/filtSubtract			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the filtSubtract method works with a vector of AOs as input.	Test that the filtSubtract method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the filtSubtract method works with a matrix of AOs as input.	Test that the filtSubtract method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the filtSubtract method works with a list of AOs as input.	Test that the filtSubtract method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the filtSubtract method works with a mix of different shaped AOs as input. DOES NOT APPLY TO THIS METHOD	Test that the filtSubtract method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the filtSubtract method properly applies history.	Test that the result of applying the filtSubtract method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'filtSubtract'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/filtSubtract			
07	Tests that the filtSubtract method can not modify the input AO.	Test that the tfe method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass

Table 44: Unit tests for ao/filtSubtract.

ao/filter			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the filter method works with a vector of AOs as input.	Test that the filter method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data. 3) Check the output-filter	pass
03	Tests that the filter method works with a matrix of AOs as input.	Test that the filter method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data. 3) Check the output-filter	pass
04	Tests that the filter method works with a list of AOs as input.	Test that the filter method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the filter method works with a mix of different shaped AOs as input.	Test that the filter method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the filter method properly applies history.	Test that the result of applying the filter method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'filter'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the filter method can modify the input AO.	Test that the filter method can modify the input AO by calling with no output.	pass

ao/filter			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is filter(at1).	pass
08	Tests that the filter method keeps the data shape of the input object.	Test that the filter method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Tests that the filter method with a iir filter which have a different sample rate as the input object.	Test that the filter method can change the sample rate of the filter and that the filter-object is returned.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a iir filter.	pass
10	Tests that the filter method with a fir filter.	Test that the filter method with a fir filter.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a fir filter.	pass
11	Check that the [ao/filter] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests the filter method with an AO and fsdata-object and an iir filter.	Test that the result of the filter method is the product of the AO and the response of the filter.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a iir filter. 3) Check the units	pass
13	Tests that the filter method works with a bank of parallel filters.	Test that the filter method works for bank of parallel filters.	pass
		1) Check that the number of elements in 'out' is the same of the number of elements in 'input' 2) Check that output AO contains the correct number of data. 3) Check that output AO contains the correct data. 4) Check that histout is properly assigned	pass
14	Tests that the filter method works with a bank of serial filters.	Test that the filter method works for bank of serial filters.	pass

ao/filter			
		1) Check that the number of elements in 'out' is the same of the number of elements in 'input' 2) Check that output AO contains the correct number of data. 3) Check that output AO contains the correct data. 4) Check that histout is properly assigned	pass
15	Tests the filter method with an AO of fsdata-object and a bank of iir parallel filter.	Test that the result of the filter method is the product of the AO and the response of the parallel filter bank.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a iir filter. 3) Check the units	pass
16	Tests the filter method with an AO of fsdata-object and a bank of iir serial filter.	Test that the result of the filter method is the product of the AO and the response of the serial filter bank.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a iir filter. 3) Check the units	pass
17	Tests the filter method with an AO and fsdata-object and an mfir filter.	Test that the result of the filter method is the product of the AO and the response of the filter.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a fir filter. 3) Check the units	pass
18	Test that the filter method works with a filter embedded into a matrix and input in a plist	Test that the filter method works with a filter embedded into a matrix	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output AO contains the correct data. 3) Check the output-filter	pass
19	Test that the filter method works with a filter embedded into a matrix and input as a second input	Test that the filter method works with a filter embedded into a matrix	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output AO contains the correct data. 3) Check the output-filter	pass

ao/filter			
20	Test that the filter method does not works with a N-dim matrix of filters input in a plist.	Test that the filter method does not works with a N-dim matrix of filters input in a plist.	pass
		1) Nothing to do.	pass
21	Test that the filter method does not works with a N-dim matrix of filters input as a second object	Test that the filter method does not works with a N-dim matrix of filters input in a plist.	pass
		1) Nothing to do.	pass

Table 45: Unit tests for ao/filter.

ao/filtfilt			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the filtfilt method works with a vector of AOs as input.	Test that the filtfilt method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data. 3) Check the output-filter	pass
03	Tests that the filtfilt method works with a matrix of AOs as input.	Test that the filtfilt method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data. 3) Check the output-filter	pass
04	Tests that the filtfilt method works with a list of AOs as input.	Test that the filtfilt method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the filtfilt method works with a mix of different shaped AOs as input.	Test that the filtfilt method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the filtfilt method properly applies history.	Test that the result of applying the filtfilt method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'filtfilt'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the filtfilt method can modify the input AO.	Test that the filtfilt method can modify the input AO by calling with no output.	pass

ao/filtfilt			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is filtfilt(at1).	pass
08	Tests that the filtfilt method keeps the data shape of the input object.	Test that the filtfilt method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Tests that the filtfilt method with a iir filter which have a different sample rate as the input object.	Test that the filtfilt method can change the sample rate of the filter and that the filtfilt-object is returned.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a iir filtfilt.	pass
10	Tests that the filtfilt method with a fir filter.	Test that the filtfilt method with a fir filter.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a fir filter.	pass
11	Check that the [ao/filtfilt] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests the filtfilt method with an AO and fsdata-object and an iir filter.	Test that the result of the filtfilt method is the product of the AO and the response of the filter.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a iir filter. 3) Check the units	pass
13	Tests the filtfilt method with an AO and fsdata-object and an iir filter.	Test that the result of the filtfilt method is the product of the AO and the response of the filter.	pass
		1) Check that the output AO contains the correct data. 2) Check that the second output is a fir filter. 3) Check the units	pass

Table 46: Unit tests for ao/filtfilt.

ao/find			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the find method works with a vector of AOs as input.	Test that the find method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the find method works with a matrix of AOs as input.	Tests that the find method works with a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the find method works with a list of AOs as input.	Tests that the find method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
05	Tests that the find method works with a mix of different shaped AOs as input.	Tests that the find method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
06	Tests that the find method properly applies history.	Test that the result of applying the find method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'finf'. 2) Check that the rebuilt object is the same object as the input.	pass
07	Tests that the find method can modify the input AO.	Test that the find method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/find			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the abs value of the copy 4) Check that out and amodi are the same	pass
08	Test that the find method works with different queries to the x-/y- axis.	Test that the find method works with different queries to the x-/y-axis.	pass
		1) Check the output	pass
09	Test the shape of the output.	Test that the find method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the find method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/find] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Test that the find method works with AOs which have cdata.	Test that the find method works with AOs which have cdata.	pass
		1) Check the output	pass
13	Test that the find method works with a plist which contains different queries to the x-/y-axis.	Test that the find method works with a plist which contains different queries to the x-/y- axis.	pass
		1) Check the output	pass

Table 47: Unit tests for ao/find.

ao/firwhiten			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the firwhiten method works with a vector of AOs as input.	Test that the firwhiten method works for a vector of AOs as input.	pass
		1) Check the number of elements in 'out' 2) Check the number of filters (outf) and noise-floor estimates (outxx) 3) Check that each output AO contains the correct data.	pass
03	Tests that the firwhiten method works with a matrix of AOs as input.	Test that the firwhiten method works for a matrix of AOs as input.	pass
		1) Check the number of elements in 'out' 2) Check the number of filters (outf) and noise-floor estimates (outxx) 3) Check that each output AO contains the correct data.	pass
04	Tests that the firwhiten method works with a list of AOs as input.	Test that the firwhiten method works for a list of AOs as input.	pass
		1) Check the number of elements in 'out' 2) Check the number of filters (outf) and noise-floor estimates (outxx) 3) Check that each output AO contains the correct data.	pass
05	Tests that the firwhiten method works with a mix of different shaped AOs as input.	Test that the firwhiten method works with an input of matrices and vectors and single AOs.	pass
		1) Check the number of elements in 'out' 2) Check the number of filters (outf) and noise-floor estimates (outxx) 3) Check that each output AO contains the correct data.	pass
06	Tests that the firwhiten method properly applies history.	Test that the result of applying the firwhiten method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'firwhiten'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/firwhiten			
07	Tests that the firwhiten method can modify the input AO.	Test that the firwhiten method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is firwhiten(at1).	pass
08	Tests that the firwhiten method keeps the data shape of the input object.	Test that the firwhiten method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Tests that the firwhiten method with a complex plist.	Test that the result of applying the firwhiten method with a complex plist can be processed back to a m-file.	pass
		1) Check the output data 2) Check the output filter 3) Check the noise-floor estimates 4) Check that the re-built object is the same object as 'out'.	pass
10	Test the spectral falttening capability of firwhiten method.	Test that the application of the firwhiten method enhances the spectral flatness of input data.	pass
		1) Calculate PSD of input and whitened data 2) Compare relative spectral flatness coefficients	pass
11	Check that the [ao/firwhiten] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the errors are cleared for this method.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output has no error fields	pass

Table 48: Unit tests for ao/firwhiten.

ao/fixfs			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the fixfs method works with a vector of AOs as input.	Test that the fixfs method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the fixfs method works with a matrix of AOs as input.	Test that the fixfs method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the fixfs method works with a list of AOs as input.	Test that the fixfs method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the fixfs method works with a mix of different shaped AOs as input.	Test that the fixfs method works with an input of matrices and vectors and single AOs. Additionally define a 'fs' in a plist.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data. 3) Check that 't0' and 'fs' are correct.	pass
06	Tests that the fixfs method properly applies history.	Test that the result of applying the fixfs method can be processed back to an m-file. Additionally define a 'fs' in a plist.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'fixfs'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the fixfs method can modify the input AO.	Test that the fixfs method can modify the input AO by calling with no output.	pass

ao/fixfs			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is fixfs(at1).	pass
08	Tests that the fixfs method keeps the data shape of the input object.	Test that the fixfs method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the fixfs method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the fixfs method works with a list of AOs as input and different 't0' and 'fs' for the inputs.	Test that the fixfs method works for a list of AOs as input and different 't0' and 'fs'	pass
		1) Check that the number of elements in 'out' is the same as the number in the input. 2) Check that each output AO contains the correct data. 3) Check that each output contains the correct frequency and start time.	pass
11	Check that the [ao/fixfs] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests that the fixfs method works with the method 'samples'.	Test that the fixfs method works for the method 'samples'.	pass
		1) Check that each output AO contains the correct data. 2) Check that each output contains the correct frequency and start time.	pass
13	Tests that the fixfs method works with antialising filter.	Test that the fixfs method works for the antialising filters iir and fir.	pass

ao/fixfs			
		1) Check that each output AO contains the correct data. 2) Check that each output contains the correct frequency and start time.	pass
14	Tests that the fixfs method works with an AO with non evenly sampled data.	Test that the fixfs method works for an AO with non evenly samples data.	pass
		1) Check that each output AO contains the correct data. 2) Check that each output contains the correct frequency and start time.	pass

Table 49: Unit tests for ao/fixfs.

ao/fs			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the fs method works with a vector of AOs as input.	The fs method doesn't work with a vector of AOs. Nothing to do	pass
			pass
03	Tests that the fs method works with a matrix of AOs as input.	The fs method doesn't work with a matrix of AOs. Nothing to do	pass
			pass
04	Tests that the fs method works with a list of AOs as input.	The fs method doesn't work with a list of AOs. Nothing to do	pass
			pass
05	Tests that the fs method works with a mix of different shaped AOs as input.	The fs method can only return the fs value of one AO. Nothing to do	pass
			pass
06	Tests that the fs method properly applies history.	The fs method doesn't change the AO, thus will no history added. Nothing to do	pass
			pass
07	Tests that the fs method works for AOs with different data objects.	Test that the fs method returns the fs value for AOs with cdata, fsdata, tsdata and xydata objects.	pass
		1) Check the output.	pass

Table 50: Unit tests for ao/fs.

ao/ge			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ge method compare an AO with scalar value	Test that the ge method works with relational operators and the function command. Use for this AOs with different data objects.	pass
		1) Check that the result of the 'relational operator' and the 'function' command are the same. 2) Check that each output AO contains the correct data.	pass
03	Tests that the ge method compare an AO with one other AO	Test that the ge method works with relational operators and the function command. Use for this AOs with different data objects. Remark that both AOs must have the same size.	pass
		1) Check that the result of the 'relational operator' and the 'function' command are the same. 2) Check that each output AO contains the correct data.	pass

Table 51: Unit tests for ao/ge.

ao/get			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the get method of the ao class.	Test that the get returns returns the value of the specified property. Do this for all properties of the AO.	pass
		1) Check the correct value of the output	pass
03	Tests that the get method works with a plist.	Test that the get returns returns the value of the specified property which is defined in a plist.	pass
		1) Check the correct value of the output	pass
04	Tests the get method of the ao class.	Test that the get throws an error if the input are more than one AO.	pass
		1) Nothing to test	pass

Table 52: Unit tests for ao/get.

ao/gt			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the gt method compare an AO with scalar value	Test that the gt method works with relational operators and the function command. Use for this AOs with different data objects.	pass
		1) Check that the result of the 'relational operator' and the 'function' command are the same. 2) Check that each output AO contains the correct data.	pass
03	Tests that the gt method compare an AO with one other AO	Test that the gt method works with relational operators and the function command. Use for this AOs with different data objects. Remark that both AOs must have the same size.	pass
		1) Check that the result of the 'relational operator' and the 'function' command are the same. 2) Check that each output AO contains the correct data.	pass

Table 53: Unit tests for ao/gt.

ao/heterodyne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the heterodyne method works with a vector of AOs as input.	Test that the heterodyne method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the heterodyne method works with a matrix of AOs as input.	Test that the heterodyne method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the heterodyne method works with a list of AOs as input.	Test that the heterodyne method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the heterodyne method works with a mix of different shaped AOs as input.	Test that the heterodyne method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the heterodyne method properly applies history.	Test that the result of applying the heterodyne method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'heterodyne'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests the heterodyne method functionality.	Build reference signal, mixed signal and heterodyne the 2nd to obtain the first Downsample is set to 'no'	pass

ao/heterodyne			
		Test that we can recover the initial signal after heterodyne up to a numerical error given by tol	pass
08	Tests the heterodyne method functionality.	Build reference signal, mixed signal and heterodyne the 2nd to obtain the first Downsample is set to 'yes'	pass
		Test that we can recover the initial signal after heterodyne up to a numerical error given by tol	pass
11	Check that the [ao/heterodyne] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 54: Unit tests for ao/heterodyne.

ao/hist			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the hist method works with a vector of AOs as input.	Test that the hist method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the hist method works with a matrix of AOs as input.	Test that the hist method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the hist method works with a list of AOs as input.	Test that the hist method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the hist method works with a mix of different shaped AOs as input.	Test that the hist method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the hist method properly applies history.	Test that the result of applying the hist method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'hist'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the hist method can modify the input AO.	Test that the hist method can modify the input AO by calling with no output. Remark that the command at1.hist() doesn't call the histogram method because an AO have a property with the name 'hist'. Thus the command at1.hist returns the value in the property hist.	pass

ao/hist			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is hist(at1).	pass
08	Tests that the hist method keeps the data shape of the input object.	Test that the hist method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the hist method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Control the method with a plist.	Test the hist method with a factor and an offset.	pass
		1) Check that the hist method with defined number of bins 2) Check that the re-built object is the same object as 'out'.	pass
11	Check that the [ao/hist] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Control the method with a plist.	Test the hist method with a factor and an offset.	pass
		1) Check that the hist method with set of bin centers 2) Check that the re-built object is the same object as 'out'.	pass

Table 55: Unit tests for ao/hist.

ao/ifft			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ifft method works with a vector of AOs as input.	Test that the ifft method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the ifft method works with a matrix of AOs as input.	Test that the ifft method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the ifft method works with a list of AOs as input.	Test that the ifft method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the ifft method works with a mix of different shaped AOs as input.	Test that the ifft method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the ifft method properly applies history.	Test that the result of applying the ifft method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ifft'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the ifft method can modify the input AO.	Test that the ifft method can modify the input AO by calling with no output.	pass
		1) Check that 'at2' and 'ain' are now different. 2) Check that 'ain' is ifft(at2).	pass

ao/ifft			
08	Tests that the ifft method keeps the data shape of the input object.	Test that the ifft method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the ifft method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the ifft method agrees with MATLAB's ifft when configured to use the same parameters.	Test that the applying ifft works on two AOs.	pass
		1) Check that output agrees with the output of MATLAB's ifft.	pass
11	Check that the [ao/ifft] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests that the ifft method agrees with MATLAB's ifft when configured to use the same parameters.	Test that the applying ifft works on a single AO with 'nonsymmetric' option.	pass
		1) Check that output agrees with the output of MATLAB's ifft.	pass

Table 56: Unit tests for ao/ifft.

ao/imag			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the imag method works with a vector of AOs as input.	Test that the imag method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the imag method works with a matrix of AOs as input.	Test that the imag method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the imag method works with a list of AOs as input.	Test that the imag method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the imag method works with a mix of different shaped AOs as input.	Test that the imag method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the imag method properly applies history.	Test that the result of applying the imag method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'imag'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the imag method can modify the input AO.	Test that the imag method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is imag(at1).	pass

ao/imag			
08	Control the method with a plist.	Test that the imag method can modify the single axis controlled by the plist and the resuld can be processed back to an m-file.	pass
		1) Check that the imag method applies to the x-axis 2) Check that the imag method applies to the y-axis 3) Check that the imag method applies to both axes 4) Check that the re-built objects are the same object as 'out[1..3]'.	pass
09	Control the method with a plist.	Test that the imag method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the imag method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/imag] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 57: Unit tests for ao/imag.

ao/index			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the index method works with a vector of AOs as input.	Test that the index method works for a vector of AOs as input. The following indexing should work: I = [1 2 3] or (I/J) = [(1,1), (1,2), (1,3)]	pass
		1) Check that the index method selects the correct object.	pass
03	Tests that the index method works with a matrix of AOs as input.	Test that the index method works for a matrix of AOs as input. The following indexing should work: I = [1 3 5] or (I/J) = [(1,1), (1,2), (1,3)] [2 4 6] [(2,1), (2,2), (2,3)]	pass
		1) Check that the index method selects the correct object.	pass
04	Tests that the index method works with a list of AOs as input.	The index method doesn't work for a list of AOs as input.	pass
		1) Nothing to test.	pass
05	Tests that the index method properly applies history.	Test that the result of index have an additional history step.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'index'.	pass
06	Tests that the index method works for the modifier command.	Tests that the index method works for the modifier command.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
07	Control the method with a plist.	Test that the index method can be controled with a plist.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
08	Test that the index method selects more objects if I have more indices.	Test that the index method selects more objects if I have more indices.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass

ao/index			
11	Check that the [ao/index] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 58: Unit tests for ao/index.

ao/integrate			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the integrate method works with a vector of AOs as input.	Test that the integrate method works for a vector of AOs as input. Use for this test the trapezoidal method.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the integrate method works with a matrix of AOs as input.	Test that the integrate method works for a matrix of AOs as input. Use for this test the Trapezoidal method.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the integrate method works with a list of AOs as input.	Test that the integrate method works for a list of AOs as input. Use for this test the trapezoidal.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the integrate method works with a mix of integrateerent shaped AOs as input.	Test that the integrate method works with an input of matrices and vectors and single AOs. Use for this test the trapezoidal.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the integrate method properly applies history.	Test that the result of applying the integrate method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'integrate'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/integrate			
07	Tests that the integrate method can modify the input AO.	Test that the integrate method can modify the input AO by calling with no output. Use for this test the trapezoidal.	pass
		1) Check that 'at1' and 'ain' are now integrateerent. 2) Check that 'ain' is integrate(at1).	pass
08	Tests that the integrate method keeps the data shape of the input object.	Test that the integrate method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the integrate method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Control the method with a plist.	Test the trapezoidal.	pass
		1) Check that the integrate method uses the trapezoidal method. 2) Check that the rebuilt object is the same object as 'out'.	pass
11	Check that the [ao/integrate] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 59: Unit tests for ao/integrate.

ao/interp			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the interp method works with a vector of AOs as input.	Test that the interp method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the interp method works with a matrix of AOs as input.	Test that the interp method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the interp method works with a list of AOs as input.	Test that the interp method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the interp method works with a mix of different shaped AOs as input.	Test that the interp method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the interp method properly applies history.	Test that the result of applying the interp method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'interp'. 2) Check that the rebuilt object is the same as 'out'.	pass
07	Tests that the interp method can modify the input AO.	Test that the interp method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is interp(at1).	pass

ao/interp			
08	Tests that the interp method keeps the data shape of the input object.	Test that the interp method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the interp method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the interp method can different interpolations.	Test that the interp method can all of MATLAB interpolates methods. 'nearest' - Nearest neighbor interpolation 'linear' - Linear interpolation 'spline' - Cubic spline interpolation (see UTPs above) 'pchip' - Piecewise cubic Hermite interpolation	pass
		1) Check the different interpolations 2) Check that the re-built objects are the same as 'out1..3'.	pass
11	Check that the [ao/interp] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 60: Unit tests for ao/interp.

ao/interpmissing			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the interpmissing method works with a vector of AOs as input.	Test that the interpmissing method works for a vector of AOs as input. Known gaps at the position idx = 30 and 51	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the interpmissing method works with a matrix of AOs as input.	Test that the interpmissing method works for a matrix of AOs as input. Known gaps at the position idx = 30 and 51	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the interpmissing method works with a list of AOs as input.	Test that the interpmissing method works for a list of AOs as input. Known gaps at the position idx = 30 and 51	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the interpmissing method works with a mix of different shaped AOs as input.	Test that the interpmissing method works with an input of matrices and vectors and single AOs. Known gaps at the position idx = 30 and 51	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the interpmissing method properly applies history.	Test that the result of applying the interpmissing method can be processed back to an m-file.	pass

ao/interpmissing			
		1) Check that the last entry in the history of 'out' corresponds to 'interpmissing'. 2) Check that the re-built object is the same as 'out'.	pass
07	Tests that the interpmissing method can modify the input AO.	Test that the interpmissing method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign). Known gaps at the position idx = 30 and 51	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is interpmissing(at1).	pass
08	Control the method with a plist.	Test that the interpmissing method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the interpmissing method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the interpmissing method can change the tolerance for finding missing samples.	Test that the interpmissing method works with a plist which changes the tolerance. Known gaps at the position idx = 30 and 51 with the width of 40 and 60	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data. 3) Check that the re-built object is the same as 'out'.	pass
11	Check that the [ao/interpmissing] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass

ao/interpmissing			
		1) Check that the output contains the same plotinfo plist	pass

Table 61: Unit tests for ao/interpmissing.

ao/inv			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the inv method works with a vector of AOs as input.	Test that the inv method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the inv method works with a matrix of AOs as input.	Test that the inv method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the inv method works with a list of AOs as input.	Test that the inv method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the inv method works with a mix of different shaped AOs as input.	Test that the inv method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the inv method properly applies history.	Test that the result of applying the inv method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'inv'. 2) Check that the rebuilt object is the same as 'out'.	pass
07	Tests that the inv method can modify the input AO.	Test that the inv method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at4' and 'ain' are now different. 2) Check that 'ain' is inv(at4).	pass

ao/inv			
08	Control the method with a plist.	Test that the inv method can modify the single axis controlled by the plist and the resuld can be processed back to an m-file.	pass
		1) Check that the inv method applies to the x-axis 2) Check that the inv method applies to the y-axis 3) Check that the inv method applies to both axes 4) Check that the re-built objects are the same as 'out1..3'.	pass
09	Control the method with a plist.	Test that the inv method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the inv method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/inv] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 62: Unit tests for ao/inv.

ao/isprop			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isprop method works with a vector of AOs as input.	Test that the isprop method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output contains the correct data.	pass
03	Tests that the isprop method works with a matrix of AOs as input.	Test that the isprop method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output contains the correct data.	pass
04	Tests that the isprop method works with a list of AOs as input.	Test that the isprop method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the isprop method works with a mix of different shaped AOs as input.	Test that the isprop method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the isprop method properly applies history.	The method isprop doesn't change the object, thus it is not necessary to apply history.	pass
			pass
07	Tests that the isprop method works for each property.	Test that the isprop method works for the properties: 'data', 'mfile', 'mfilename', 'mdlfile', 'mdlfilename', 'procinfo', 'plotinfo', 'description', 'hist', 'name'	pass
		1) Check that each output contains the correct data.	pass
08	Test the negatice case and the not function command.	Test that the isprop method retrun false for a unknown property and for methods of the object.	pass

ao/isprop			
		1) Check that each output contains the correct data.	pass

Table 63: Unit tests for ao/isprop.

join/fsdata			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the join method works with a vector of AOs as input.	Test that the join method works for a vector of AOs as input.	pass
		1) Check that the output is exact one AO. 2) Check that the output have the correct data. 3) Check the re-built object	pass
03	Tests that the join method works with a matrix of AOs as input.	Tests that the join method works with a matrix of AOs as input.	pass
		1) Check that the output is exact one AO. 2) Check that the output have the correct data. 3) Check the re-built object	pass
04	Tests that the join method works with a list of AOs as input.	Tests that the join method works with a list of AOs as input.	pass
		1) Check that the output is exact one AO. 2) Check that the output have the correct data. 3) Check the re-built object	pass
05	Tests that the join method works with a mix of different shaped AOs as input.	Tests that the join method works with a mix of different shaped AOs as input.	pass
		1) Check that the output is exact one AO. 2) Check that the output have the correct data. 3) Check the re-built object	pass
06	Tests that the join method properly applies history.	Test that the result of applying the join method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'join'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the join method can modify the input AO.	Test that the join method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

join/fsdata			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is joined 4) Check that out and amodi are the same	pass
08	Test the shape of the output.	Test that the join method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
11	Check that the [join/fsdata] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 64: Unit tests for join/fsdata.

join/tsdata			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the join method works with a vector of AOs as input.	Test that the join method works for a vector of AOs as input.	pass
		1) Check that the output is exact one AO. 2) Check that the output have the correct data. 3) Check the re-built object	pass
03	Tests that the join method works with a matrix of AOs as input.	Test that the join method works for a matrix of AOs as input.	pass
		1) Check that the output is exact one AO. 2) Check that the output have the correct data. 3) Check the re-built object	pass
04	Tests that the join method works with a list of AOs as input.	Tests that the join method works with a list of AOs as input.	pass
		1) Check that the output is exact one AO. 2) Check that the output have the correct data. 3) Check the re-built object	pass
05	Tests that the join method works with a mix of different shaped AOs as input.	Tests that the join method works with a mix of different shaped AOs as input.	pass
		1) Check that the output is exact one AO. 2) Check that the output have the correct data. 3) Check the re-built object	pass
06	Tests that the join method properly applies history.	Test that the result of applying the join method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'join'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the join method can modify the input AO.	Test that the join method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

join/tsdata			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is joined 4) Check that out and amodi are the same	pass
08	Test that the join method fills the gaps with zeros.	Test that the join method fills the gaps with zeros.	pass
		1) Check that the join method filled the gaps with zero 2) Check that the re-built objects are the same as out[1..2]	pass
09	Test the shape of the output.	Test that the join method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Test the x-values.	Test the x-values.	pass
		1) Check the x-values for the case where we 'zerofill' or not. 2) Compare the outputs to see that switching the order did not matter, because if the different t0 of the data, that get sorted anyways.	pass
11	Check that the [join/tsdata] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 65: Unit tests for join/tsdata.

ao/lcohere			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the lcohere method works with a vector of AOs as input.	Test that the lcohere method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the lcohere method doesn't work with a matrix of AOs as input.	Test that the lcohere method doesn't work for a matrix of AOs as input.	pass
		1) Nothing to do	pass
04	Tests that the lcohere method works with a list of AOs as input.	Test that the lcohere method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the lcohere method doesn't work with a mix of different shaped AOs as input.	Test that the lcohere method doesn't work with an input of matrices and vectors and single AOs.	pass
		1) Nothing to check.	pass
06	Tests that the lcohere method properly applies history.	Test that the result of applying the lcohere method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'lcohere'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the lcohere method can not modify the input AO.	Test that the lcohere method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
08	Test the shape of the output.	Test that the plus method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass

ao/lcohere			
		1) Check that the shpe of the output data doesn't change.	pass
09	Check that the lcohere method pass back the output objects to a list of output variables or to a single variable.	This test is not longer necessary because the cohere method pass back always only one object.	pass
		1) Nothing to check.	pass
11	Check that the [ao/lcohere] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests symmetry properties of complex-coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) complex lcoherence of the white noise series 4) compare C(x,y) with conj(C(y,x)) 5) compare C(x,x) and C(y,y) with 1	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) complex log-scale coherence of the white noise	pass
		1) Check that C(x,y) equals conj(C(y,x)) 2) Check that C(x,x) equals 1 2) Check that C(y,y) equals 1	pass
13	Tests symmetry properties of magnitude-squared log-scale coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) magnitude-squared log-scale coherence of the white noise series 4) compare C(x,y) with C(y,x) 5) compare C(x,x) and C(y,y) with 1	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) magnitude-squared log-scale coherence of the white noise	pass
		1) Check that C(x,y) equals C(y,x) 1) Check that C(x,x) equals 1 1) Check that C(y,y) equals 1	pass
14	Tests symmetry properties of complex-coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) complex log-scale coherence of the combination of white noise series 4) compare C(x,y) with 1	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) complex log-scale coherence of the combination of noise	pass
		1) Check that the complex coherence equals 1	pass

ao/lcohere			
15	Tests symmetry properties of complex-coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) magnitude-squared log-scale coherence of the	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) magnitude-squared log-scale coherence of the combination of noise	pass
		1) Check that the magnitude-squared coherence equals 1	pass
16	Tests symmetry properties of complex-coherence: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) magnitude-squared log-scale coherence M of the combination of white noise series 4) complex log-scale coherence C of the combination	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) magnitude-squared log-scale coherence of the combination of noise 6) complex log-scale coherence of the combination of noise	pass
		1) Check that the magnitude-squared coherence equals the square modulus of the complex coherence	pass
17	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) complex cohere of the white noise series 4) compares the units of the input and	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) complex cohere of the white noise	pass
		1) Check that (complex coherence yunits) equals [1] 2) Check that (complex coherence xunits) equals [Hz]	pass
18	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) magnitude-squared cohere of the white noise series 4) compares the units of the input and output	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) magnitude-squared cohere of the white noise	pass
		1) Check that (magnitude-squared coherence yunits) equals [1] 2) Check that (magnitude-squared coherence xunits) equals [Hz]	pass

ao/lcohere			
30	Tests handling of special cases: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) the same noise series 3) lcohere of the white noise series 4) compares the output to unity	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: the same data as 1) and 2) 4) lcohere of the series	pass
		1) Check that calculated lcohere equals 1	pass

Table 66: Unit tests for ao/lcohere.

ao/lcpsd			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the lcpsd method works with a vector of AOs as input.	Test that the lcpsd method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the lcpsd method doesn't work with a matrix of AOs as input.	Test that the lcpsd method doesn't work for a matrix of AOs as input.	pass
		1) Nothing to check.	pass
04	Tests that the lcpsd method works with a list of AOs as input.	Test that the lcpsd method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the lcpsd method doesn't work with a mix of different shaped AOs as input.	Test that the lcpsd method doesn't work with an input of matrices and vectors and single AOs.	pass
		1) Nothing to check.	pass
06	Tests that the lcpsd method properly applies history.	Test that the result of applying the lcpsd method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'lcpsd'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the lcpsd method can not modify the input AO.	Test that the lcpsd method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
08	Test the shape of the output.	Test that the plus method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass

ao/lcpsd			
		1) Check that the shpe of the output data doesn't change.	pass
09	Check that the lcpsd method pass back the output objects to a list of output variables or to a single variable.	This test is not longer necessary because the cohere method pass back always only one object.	pass
		1) Nothing to check	pass
11	Check that the [ao/lcpsd] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests that differently sized data sets are treated properly	Test that applying lcpsd works on two AOs.	pass
		1) Check that lcpsd used the length of the shortest ao.	pass
17	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) LCPSD of the white noise series 4) compares the	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) LCPSD of the white noise	pass
		1) Check that (calculated LCPSD yunits) equals input_1 units*input_2 units/Hz	pass
18	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) LCPSD of the white noise series Comparison with LPSD: 4) compares the off-diagonal terms to check they are complex-conjugated 5) compares the diagonal terms with PSD of the individual noise	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) LCPSD of the white noise 6) LPSD of the white noise	pass
		1) Check that LCPSD(x,x) equals LPSD(x) 2) Check that LCPSD(y,y) equals LPSD(y) 3) Check that LCPSD(x,y) equals conj(LCPSD(y,x))	pass

Table 67: Unit tests for ao/lcpsd.

ao/le			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the le method compare an AO with scalar value	Test that the le method works with relational operators and the function command. Use for this AOs with different data objects.	pass
		1) Check that the result of the 'relational operator' and the 'function' command are the same. 2) Check that each output AO contains the correct data.	pass
03	Tests that the le method compare an AO with one other AO	Test that the le method works with relational operators and the function command. Use for this AOs with different data objects. Remark that both AOs must have the same size.	pass
		1) Check that the result of the 'relational operator' and the 'function' command are the same. 2) Check that each output AO contains the correct data.	pass

Table 68: Unit tests for ao/le.

ao/len			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the len method works with a vector of AOs as input.	Test that the len method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output contains the correct data.	pass
03	Tests that the len method works with a matrix of AOs as input.	Test that the len method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the len method works with a list of AOs as input.	Test that the len method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the len method works with a mix of different shaped AOs as input.	Test that the len method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the len method properly applies history.	The len method doesn't change the object, thus it is not necessary to test the history. Nothing to do.	pass
			pass

Table 69: Unit tests for ao/len.

ao/linSubtract			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the linSubtract method works with a vector of AOs as input.	Test that the linSubtract method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the linSubtract method works with a matrix of AOs as input.	Test that the linSubtract method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the linSubtract method works with a list of AOs as input.	Test that the linSubtract method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the linSubtract method works with a mix of different shaped AOs as input.	Test that the linSubtract method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the linSubtract method properly applies history.	Test that the result of applying the linSubtract method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'linSubtract'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the linSubtract method can not modify the input AO.	Test that the tfe method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass

ao/linSubtract			
11	Check that the [ao/linSubtract] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 70: Unit tests for ao/linSubtract.

ao/lincom			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the lincom method works with a vector of AOs as input.	Test that the lincom method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1 2) Check that the output AO contains the correct units 3) Check that the output AO contains the correct data.	pass
04	Tests that the lincom method works with a list of AOs as input.	Test that the lincom method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that the output AO contains the correct units 3) Check that the output AO contains the correct data.	pass
06	Tests that the lincom method properly applies history.	Test that the result of applying the lincom method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'lincom'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the lincom method can not modify the input AO.	Test that the lincom method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
08	Test the shape of the output.	Test that the plus method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the output data doesn't change.	pass
11	Check that the [ao/lincom] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass

ao/lincom			
		1) Check that the output contains the same plotinfo plist	pass
12	Tests that the lincom method works with a vector of AOs as input + a pest object.	Test that the lincom method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1 2) Check that the output AO contains the correct units 3) Check that the output AO contains the correct data.	pass
14	Tests that the lincom method works with a list of AOs as input + a pest object.	Test that the lincom method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that the output AO contains the correct units 3) Check that the output AO contains the correct data.	pass

Table 71: Unit tests for ao/lincom.

ao/ln			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/ln] method works with a vector of objects as input.	Test that the [ao/ln] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/ln] method works with a matrix of objects as input.	Test that the [ao/ln] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/ln] method works with a list of objects as input.	Test that the [ao/ln] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/ln] method works with a mix of different arrays of objects as input.	Tests that the [ao/ln] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/ln] method properly applies history.	Test that the result of applying the [ao/ln] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/ln]'. 2) Check that the rebuilt object is the same object as the input.	pass
07	Tests that the [ao/ln] method can modify the input AO.	Test that the [ao/ln] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/ln			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/ln] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/ln] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/ln] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/ln] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 72: Unit tests for ao/ln.

ao/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 73: Unit tests for ao/loadobj.

ao/log			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/log] method works with a vector of objects as input.	Test that the [ao/log] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/log] method works with a matrix of objects as input.	Test that the [ao/log] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/log] method works with a list of objects as input.	Test that the [ao/log] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/log] method works with a mix of different arrays of objects as input.	Tests that the [ao/log] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/log] method properly applies history.	Test that the result of applying the [ao/log] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/log]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/log] method can modify the input AO.	Test that the [ao/log] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/log			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/log] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/log] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/log] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/log] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 74: Unit tests for ao/log.

ao/log10			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/log10] method works with a vector of objects as input.	Test that the [ao/log10] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/log10] method works with a matrix of objects as input.	Test that the [ao/log10] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/log10] method works with a list of objects as input.	Test that the [ao/log10] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/log10] method works with a mix of different arrays of objects as input.	Tests that the [ao/log10] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/log10] method properly applies history.	Test that the result of applying the [ao/log10] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/log10]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/log10] method can modify the input AO.	Test that the [ao/log10] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/log10			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/log10] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/log10] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/log10] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/log10] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 75: Unit tests for ao/log10.

ao/lpsd			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the lpsd method works with a vector of AOs as input.	Test that the lpsd method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output object contains the correct values.	pass
03	Tests that the lpsd method works with a matrix of AOs as input.	Test that the lpsd method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output object contains the correct values.	pass
04	Tests that the lpsd method works with a list of AOs as input.	Test that the lpsd method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the lpsd method works with a mix of different shaped AOs as input.	Test that the lpsd method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the lpsd method properly applies history.	Test that the result of applying the lpsd method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'lpsd'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Check that the lpsd method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass

ao/lpsd			
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
08	Tests that the lpsd method agrees with MATLAB's pwelch if the frequency resolution of the latter is changed to match those used in lpsd	Test that the applying psd works on a single AO.	pass
		1) Check that the DFT equations (eq. (3) and (4) in [1]) are fulfilled. 2) Check that we get the same outputs for each frequency bin when computing the psd using matlab's fft with the frequency resolution values retrieved by lpsd (and already tested in (1)) [1] G. Heinzel, lpsd revisited: ltf, S2-AEI-TN-3052	pass
11	Check that the [ao/lpsd] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
17	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) LPSD of the white noise 3) compares the units of the input and output	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) LPSD of the white noise	pass
		1) Check that (calculated LPSD yunits) equals (input units)$\hat{}$2 / Hz 2) Check that (calculated LPSD xunits) equals Hz	pass
18	Tests handling of units: 1) white noise produced from uniform pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) LASD of the white noise 3) compares the units of the input and output	1) Prepare the test tsdata: white noise from uniform distribution + offset 2) Assign a random unit 3) LASD of the white noise	pass
		1) Check that (calculated LASD yunits) equals (input units) / Hz$\hat{}$(1/2) 2) Check that (calculated LASD xunits) equals Hz	pass
19	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) LPS of the white noise 3) compares the units of the input and output	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) LPS of the white noise	pass
		1) Check that (calculated LPS yunits) equals (input units)$\hat{}$2 2) Check that (calculated LPS xunits) equals Hz	pass

ao/lpsd			
20	Tests handling of units: 1) white noise produced from uniform distribution, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) LAS of the white noise 3) compares the units of the input and output	1) Prepare the test tsdata: white noise from uniform distribution + offset 2) Assign a random unit 3) LAS of the white noise	pass
		1) Check that (calculated LAS yunits) equals (input units) 2) Check that (calculated LAS xunits) equals Hz	pass
41	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (Rectangular) 3) lpsd passing the window object (Rectangular type) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Rectangular window (name) 4) Estimate the psd without detrending, Rectangular window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass
42	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (BH92) 3) lpsd passing the window object (BH92 type) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, BH92 window (name) 4) Estimate the psd without detrending, BH92 window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass
43	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (Hamming) 3) lpsd passing the window object (Hamming type) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Hamming window (name) 4) Estimate the psd without detrending, Hamming window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass
44	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (Hanning) 3) lpsd passing the window object (Hanning type) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Hanning window (name) 4) Estimate the psd without detrending, Hanning window (object) 5) Compare results	pass

ao/lpsd			
		1) Check that calculated psds are identical	pass
45	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (Bartlett) 3) lpsd passing the window object (Bartlett type) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Bartlett window (name) 4) Estimate the psd without detrending, Bartlett window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass
46	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (Nuttall3) 3) lpsd passing the window object (Nuttall3 type) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Nuttall3 window (name) 4) Estimate the psd without detrending, Nuttall3 window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass
47	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (Kaiser, psll = 150) 3) lpsd passing the window object (Kaiser type, psll = 150) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Kaiser window (name) 4) Estimate the psd without detrending, Kaiser window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass
48	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (Kaiser, psll = default) 3) lpsd passing the window object (Kaiser type, psll = default) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Kaiser window (name) 4) Estimate the psd without detrending, Kaiser window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass

ao/lpsd			
49	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (Nuttall4) 3) lpsd passing the window object (Nuttall4 type) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Nuttall4 window (name) 4) Estimate the psd without detrending, Nuttall4 window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass
50	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) lpsd passing the window name (SFT3F) 3) lpsd passing the window object (SFT3F type) 4) compares the two psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, SFT3F window (name) 4) Estimate the psd without detrending, SFT3F window (object) 5) Compare results	pass
		1) Check that calculated psds are identical	pass

Table 76: Unit tests for ao/lpsd.

ao/lscov			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the lscov method works with a vector of AOs as input.	Test that the lscov method works for a vector of AOs as input.	pass
		1) Check the data type of the output 2) Check that each output AO contains the correct data.	pass
03	Tests that the lscov method works with a matrix of AOs as input.	Tests that the lscov method works with a matrix of AOs as input.	pass
		1) Check the data type of the output 2) Check that each output AO contains the correct data.	pass
04	Tests that the lscov method works with a list of AOs as input.	Tests that the lscov method works with a list of AOs as input.	pass
		1) Check the data type of the output 2) Check that each output AO contains the correct data.	pass
05	Tests that the lscov method properly applies history.	Test that the result of applying the lscov method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'lscov'. 2) Check that the rebuilt object is the same object as the input.	pass
06	The lscov method can not be used as a modifer method.	The lscov method can not be used as a modifer method.	pass
		1) Nothing to check.	pass
07	Check that the lscov method uses weights for the fit.	Check that the lscov method uses weights for the fit.	pass
		1) Check the output data 2) Check the yunits 3) Check that 'out1' and 'out2' have the same data 4) Check that 'out3' and 'out4' have the same data 5) Check the re-built objects	pass

ao/lscov			
11	Check that the [ao/lscov] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 77: Unit tests for ao/lscov.

ao/lt			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the lt method compare an AO with scalar value	Test that the lt method works with relational operators and the function command. Use for this AOs with different data objects.	pass
		1) Check that the result of the 'relational operator' and the 'function' command are the same. 2) Check that each output AO contains the correct data.	pass
03	Tests that the lt method compare an AO with one other AO	Test that the lt method works with relational operators and the function command. Use for this AOs with different data objects. Remark that both AOs must have the same size.	pass
		1) Check that the result of the 'relational operator' and the 'function' command are the same. 2) Check that each output AO contains the correct data.	pass

Table 78: Unit tests for ao/lt.

ao/ltfe			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ltfe method works with a vector of AOs as input.	Test that the ltfe method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the ltfe method doesn't work with a matrix of AOs as input.	Test that the ltfe method doesn't work for a matrix of AOs as input.	pass
		1) Nothing to check.	pass
04	Tests that the ltfe method works with a list of AOs as input.	Test that the ltfe method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the ltfe method doesn't work with a mix of different shaped AOs as input.	Test that the ltfe method doesn't work with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the ltfe method properly applies history.	Test that the result of applying the ltfe method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ltfe'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the ltfe method can not modify the input AO.	Test that the ltfe method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
08	Test the shape of the output.	Test that the ltfe method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass

ao/ltfe			
		1) Check that the shpe of the output data doesn't change.	pass
09	Check that the ltfe method pass back the output objects to a list of output variables or to a single variable.	This test is not longer necessary because the ltfe method pass back always only one object.	pass
		1) Nothing to check	pass
11	Check that the [ao/ltfe] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
17	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) Ltfe of the white	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) Ltfe of the white noise	pass
		1) Check that (calculated ltfe yunits) equals [1/Hz]	pass
30	Tests handling of special cases: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) the same noise series 3) ltfe of the white noise series 4) compares the output to unity	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: the same data as 1) and 2) 4) ltfe of the series	pass
		1) Check that calculated ltfe equals 1	pass

Table 79: Unit tests for ao/ltfe.

ao/max			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/max] method works with a vector of objects as input.	Test that the [ao/max] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/max] method works with a matrix of objects as input.	Test that the [ao/max] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/max] method works with a list of objects as input.	Test that the [ao/max] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/max] method works with a mix of different arrays of objects as input.	Tests that the [ao/max] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/max] method properly applies history.	Test that the result of applying the [ao/max] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/max]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/max] method can modify the input AO.	Test that the [ao/max] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/max			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/max] value of the copy 4) Check that out and amodi are the same	pass
208	Test that the [ao/max] method uses the plist to get the axis. This is intended to test methods like ao/max and ao/min which only allow 'x' and 'y' choices.	Test that the [ao/max] method uses the plist to get the axis.	pass
		1) Check that the [ao/max] method applies to the x-axis 2) Check that the [ao/max] method applies to the y-axis 3) Check that the [ao/max] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/max] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/max] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/max] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 80: Unit tests for ao/max.

ao/mcmc			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the metropolis2D method works with a vector of AOs as input.	Test that the metropolis2D method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the metropolis2D method works with a matrix of AOs as input.	Test that the metropolis2D method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the metropolis2D method works with a list of AOs as input.	Test that the metropolis2D method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the metropolis2D method works with a mix of different shaped AOs as input.	Test that the metropolis2D method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the metropolis2D method properly applies history.	Test that the result of applying the metropolis2D method can be processed back to an m-file.	pass

ao/mcmc			
		1) Check that the last entry in the history of 'm' corresponds to 'metropolis2D'. 2) Check that the re-built object is the same object as 'm'. THIS IS NOT TRUE FOR METROPOLIS BECAUSE THE OUTPUT CHAIN IS ONLY EQUAL STATISTICALLY	pass
07	Tests that the metropolis2D method can not modify the input AO.	Test that the tfe method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass

Table 81: Unit tests for ao/mcmc.

ao/md5			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the md5 method works with a vector of AOs as input.	Test that the md5 method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' are the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the md5 method works with a matrix of AOs as input.	Tests that the md5 method works with a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' are the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the md5 method works with a list of AOs as input.	Tests that the md5 method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' are the same as in the input 2) Check that each output AO contains the correct data.	pass
05	Tests that the md5 method works with a mix of different shaped AOs as input.	Tests that the md5 method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' are the same as in the input 2) Check that each output AO contains the correct data.	pass
06	Tests that the md5 method retruns for one input a string and not a cell of a string	Special case for one input because in this case retruns md5 a string and not a cell.	pass
		1) Check that the output is a string 2) Check that each output AO contains the correct data.	pass

Table 82: Unit tests for ao/md5.

ao/mean			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/mean] method works with a vector of objects as input.	Test that the [ao/mean] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/mean] method works with a matrix of objects as input.	Test that the [ao/mean] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/mean] method works with a list of objects as input.	Test that the [ao/mean] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/mean] method works with a mix of different arrays of objects as input.	Tests that the [ao/mean] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/mean] method properly applies history.	Test that the result of applying the [ao/mean] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/mean]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/mean] method can modify the input AO.	Test that the [ao/mean] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/mean			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/mean] value of the copy 4) Check that out and amodi are the same	pass
108	Test that the [ao/mean] method uses the plist to get the axis. This is intended to test methods like ao/mean and ao/std which return different data types depending on which axis is selected.	Test that the [ao/mean] method uses the plist to get the axis.	pass
		1) Check that the [ao/mean] method applies to the x-axis 2) Check that the [ao/mean] method applies to the y-axis 3) Check that the [ao/mean] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/mean] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/mean] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/mean] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 83: Unit tests for ao/mean.

ao/median			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/median] method works with a vector of objects as input.	Test that the [ao/median] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/median] method works with a matrix of objects as input.	Test that the [ao/median] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/median] method works with a list of objects as input.	Test that the [ao/median] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/median] method works with a mix of different arrays of objects as input.	Tests that the [ao/median] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/median] method properly applies history.	Test that the result of applying the [ao/median] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/median]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/median] method can modify the input AO.	Test that the [ao/median] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/median			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/median] value of the copy 4) Check that out and amodi are the same	pass
108	Test that the [ao/median] method uses the plist to get the axis. This is intended to test methods like ao/mean and ao/std which return different data types depending on which axis is selected.	Test that the [ao/median] method uses the plist to get the axis.	pass
		1) Check that the [ao/median] method applies to the x-axis 2) Check that the [ao/median] method applies to the y-axis 3) Check that the [ao/median] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/median] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/median] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/median] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 84: Unit tests for ao/median.

ao/min			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/min] method works with a vector of objects as input.	Test that the [ao/min] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/min] method works with a matrix of objects as input.	Test that the [ao/min] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/min] method works with a list of objects as input.	Test that the [ao/min] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/min] method works with a mix of different arrays of objects as input.	Tests that the [ao/min] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/min] method properly applies history.	Test that the result of applying the [ao/min] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/min]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/min] method can modify the input AO.	Test that the [ao/min] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/min			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/min] value of the copy 4) Check that out and amodi are the same	pass
208	Test that the [ao/min] method uses the plist to get the axis. This is intended to test methods like ao/max and ao/min which only allow 'x' and 'y' choices.	Test that the [ao/min] method uses the plist to get the axis.	pass
		1) Check that the [ao/min] method applies to the x-axis 2) Check that the [ao/min] method applies to the y-axis 3) Check that the [ao/min] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/min] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/min] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/min] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 85: Unit tests for ao/min.

ao/minus			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
rule1 (tsdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass

ao/minus			
rule1 (cdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single tsdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single fsdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector cdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector cdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single tsdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single fsdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP cdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ tsdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ fsdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP cdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/minus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
minus			pass
			pass
tests (fsdata and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (AO no data and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and AO no data)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different fs in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x units in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x values in fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

ao/minus			
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

Table 86: Unit tests for ao/minus.

ao/mode			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/mode] method works with a vector of objects as input.	Test that the [ao/mode] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/mode] method works with a matrix of objects as input.	Test that the [ao/mode] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/mode] method works with a list of objects as input.	Test that the [ao/mode] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/mode] method works with a mix of different arrays of objects as input.	Tests that the [ao/mode] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/mode] method properly applies history.	Test that the result of applying the [ao/mode] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/mode]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/mode] method can modify the input AO.	Test that the [ao/mode] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/mode			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/mode] value of the copy 4) Check that out and amodi are the same	pass
108	Test that the [ao/mode] method uses the plist to get the axis. This is intended to test methods like ao/mean and ao/std which return different data types depending on which axis is selected.	Test that the [ao/mode] method uses the plist to get the axis.	pass
		1) Check that the [ao/mode] method applies to the x-axis 2) Check that the [ao/mode] method applies to the y-axis 3) Check that the [ao/mode] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/mode] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/mode] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/mode] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 87: Unit tests for ao/mode.

ao/mpower			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the mpower method works with a vector of AOs as input.	Test that the mpower method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1 2) Check that each output AO contains the correct data. 3) Check the y-units	pass
03	Tests that the mpower method works with a matrix of AOs as input.	Test that the mpower method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1 2) Check that each output AO contains the correct data.	pass
04	Tests that the mpower method works with a list of AOs as input.	Test that the mpower method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that each output AO contains the correct data. 3) Check the y-units	pass
05	Tests that the mpower method works with a mix of different shaped AOs as input.	Test that the mpower method works with an input of matrices and vectors and single AOs.	pass
		1) Nothing to check	pass
06	Tests that the mpower method properly applies history.	Test that the result of applying the mpower method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' and 'out2' corresponds to 'mpower'. 2) Check that the re-built objects are the same objectd as 'out1' and 'out2'.	pass
07	Tests that the mpower method can modify the input AO.	Test that the mpower method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at4' and 'ain' are now different. 2) Check that 'ain' is mpower(at4).	pass

ao/mpower			
08	Test the shape of the output.	Test that the mpower method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Nothinf to check	pass

Table 88: Unit tests for ao/mpower.

ao/mrdivide			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
rule1 (tsdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass

ao/mrdivide			
rule1 (cdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single tsdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single fsdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector cdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector cdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single tsdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single fsdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP cdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ tsdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ fsdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP cdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/mrdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rdivide			pass
			pass
tests (fsdata and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (AO no data and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and AO no data)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different fs in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x values in fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

ao/mrdivide			

Table 89: Unit tests for ao/mrdivide.

ao/mtimes			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
04	Tests that the mtimes method works with a list of AOs as input.	Test that the mtimes method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that each output AO contains the correct data. 3) Check the y-units	pass
06	Tests that the mtimes method properly applies history.	Test that the result of applying the mtimes method can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mtimes'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the mtimes method can not be used as a modifier method.	Tests that the mtimes method can not be used as a modifier method. The command should fail.	pass
		1) Nothing to test.	pass
08	Test the shape of the output.	Test that the mtimes method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Test the method with all data objects.	Test that the mtimes method works with cdata-, fsdata-, tsdata-, and xydata objects	pass
		1) Check that the shpe of the data doesn't change. 2) Check that re-building of output is the same as the output	pass

Table 90: Unit tests for ao/mtimes.

ao/ne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ne method works with a vector of AOs as input.	Test that the ne method works for a vector of AOs as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
03	Tests that the ne method works with a matrix of AOs as input.	Test that the ne method works for a matrix of AOs as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
04	Tests that the ne method works with a list of AOs as input.	The ne method doesn't works for a list of AOs as input. Nothing to do.	pass
			pass
05	Tests that the ne method works with a mix of different shaped AOs as input.	The ne method doesn't works for a list of AOs as input. Nothing to do.	pass
			pass
06	Tests that the ne method properly applies history.	The ne method doesn't change the AO, thus will no history added. Nothing to do	pass
			pass
07	Test the ne method with an exception list. The function ao/ne use the function ao/eq so it is not necessary to check all possibilities of the exception list.	Test the ne method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because aa is created at an other time.	pass
		1) Check the output.	pass
08	Test the ne method with an exception list which is in a plist.	Test that the ne method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 91: Unit tests for ao/ne.

ao/noisegen1D			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the noisegen1D method works with a vector of AOs as input.	Test that the noisegen1D method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'av' 2) Check that each output AO contains the correct data.	pass
03	Tests that the noisegen1D method works with a matrix of AOs as input.	Test that the noisegen1D method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the noisegen1D method works with a list of AOs as input.	Test that the noisegen1D method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the noisegen1D method works with a mix of different shaped AOs as input.	Test that the noisegen1D method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the noisegen1D method properly applies history.	Test that the result of applying the noisegen1D method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'noisegen1D'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the noisegen1D method can modify the input AO.	Test that the noisegen1D method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/noisegen1D			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is changed	pass
08	Test the shape of the output.	Test that the noisegen1D method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the noisegen1D method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the noisegen1D method properly applies history for set filter.	Test that the result of applying the noisegen1D method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'noisegen1D'. 2) Check that the re-built object is the same object as the input.	pass
11	Tests that the noisegen1D method can modify the input AO for set 'filter'.	Test that the noisegen1D method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed	pass
12	Check that the noisegen1D method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 92: Unit tests for ao/noisegen1D.

ao/noisegen2D			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the noisegen2D method works with a vector of AOs as input.	Test that the noisegen2D method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'acv2' 2) Check that each output AO contains the correct data.	pass
03	Tests that the noisegen2D method works with a matrix of AOs as input.	Test that the noisegen2D method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the noisegen2D method works with a list of AOs as input.	Test that the noisegen2D method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the noisegen2D method works with a mix of different shaped AOs as input.	Test that the whiten1D method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the noisegen2D method properly applies history.	Test that the result of applying the noisegen2D method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'noisegen2D'. 2) Check that the re-built object is the same object as the input.	pass
07	The noisegen2D method can not modify the input AO.	The noisegen2D method can not modify the input AO.	pass
		1) Nothing to do.	pass

ao/noisegen2D			
08	Test the shape of the output.	Test that the noisegen2D method keeps the data shape of the input object. The input AO must be a couple AO with row data and a couple AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the noisegen2D method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the noisegen2D method properly applies history.	Test that the result of applying the noisegen2D method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'noisegen2D'. 2) Check that the re-built object is the same object as the input.	pass

Table 93: Unit tests for ao/noisegen2D.

ao/norm			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the norm method works with a vector of AOs as input.	Test that the norm method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the norm method works with a matrix of AOs as input.	Test that the norm method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the norm method works with a list of AOs as input.	Test that the norm method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the norm method works with a mix of different shaped AOs as input.	Test that the norm method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the norm method properly applies history.	Test that the result of applying the norm method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'norm'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the norm method can modify the input AO.	Test that the norm method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/norm			
		1) Check that 'at4' and 'ain' are now different. 2) Check that 'ain' is norm(at4).	pass
08	Control the method with a plist.	Test that the norm method can modify the single axis controlled by the plist and the resuld can be processed back to an m-file.	pass
		1) Check that the norm method applies with different options 2) Check that the re-built objects are the same object as 'out[1..4]'.	pass
09	Control the method with a plist.	Test that the norm method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass

Table 94: Unit tests for ao/norm.

ao/offset			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the offset method works with a vector of AOs as input.	Test that the offset method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the offset method works with a matrix of AOs as input.	Test that the offset method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the offset method works with a list of AOs as input.	Test that the offset method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the offset method works with a mix of different shaped AOs as input.	Test that the offset method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the offset method properly applies history.	Test that the result of applying the offset method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'offset'. 2) Check that the rebuilt object is the same object as the input.	pass
07	Tests that the offset method can modify the input AO.	Test that the offset method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/offset			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified have an offset of 3 4) Check that out and amodi are the same	pass
08	Test that the offset method uses the offset in a plist.	Test that the offset method uses the offset in a plist.	pass
		1) Check that the offset uses the offset in the plist 2) Check that the re-built object is the same as 'out'	pass
09	Test the shape of the output.	Test that the offset method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the offset method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/offset] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 95: Unit tests for ao/offset.

ao/or			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
rule1 (tsdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass

ao/or			
rule1 (cdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single tsdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single fsdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector cdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector cdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single tsdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single fsdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP cdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ tsdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ fsdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP cdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/or			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
or			pass
			pass
tests (fsdata and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (AO no data and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and AO no data)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different fs in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x values in fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

ao/or			

Table 96: Unit tests for ao/or.

ao/phase			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/phase] method works with a vector of objects as input.	Test that the [ao/phase] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/phase] method works with a matrix of objects as input.	Test that the [ao/phase] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/phase] method works with a list of objects as input.	Test that the [ao/phase] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/phase] method works with a mix of different arrays of objects as input.	Tests that the [ao/phase] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/phase] method properly applies history.	Test that the result of applying the [ao/phase] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/phase]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/phase] method can modify the input AO.	Test that the [ao/phase] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/phase			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/phase] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/phase] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/phase] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/phase] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 97: Unit tests for ao/phase.

ao/plus			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
rule1 (tsdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass

ao/plus			
rule1 (cdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single tsdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single fsdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector cdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector cdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single tsdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single fsdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP cdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ tsdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ fsdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP cdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/plus			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
plus			pass
			pass
tests (fsdata and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (AO no data and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and AO no data)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different fs in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x units in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x values in fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

ao/plus			
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

Table 98: Unit tests for ao/plus.

ao/polyfit			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the polyfit method fails with with a vector of AOs as input.	Test that the polyfit method works with a vector of AOs as input.	pass
		1) Check we have the correct number of output objects in the version with single output 2) Check we have the correct number of elements in the output objects	pass
03	Tests that the polyfit method works with a matrix of AOs as input.	Test that the polyfit method works for a matrix of AOs as input.	pass
		1) Check we have the correct number of output objects	pass
04	Tests that the polyfit method works with a list of AOs as input.	Test that the polyfit method works for a list of AOs as input.	pass
		1) Check we have the correct number of output objects in the version with single output 2) Check we have the correct number of elements in the output objects	pass
05	Tests that the polyfit method works with a mix of different shaped AOs as input.	Test that the polyfit method works with an input of matrices and vectors and single AOs.	pass
		1) Check we have the correct number of output objects	pass
06	Tests that the polyfit method properly applies history.	Test that the result of applying the polyfit method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'polyfit'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the polyfit method cannot modify the input AO.	Test that the polyfit method cannot modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Nothing to check	pass

ao/polyfit			
08	Tests that the polyfit method handles units correctly.	Tests that the polyfit method handles units correctly.	pass
		1) Check that the units in both cases are yunits/(xunits)$\hat{}$(j)	pass
09	Tests that the ao/polyfit method agrees with MATLAB's polyfit when configured to use the same parameters.	Test that applying polyfit works on a single AO.	pass
		1) Check that output agrees with the output of MATLAB's polyfit.	pass

Table 99: Unit tests for ao/polyfit.

ao/power			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the power method works with a vector of AOs as input.	Test that the power method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1 2) Check that each output AO contains the correct data.	pass
03	Tests that the power method works with a matrix of AOs as input.	Test that the power method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1 2) Check that each output AO contains the correct data.	pass
04	Tests that the power method works with a list of AOs as input.	Test that the power method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that each output AO contains the correct data. 3) Check the y-units	pass
05	Tests that the power method works with a mix of different shaped AOs as input.	Test that the power method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is 1. 2) Check that each output AO contains the correct data.	pass
06	Tests that the power method properly applies history.	Test that the result of applying the power method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'power'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the power method can modify the input AO.	Test that the power method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/power			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is power(at1).	pass
08	Test the shape of the output.	Test that the power method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Test the method with all data objects.	Test that the power method works with cdata-, fsdata-, tsdata-, and xydata objects	pass
		1) Check that the shpe of the data doesn't change. 2) Check that re-building of output is the same as the output	pass

Table 100: Unit tests for ao/power.

ao/psd			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the psd method works with a vector of AOs as input.	Test that the psd method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output object contains the correct values.	pass
03	Tests that the psd method works with a matrix of AOs as input.	Test that the psd method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output object contains the correct values.	pass
04	Tests that the psd method works with a list of AOs as input.	Test that the psd method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the psd method works with a mix of different shaped AOs as input.	Test that the psd method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the psd method properly applies history.	Test that the result of applying the psd method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'psd'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the psd method agrees with MATLAB's pwelch, within some tolerance, when configured to use the same parameters.	Test that the applying psd works on a single AO.	pass
		1) Check that output agrees with the output of MATLAB's pwelch within tolerance.	pass

ao/psd			
08	Check that the psd method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Tests that the psd method agrees with MATLAB's pwelch when configured to use the same parameters.	Test that the applying psd works on a single AO.	pass
		1) Check that output agrees with the output of MATLAB's pwelch.	pass
10	Tests that the psd method agrees with MATLAB's pwelch when configured to use the same parameters.	Test that the applying psd works on a single AO.	pass
		1) Check that output agrees with the output of MATLAB's pwelch.	pass
11	Check that the [ao/psd] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests "conservation of energy": 1) white noise produced from uniform pdf, with a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) psd of the white noise 4) compares the sqrt(sum(S * df)) with the standard deviation s	1) Prepare the test tsdata: white noise from uniform distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd	pass
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
13	Tests "conservation of energy" with fixed parameters: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) psd of the white noise 4) compares the sqrt(sum(S * df)) with the standard deviation s	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd	pass
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
14	Tests "conservation of energy" with fixed parameters: 1) white noise produced from uniform pdf, with a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) psd of the white noise 4) compares the sqrt(sum(S * df)) with the standard deviation s	1) Prepare the test tsdata: white noise from uniform distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd	pass

ao/psd			
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
15	Tests "conservation of energy" with fixed parameters: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) psd of the white noise 4) compares the	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd	pass
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
16	Tests "conservation of energy" with fixed parameters: 1) white noise produced from uniform pdf, with a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) psd of the white noise 4) compares the	1) Prepare the test tsdata: white noise from uniform distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd	pass
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
17	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) PSD of the white noise 3) compares the units of the input and output	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) PSD of the white noise	pass
		1) Check that (calculated PSD yunits) equals (input units)$\hat{}$2 / Hz 2) Check that (calculated PSD xunits) equals Hz	pass
18	Tests handling of units: 1) white noise produced from uniform pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) ASD of the white noise 3) compares the units of the input and output	1) Prepare the test tsdata: white noise from uniform distribution + offset 2) Assign a random unit 3) ASD of the white noise	pass
		1) Check that (calculated ASD yunits) equals (input units) / Hz$\hat{}$(1/2) 2) Check that (calculated ASD xunits) equals Hz	pass
19	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) PS of the white noise 3) compares the units of the input and output	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) PS of the white noise	pass
		1) Check that (calculated PS yunits) equals (input units)$\hat{}$2 2) Check that (calculated PS xunits) equals Hz	pass
20	Tests handling of units: 1) white noise produced from uniform distribution, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) AS of the white noise 3) compares the units of the input and output	1) Prepare the test tsdata: white noise from uniform distribution + offset 2) Assign a random unit 3) AS of the white noise	pass

ao/psd			
		1) Check that (calculated AS yunits) equals (input units) 2) Check that (calculated AS xunits) equals Hz	pass
21	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Welch window and no detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
22	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Welch window and mean detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
23	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Welch window and linear detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass

ao/psd			
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
24	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Hanning window and no detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
25	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Hanning window and mean detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the mean performed on the individual segments 8) Checks on averaged PSDs: mean and standard deviation of all the PSD points 9) Compares the grand-mean with the estimated white noise level	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass

ao/psd			
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
26	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Hanning window and linear detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
27	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Hamming window and no detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the mean performed on the individual segments 8) Checks on averaged PSDs: mean and standard deviation of all the PSD points 9) Compares the grand-mean with the estimated white noise level	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass

ao/psd			
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
28	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Hamming window and mean detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
29	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Hamming window and linear detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the mean performed on the individual segments 8) Checks on averaged PSDs: mean and standard deviation of all the PSD points 9) Compares the grand-mean with the estimated white noise level	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass

ao/psd			
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
30	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: BH92 window and no detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
31	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: BH92 window and mean detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the mean performed on the individual segments 8) Checks on averaged PSDs: mean and standard deviation of all the PSD points 9) Compares the grand-mean with the estimated white noise level	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass

ao/psd			
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
32	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: BH92 window and linear detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the mean performed on the individual segments 8) Checks on averaged PSDs: mean and standard deviation of all the PSD points 9) Compares the grand-mean with the estimated white noise level	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
33	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Kaiser200 window and no detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the mean performed on the individual segments 8) Checks on averaged PSDs: mean and standard deviation of all the PSD points 9) Compares the grand-mean with the estimated white noise level	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass

ao/psd			
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
34	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Kaiser200 window and mean detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
35	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) Calculate the expected level of noise from sample statistics 3) Calculates the PSD of the individual parts, with: Kaiser200 window and linear detrend 4) Evaluate the mean and standard deviation of the Checks on individual PSDs: 5) Checks on individual PSDs: mean and standard deviation of the PSD points 6) Evaluate the expected value, estimated from the std of the individual segments 7) Compares with the mean performed on the individual segments 8) Checks on averaged PSDs: mean and standard deviation of all the PSD points 9) Compares the grand-mean with the estimated white noise level	1) Split the reference data into different segments 2) Calculates the PSD of the individual parts	pass

ao/psd			
		1) Evaluate the mean and standard deviation of the Checks on individual PSDs: 2) Checks on individual PSDs: mean and standard deviation of the PSD points 3) Evaluate the expected value, estimated from the std of the individual segments 4) Compares with the mean performed on the individual segments 5) Checks on averaged PSDs: mean and standard deviation of all the PSD points 6) Compares the grand-mean with the estimated white noise level	pass
38	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) add a given trend of order n 4) psd of the noise, with proper	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Add a trend 4) Estimate the psd	pass
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
39	Tests "conservation of energy": 1) white noise produced from uniform pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) add a given trend of order n 4) psd of the noise, with proper	1) Prepare the test tsdata: white noise from uniform distribution + offset 2) Calculate the statistical parameters 3) Add a trend 4) Estimate the psd	pass
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
40	Tests "conservation of energy": 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) add a given trend of order n 4) psd of the noise, with proper	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Add a trend 4) Estimate the psd	pass
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
41	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, Rectangular window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5) compares the to psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Rectangular window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass

ao/psd			
		1) Check that calculated psds are identical within a part in 10Î12	pass
42	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, BH92 window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5) compares the to psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, BH92 window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10Î12	pass
43	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, Hamming window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5) compares the to psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Hamming window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10Î12	pass
44	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, Hanning window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5) compares the to psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Hanning window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10Î12	pass
45	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, Bartlett window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5) compares the to psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Bartlett window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10Î12	pass

ao/psd			
46	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, Nuttall3 window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5) compares the to psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Nuttall3 window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10Î12	pass
47	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, Kaiser psll = random window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5)	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Kaiser psll = random window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10Î12	pass
48	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, Kaiser psll = default window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5)	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Kaiser psll = default window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10Î12	pass
49	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, Nuttall4 window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5) compares the to psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, Nuttall4 window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10Î12	pass

ao/psd			
50	Tests the effect of windowing: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, SFT3F window 3) Apply the detrending and the window manually 4) psd of the noise, without detrending, Rectangular window 5) compares the to psds	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd without detrending, SFT3F window 4) Manually apply window to the data 5) Estimate the psd without detrending, Rectangular window	pass
		1) Check that calculated psds are identical within a part in 10$\hat{}$12	pass
51	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise,	1) Prepare the test tsdata: white noise from normal distribution + offset 2) psd of the noise, without detrending, random window, set number of averages	pass
		1) Check that calculated navs are identical to those requested	pass
52	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run psd again, with the nfft used	1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run psd again, with the nfft used	pass
		1) Check that calculated objects S1 and S2 are identical	pass
53	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run psd again, with the nfft used 6) compare navs, nfft, psds	1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run psd again, with the nfft used 6) run psd again, with conflicting parameters, and verify it uses nfft rather than navs	pass
		1) Check that calculated objects S1 and S2 are identical 2) Check that S3 used different values	pass
54	Tests "conservation of energy": 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd order) 2) evaluate the sample mean value m and standard deviation s 3) psd of the white noise 4) compares the sqrt(sum(S * df)) with the standard deviation s	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd	pass

ao/psd			
		1) Check that calculated psd energy equals the rms content of the tsdata, estimated by the std of the sample	pass
60	Tests the 'basic' call: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, defualt detrending, default window, no averaging	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd with default detrending, default window, no averaging	pass
		1) Nothing to check	pass
61	Tests the 'basic' call: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, defualt detrending, default window, no averaging	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd with default detrending, default window, no averaging	pass
		1) Nothing to check	pass
62	Tests the 'basic' call: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, defualt detrending, default window, no averaging	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd with default detrending, default window, no averaging	pass
		1) Nothing to check	pass
63	Tests the 'basic' call: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) psd of the noise, defualt detrending, default window, no averaging	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Calculate the statistical parameters 3) Estimate the psd with default detrending, default window, no averaging	pass
		1) Nothing to check	pass

Table 101: Unit tests for ao/psd.

ao/rdivide			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
rule1 (tsdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass

ao/rdivide			
rule1 (cdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single tsdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single fsdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector cdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector cdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single tsdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single fsdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP cdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ tsdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ fsdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP cdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/rdivide			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rdivide			pass
			pass
tests (fsdata and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (AO no data and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and AO no data)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different fs in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x values in fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

ao/rdivide			

Table 102: Unit tests for ao/rdivide.

ao/real			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the real method works with a vector of AOs as input.	Test that the real method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the real method works with a matrix of AOs as input.	Test that the real method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the real method works with a list of AOs as input.	Test that the real method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the real method works with a mix of different shaped AOs as input.	Test that the real method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the real method properly applies history.	Test that the result of applying the real method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'real'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the real method can modify the input AO.	Test that the real method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is real(at1).	pass

ao/real			
08	Control the method with a plist.	Test that the real method can modify the single axis controlled by the plist and the resuld can be processed back to an m-file.	pass
		1) Check that the real method applies to the x-axis 2) Check that the real method applies to the y-axis 3) Check that the real method applies to both axes 4) Check that the re-built objects are the same object as 'out[1..3]'.	pass
09	Control the method with a plist.	Test that the real method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the real method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/real] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 103: Unit tests for ao/real.

ao/rebuild			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rebuild method works with a vector of AO objects as input.	Test that the rebuild method works for a vector of AO objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the rebuild method works with a matrix of AO objects as input.	Test that the rebuild method works for a matrix of AO objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the rebuild method works with a list of AO objects as input.	Test that the rebuild method works for a list of AO objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the rebuild method works with a mix of different shaped AO objects as input.	Test that the rebuild method works with an input of matrices and vectors and single AO objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the rebuild method properly applies history.	The method rebuild doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Check that the rebuild method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/rebuild] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 104: Unit tests for ao/rebuild.

ao/resample			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the resample method works with a vector of AOs as input.	Test that the resample method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the resample method works with a matrix of AOs as input.	Test that the resample method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the resample method works with a list of AOs as input.	Test that the resample method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the resample method works with a mix of different shaped AOs as input.	Test that the resample method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the resample method properly applies history.	Test that the result of applying the resample method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'resample'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/resample			
07	Tests that the resample method can modify the input AO.	Test that the resample method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is resample(at1).	pass
08	Control the method with a plist.	Test that the resample method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the resample method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/resample] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 105: Unit tests for ao/resample.

ao/rms			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rms method works with a vector of AOs as input.	Test that the rms method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the rms method works with a matrix of AOs as input.	Test that the rms method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the rms method works with a list of AOs as input.	Test that the rms method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the rms method works with a mix of different shaped AOs as input.	Test that the rms method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the rms method properly applies history.	Test that the result of applying the rms method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rms'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the rms method can modify the input AO.	Test that the rms method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/rms			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is rms(at1). 3) Check the algorithm	pass
08	Control the method with a plist.	Test that the rms method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the rms method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/rms] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 106: Unit tests for ao/rms.

ao/round			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/round] method works with a vector of objects as input.	Test that the [ao/round] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/round] method works with a matrix of objects as input.	Test that the [ao/round] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/round] method works with a list of objects as input.	Test that the [ao/round] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/round] method works with a mix of different arrays of objects as input.	Tests that the [ao/round] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/round] method properly applies history.	Test that the result of applying the [ao/round] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/round]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/round] method can modify the input AO.	Test that the [ao/round] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/round			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/round] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/round] method uses the plist to get the axis.	Test that the [ao/round] method uses the plist to get the axis.	pass
		1) Check that the [ao/round] method applies to the x-axis 2) Check that the [ao/round] method applies to the y-axis 3) Check that the [ao/round] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/round] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/round] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/round] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 107: Unit tests for ao/round.

ao/sDomainFit			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the sDomainFit method works with a vector of AOs as input.	Test that the sDomainFit method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'av' 2) Check that each output AO contains the correct data.	pass
03	Tests that the sDomainFit method works with a matrix of AOs as input.	Test that the sDomainFit method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'am' 2) Check that each output AO contains the correct data.	pass
04	Tests that the sDomainFit method works with a list of AOs as input.	Test that the sDomainFit method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the sDomainFit method works with a mix of different shaped AOs as input.	Test that the sDomainFit method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the sDomainFit method properly applies history.	Test that the result of applying the sDomainFit method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'sDomainFit'. 2) Check that the re-built object is the same object as the input.	pass
07	sDomainFit cannot modify the input AO.	Test that the sDomainFit method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/sDomainFit			
		1) Nothing to do.	pass
09	Check that the sDomainFit method pass back the output objects to a single variable correctly.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the sDomainFit method return the correct coefficients	Test that the sDomainFit method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that output contains the correct coefficients.	pass

Table 108: Unit tests for ao/sDomainFit.

ao/save			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the save method works with a vector of AOs as input.	Test that the save method works for a vector of AOs as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'atvec' 2) Check that the loaded objects are the same as the saved objects except the the history because the load-constructor adds one history step. 3) The outputs 'out1' and 'out2' must be the same except the history properties 'methodInfo', 'plistUsed' and 'proctime'.	pass
03	Tests that the save method works with a matrix of AOs as input.	Test that the save method works for a matrix of AOs as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'atmat' 2) Check that the loaded objects are the same as the saved objects except the the history because the load-constructor adds one history step. 3) The outputs 'out1' and 'out2' must be the same except the history properties 'methodInfo', 'plistUsed' and 'proctime'.	pass
04	Tests that the save method works with a list of AOs as input.	Test that the save method works for a list of AOs as input. Test both formats 'xml' and 'mat'.	pass

ao/save			
		1) Check that the number of elements in 'out1' and 'out2' are the same as in the list 2) Check that the loaded objects are the same as the saved objects except the the history because the load-constructor adds one history step. 3) The outputs 'out1' and 'out2' must be the same except the history properties 'methodInfo', 'plistUsed' and 'proctime'.	pass
05	Tests that the save method works with a mix of different shaped AOs as input.	Test that the save method works with an input of matrices and vectors and single AOs. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the save method properly applies history.	Test that the result of applying the save method can be processed back to an m-file. Do this for both extensions 'mat' and 'xml'	pass
		1) Check that save doesn't add a history step. 2) Check that the read object is the same as the stored object. save + load doesn't add history. 3) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the save method works with the modify command.	Test that the modify command.	pass
		1) Check that the save method applies the history. 2) Check the output against the input except the history. 3) Check the history of the output against the input.	pass
08	Control the method with a plist.	Test that the save method uses the filename which is stored in a plist.	pass
		1) Check the output	pass
09	Test the save method with an AO with complex data.	Save an AO with complex fsdata object.	pass
		1) Check the output	pass
10	Test the save method with an AO which is created from a pole/zero model	Save an AO which is created from a pzmodel.	pass
		1) Check the output	pass
11	Test the save method with a complex plist which builds the AO	Save an AO which is created from a complex plist.	pass

ao/save			
		1) Check the output	pass

Table 109: Unit tests for ao/save.

ao/scale			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/scale] method works with a vector of objects as input.	Test that the [ao/scale] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/scale] method works with a matrix of objects as input.	Test that the [ao/scale] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/scale] method works with a list of objects as input.	Test that the [ao/scale] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/scale] method works with a mix of different arrays of objects as input.	Tests that the [ao/scale] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/scale] method properly applies history.	Test that the result of applying the [ao/scale] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/scale]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/scale] method can modify the input AO.	Test that the [ao/scale] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/scale			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/scale] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/scale] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/scale] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/scale] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 110: Unit tests for ao/scale.

ao/search			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the search method works with a vector of AOs as input.	Test that the search method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out1' are the same as in 'atvec' 2) Check that the number of elements in 'out2' 3) Check that each output AO contains the correct data.	pass
03	Tests that the search method works with a matrix of AOs as input.	Test that the search method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out1' are the same as in 'atmat' 2) Check that the number of elements in 'out2' 3) Check that each output AO contains the correct data.	pass
04	Tests that the search method works with a list of AOs as input.	Test that the search method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out1' are the same as the input 2) Check that the number of elements in 'out2' 3) Check that each output AO contains the correct data.	pass
05	Tests that the search method works with a mix of different shaped AOs as input.	Test that the search method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out1' are the same as the input 2) Check that the number of elements in 'out2' 3) Check that 'out3' doesn't contain an AO 4) Check that each output AO contains the correct data.	pass
06	Tests that the search method properly applies history.	Test that the result of applying the search method can be processed back.	pass

ao/search			
		1) Check that the last entry in the history of 'out' corresponds to 'search'. 2) Check that the rebuilt object is the same object as the input.	pass
07	Tests that the modifier call for the search method doesn't work.	Tests that the modifier call for the search method doesn't work.	pass
			pass
08	Check that the search method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 111: Unit tests for ao/search.

ao/select			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the select method works with a vector of AOs as input.	Test that the select method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the select method works with a matrix of AOs as input.	Test that the select method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the select method works with a list of AOs as input.	Test that the select method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the select method works with a mix of different shaped AOs as input.	Test that the select method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the select method properly applies history.	Test that the result of applying the select method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'select'. 2) Check that the rebuilt object is the same object as the input.	pass
07	Tests that the select method can modify the input AO.	Test that the select method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/select			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the selected value of the copy 4) Check that out and amodi are the same	pass
08	Test the shape of the output.	Test that the select method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the select method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Test that the select method uses the plist to select the samples.	Test that the select method uses the plist to select the samples.	pass
		1) Check that the select method uses the samples in the plist 2) Check that the select method uses the samples in the plist and input 3) Check that the select method uses the samples in the plist and input 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
11	Check that the [ao/select] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Test that the select method also select the 'dx' and 'dy' values.	Test that the select method also select the 'dx' and 'dy' values.	pass
		1) Check the output 2) Check that the re-built object is the same as in 'out'.	pass
13	Test that the select method updates the t0.	Test that the select method updates the t0.	pass

ao/select			
		1) Check the output 2) Check that the re-built object is the same as in 'out'.	pass

Table 112: Unit tests for ao/select.

ao/setDescription			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setDescription method works with a vector of AOs as input.	Test that the setDescription method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setDescription method works with a matrix of AOs as input.	Test that the setDescription method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setDescription method works with a list of AOs as input.	Test that the setDescription method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setDescription method works with a mix of different shaped AOs as input.	Test that the setDescription method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setDescription method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setDescription method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setDescription'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setDescription'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/setDescription			
07	Tests that the setDescription method can modify the input AO.	Test that the setDescription method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct description field	pass
08	Tests that the setDescription method can set the property with a plist.	Test that the setDescription method can modify the property 'description' with a value in a plist.	pass
		1) Check that 'ain' has the correct description field 2) Check that the re-built object is the same object as 'out'.	pass
09	Check that the setDescription method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setDescription] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 113: Unit tests for ao/setDescription.

ao/setFs			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setFs method works with a vector of AOs as input.	Test that the setFs method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'avec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setFs method works with a matrix of AOs as input.	Test that the setFs method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setFs method works with a list of AOs as input.	Test that the setFs method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setFs method works with a mix of different shaped AOs as input.	Test that the setFs method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setFs method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setFs method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setFs'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setFs'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the setFs method can modify the input AO.	Test that the setFs method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct fs field	pass

ao/setFs			
08	Tests that the setFs method can set the property with a plist.	Test that the setFs method can modify the property 'fs' with a value in a plist.	pass
		1) Check that 'ain' has the correct fs field 2) Check that the rebuilt object is the same object as 'out'.	pass
09	Check that the setFs method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setFs] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 114: Unit tests for ao/setFs.

ao/setName			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setName method works with a vector of AOs as input.	Test that the setName method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output contains the correct data.	pass
03	Tests that the setName method works with a matrix of AOs as input.	Test that the setName method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output contains the correct data.	pass
04	Tests that the setName method works with a list of AOs as input.	Test that the setName method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setName method works with a mix of different shaped AOs as input.	Test that the setName method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setName method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setName method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setName'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setName'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the setName method can modify the input AO.	Test that the setName method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass

ao/setName			
08	Tests that the setName method can set the property with a plist.	Test that the setName method can modify the property 'name' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the re-built object is the same object as 'out'.	pass
09	Check that the setName method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setName] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 115: Unit tests for ao/setName.

ao/setPlotinfo			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setPlotinfo method works with a vector of AOs as input.	Test that the setPlotinfo method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setPlotinfo method works with a matrix of AOs as input.	Test that the setPlotinfo method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setPlotinfo method works with a list of AOs as input.	Test that the setPlotinfo method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setPlotinfo method works with a mix of different shaped AOs as input.	Test that the setPlotinfo method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setPlotinfo method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setPlotinfo method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setPlotinfo'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setPlotinfo'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the setPlotinfo method can modify the input AO.	Test that the setPlotinfo method can modify the input AO by calling with no output.	pass

ao/setPlotinfo			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct plotinfo field	pass
08	Tests that the setPlotinfo method can set the property with a plist.	Test that the setPlotinfo method can modify the property 'plotinfo' with a value in a plist.	pass
		1) Check that 'ain' has the correct plotinfo field 2) Check that the re-built object is the same object as 'out'.	pass
09	Check that the setPlotinfo method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 116: Unit tests for ao/setPlotinfo.

ao/setT0			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setT0 method works with a vector of AOs as input.	Test that the setT0 method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'avec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setT0 method works with a matrix of AOs as input.	Test that the setT0 method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setT0 method works with a list of AOs as input.	Test that the setT0 method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setT0 method works with a mix of different shaped AOs as input.	Test that the setT0 method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setT0 method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setT0 method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setT0'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setT0'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the setT0 method can modify the input AO.	Test that the setT0 method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct t0 field	pass

ao/setT0			
08	Tests that the setT0 method can set the property with a plist.	Test that the setT0 method can modify the property 't0' with a value in a plist.	pass
		1) Check that 'ain' has the correct t0 field 2) Check that the rebuilt object is the same object as 'out'.	pass
09	Check that the setT0 method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setT0] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 117: Unit tests for ao/setT0.

ao/setX			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setX method works with a vector of AOs as input.	Test that the setX method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'avec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setX method works with a matrix of AOs as input.	Test that the setX method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setX method works with a list of AOs as input.	Test that the setX method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setX method works with a mix of different shaped AOs as input.	Test that the setX method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setX method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setX method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setX'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setX'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the setX method can modify the input AO.	Test that the setX method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct 'x' field	pass

ao/setX			
08	Tests that the setX method can set the property with a plist.	Test that the setX method can modify the property 'x' with a value in a plist.	pass
		1) Check that 'ain' has the correct x field 2) Check that the rebuilt object is the same object as 'out'.	pass
09	Tests that the setX method keeps the shape of the data.	Test that the setX method keeps the shape of the data. Independent of the input	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the setX method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setX] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 118: Unit tests for ao/setX.

ao/setXY			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setXY method works with a vector of AOs as input.	Test that the setXY method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'avec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setXY method works with a matrix of AOs as input.	Test that the setXY method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setXY method works with a list of AOs as input.	Test that the setXY method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setXY method works with a mix of different shaped AOs as input.	Test that the setXY method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setXY method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setXY method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setXY'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setXY'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the setXY method can modify the input AO.	Test that the setXY method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct 'x' and 'y' field	pass

ao/setXY			
08	Tests that the setXY method can set the property with a plist.	Test that the setXY method can modify the property 'x' and 'y' with a value in a plist.	pass
		1) Check that 'ain' has the correct x and y fields 2) Check that the re-built object is the same object as 'out'.	pass
09	Tests that the setXY method keeps the shape of the data.	Test that the setXY method keeps the shape of the data. Independent of the input	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the setXY method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setXY] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 119: Unit tests for ao/setXY.

ao/setXunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setXunits method works with a vector of AOs as input.	Test that the setXunits method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'avec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setXunits method works with a matrix of AOs as input.	Test that the setXunits method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setXunits method works with a list of AOs as input.	Test that the setXunits method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setXunits method works with a mix of different shaped AOs as input.	Test that the setXunits method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setXunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setXunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setXunits'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setXunits'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the setXunits method can modify the input AO.	Test that the setXunits method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct xunits field	pass

ao/setXunits			
08	Tests that the setXunits method can set the property with a plist.	Test that the setXunits method can modify the property 'xunits' with a value in a plist.	pass
		1) Check that 'ain' has the correct xunits field 2) Check that the re-built object is the same object as 'out'.	pass
09	Check that the setXunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setXunits] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 120: Unit tests for ao/setXunits.

ao/setY			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setY method works with a vector of AOs as input.	Test that the setY method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setY method works with a matrix of AOs as input.	Test that the setY method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setY method works with a list of AOs as input.	Test that the setY method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setY method works with a mix of different shaped AOs as input.	Test that the setY method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setY method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setY method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setY'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setY'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the setY method can modify the input AO.	Test that the setY method can modify the input AO by calling with no output.	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct 'y' field	pass

ao/setY			
08	Tests that the setY method can set the property with a plist.	Test that the setY method can modify the property 'y' with a value in a plist.	pass
		1) Check that 'ain' has the correct y field 2) Check that the rebuilt object is the same object as 'out'.	pass
09	Tests that the setY method keeps the shape of the data.	Test that the setY method keeps the shape of the data. Independent of the input	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the setY method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setY] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 121: Unit tests for ao/setY.

ao/setYunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setYunits method works with a vector of AOs as input.	Test that the setYunits method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the setYunits method works with a matrix of AOs as input.	Test that the setYunits method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the setYunits method works with a list of AOs as input.	Test that the setYunits method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the setYunits method works with a mix of different shaped AOs as input.	Test that the setYunits method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the setYunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setYunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setYunits'. 1) Check that the last entry in the history of 'out2' NOT corresponds to 'setYunits'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the setYunits method can modify the input AO.	Test that the setYunits method can modify the input AO by calling with no output.	pass

ao/setYunits			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' has the correct yunits field	pass
08	Tests that the setYunits method can set the property with a plist.	Test that the setYunits method can modify the property 'yunits' with a value in a plist.	pass
		1) Check that 'ain' has the correct yunits field 2) Check that the re-built object is the same object as 'out'.	pass
09	Check that the setYunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/setYunits] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 122: Unit tests for ao/setYunits.

ao/sign			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/sign] method works with a vector of objects as input.	Test that the [ao/sign] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/sign] method works with a matrix of objects as input.	Test that the [ao/sign] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/sign] method works with a list of objects as input.	Test that the [ao/sign] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/sign] method works with a mix of different arrays of objects as input.	Tests that the [ao/sign] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/sign] method properly applies history.	Test that the result of applying the [ao/sign] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/sign]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/sign] method can modify the input AO.	Test that the [ao/sign] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/sign			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/sign] value of the copy 4) Check that out and amodi are the same	pass
09	Test the shape of the data in AOs.	Test that the [ao/sign] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/sign] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/sign] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 123: Unit tests for ao/sign.

ao/simplifyYunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the simplifyYunits method works with a vector of AOs as input.	Test that the simplifyYunits method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the simplifyYunits method works with a matrix of AOs as input.	Test that the simplifyYunits method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the simplifyYunits method works with a list of AOs as input.	Test that the simplifyYunits method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the simplifyYunits method works with a mix of different shaped AOs as input.	Test that the simplifyYunits method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the simplifyYunits method properly applies history.	Test that the result of applying the simplifyYunits method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'simplifyYunits'. 2) Check that the re-built object is the same object as the input.	pass

ao/simplifyYunits			
07	Tests that the simplifyYunits method can modify the input AO.	Test that the simplifyYunits method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input simplifies the y-units of the copy 4) Check that out and amodi are the same	pass
08	Check that the simplifyYunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Test that the simplifyYunits works for different complex y-units.	Test that the simplifyYunits works for different complex y-units.	pass
		1) Check that the correct data 2) Check that the re-built object is the same as in 'out[1..3]'.	pass
10	Test that the simplifyYunits works with a exception list.	Test that the simplifyYunits works with a exception list.	pass
		1) Check that the correct data 2) Check that the re-built object is the same as in 'out[1..2]'.	pass
11	Check that the [ao/simplifyYunits] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Test that the simplifyYunits doesn't apply the prefixes to the data.	Test that the simplifyYunits doesn't apply the prefixes to the data.	pass
		1) Check that the correct data 2) Check that the re-built object is the same as in 'out[1..2]'.	pass

Table 124: Unit tests for ao/simplifyYunits.

ao/sin			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/sin] method works with a vector of objects as input.	Test that the [ao/sin] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/sin] method works with a matrix of objects as input.	Test that the [ao/sin] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/sin] method works with a list of objects as input.	Test that the [ao/sin] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/sin] method works with a mix of different arrays of objects as input.	Tests that the [ao/sin] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/sin] method properly applies history.	Test that the result of applying the [ao/sin] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/sin]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/sin] method can modify the input AO.	Test that the [ao/sin] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/sin			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/sin] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/sin] method uses the plist to get the axis.	Test that the [ao/sin] method uses the plist to get the axis.	pass
		1) Check that the [ao/sin] method applies to the x-axis 2) Check that the [ao/sin] method applies to the y-axis 3) Check that the [ao/sin] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/sin] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/sin] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/sin] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 125: Unit tests for ao/sin.

ao/smoother			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the smoother method works with a vector of AOs as input.	Test that the smoother method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the smoother method works with a matrix of AOs as input.	Test that the smoother method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the smoother method works with a list of AOs as input.	Test that the smoother method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the smoother method works with a mix of different shaped AOs as input.	Test that the smoother method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the smoother method properly applies history.	Test that the result of applying the smoother method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'smoother'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the smoother method can modify the input AO.	Test that the smoother method can modify the input AO by calling with no output.	pass

ao/smoother			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is smoother(at5).	pass
08	Tests that the smoother method keeps the data shape of the input object.	Test that the smoother method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the smoother method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/smoother] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 126: Unit tests for ao/smoother.

ao/sort			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the sort method works with a vector of AOs as input.	Test that the sort method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the sort method works with a matrix of AOs as input.	Test that the sort method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the sort method works with a list of AOs as input.	Test that the sort method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the sort method works with a mix of different shaped AOs as input.	Test that the sort method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the sort method properly applies history.	Test that the result of applying the sort method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'sort'. 2) Check that the rebuilt object is the same object as the input.	pass
07	Tests that the sort method can modify the input AO.	Test that the sort method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/sort			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the sort value of the copy 4) Check that out and amodi are the same	pass
08	Test the shape of the output.	Test that the sort method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the sort method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Test that the sort method uses the plist to sort on the x- or y-axis	Test that the sort method uses the plist to get the axis.	pass
		1) Check that the sort method applies to the y-axis in ascend direction 2) Check that the sort method applies to the y-axis in descend direction 3) Check that the sort method applies to the x-axis in ascend direction 4) Check that the sort method applies to the x-axis in descend direction 5) Check that the re-built object is the same as in 'out[1..4]'.	pass
11	Check that the [ao/sort] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 127: Unit tests for ao/sort.

ao/spectrogram			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the spectrogram method works with a vector of AOs as input.	Test that the spectrogram method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the spectrogram method works with a matrix of AOs as input.	Test that the spectrogram method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the spectrogram method works with a list of AOs as input.	Test that the spectrogram method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the spectrogram method works with a mix of different shaped AOs as input.	Test that the spectrogram method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the spectrogram method properly applies history.	Test that the result of applying the spectrogram method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'spectrogram'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/spectrogram			
11	Check that the [ao/spectrogram] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 128: Unit tests for ao/spectrogram.

split/chunks			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the split method works with a vector of AOs as input.	Test that the split method works for a vector of AOs as input.	pass
		1) Check that the number of outputs 2) Check that each output AO contains the correct data.	pass
03	Tests that the split method works with a matrix of AOs as input.	Tests that the split method works with a matrix of AOs as input.	pass
		1) Check that the number of outputs 2) Check that each output AO contains the correct data.	pass
04	Tests that the split method works with a list of AOs as input.	Tests that the split method works with a matrix of AOs as input.	pass
		1) Check that the number of outputs 2) Check that each output AO contains the correct data.	pass
05	Tests that the split method works with a mix of different shaped AOs as input.	Tests that the split method works with a matrix of AOs as input.	pass
		1) Check that the number of outputs 2) Check that each output AO contains the correct data.	pass
06	Tests that the split method properly applies history.	Test that the result of applying the split method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'split'. 2) Check that the rebuilt object is the same object as the input.	pass
07	The split method can not modify the input AO.	The split method can not modify the input AO.	pass
		1) Nothind to do.	pass
08	Test the shape of the output.	Test that the split method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass

split/chunks			
09	Check that the split method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the split method accepts both key-words 'N' and 'chunks'.	Check that the split method accepts both key-words 'N' and 'chunks'.	pass
		1) Check that the number of outputs 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [split/chunks] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests that the split method works also split 'dx' and 'dy'.	Tests that the split method works also split 'dx' and 'dy'.	pass
		1) Check that the number of outputs 2) Check that each output AO contains the correct data.	pass
13	Check that the split method ignores the samples which doesn't fit.	Check that the split method ignores the samples which doesn't fit. Use for this test the key-word 'match'	pass
		1) Check that the number of outputs 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option.	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option.	pass
		1) Check the outputs 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 129: Unit tests for split/chunks.

split/interval			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the split method works with a vector of AOs as input.	Test that the split method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'avec' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
03	Tests that the split method works with a matrix of AOs as input.	Tests that the split method works with a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
04	Tests that the split method works with a list of AOs as input.	Tests that the split method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
05	Tests that the split method works with a mix of different shaped AOs as input.	Tests that the split method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
06	Tests that the split method properly applies history.	Test that the result of applying the split method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'split'. 2) Check that the rebuilt object is the same object as the input.	pass
07	The split method can not modify the input AO.	The split method can not modify the input AO.	pass
		1) Nothind to do.	pass

split/interval			
08	Test the shape of the output.	Test that the split method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the split method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the split method can handle AO which have a collaped x-axis.	Check that the split method can handle AO which have a collaped x-axis.	pass
		1) Check the number of outputs 2) Check the output 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [split/interval] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the split method uses the key-words: start_time and duration	Check that the split method uses the key-words: start_time and duration	pass
		1) Check the number of outputs 2) Check the output 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Check that the split method uses the key-words: timespan	Check that the split method uses the key-words: timespan	pass
		1) Check the number of outputs 2) Check the output 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Check that the split method uses the key-words: start_time and end_time	Check that the split method uses the key-words: start_time and end_time	pass

split/interval			
		1) Check the number of outputs 2) Check the output 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option. Check that the interval can be passed via two time objects.	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option. Check that the interval can be passed via two time objects.	pass
		1) Check the outputs 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
16	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option. Check that the interval can be passed via two time strings.	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option. Check that the interval can be passed via two time strings.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
17	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option. Check that the interval can be passed via one timespan object.	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option. Check that the interval can be passed via one timespan object.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
18	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option. Check that the interval can be passed via one timespan object.	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option. Check that the interval can be passed via one timespan object.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 130: Unit tests for split/interval.

split/samples			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the split method works with a vector of AOs as input.	Test that the split method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
03	Tests that the split method works with a matrix of AOs as input.	Tests that the split method works with a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
04	Tests that the split method works with a list of AOs as input.	Tests that the split method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
05	Tests that the split method works with a mix of different shaped AOs as input.	Tests that the split method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
06	Tests that the split method properly applies history.	Test that the result of applying the split method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'split'. 2) Check that the rebuilt object is the same object as the input.	pass
07	The split method can not modify the input AO.	The split method can not modify the input AO.	pass
		1) Nothind to do.	pass

split/samples			
08	Test the shape of the output.	Test that the split method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the split method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Test that the split method also splits the 'dx' and 'dy' values.	Test that the split method also splits the 'dx' and 'dy' values.	pass
		1) Check the output 2) Check that the re-built object is the same as in 'out'.	pass
11	Check that the [split/samples] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
15	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option.	Check that the split method works with notequal sampled data. Check that the split method works with 'timeshift' option.	pass
		1) Check the outputs 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 131: Unit tests for split/samples.

times/frequ			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the split method works with a vector of AOs as input.	Test that the split method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
03	Tests that the split method works with a matrix of AOs as input.	Tests that the split method works with a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
04	Tests that the split method works with a list of AOs as input.	Tests that the split method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
05	Tests that the split method works with a mix of different shaped AOs as input.	Tests that the split method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' times numbers of intervals 2) Check that each output AO contains the correct data.	pass
06	Tests that the split method properly applies history.	Test that the result of applying the split method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'split'. 2) Check that the rebuilt object is the same object as the input.	pass
07	The split method can not modify the input AO.	The split method can not modify the input AO.	pass
		1) Nothind to do.	pass

times/frequ			
08	Test the shape of the output.	Test that the split method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the split method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the split method interprets a negative end interval as a indicates count from end and a zero for the and interval as the end of the vector.	Check this behavior for time-series data.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [times/frequ] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the split method works with notequal sampled data.	Check that the split method works with notequal sampled data.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Check that the split method also split 'dx' and 'dy'.	Check that the split method also split 'dx' and 'dy' if this values have the same length of the y-values.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Check that the split method interprets a negative end interval as a indicates count from end and a zero for the and interval as the end of the vector.	Check this behavior for frequency-series data.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'. 3) Check that the output have the same 'fs', 'enbw', 'navs', 'xunits' and 'yunits'	pass

times/frequ			
15	Check that the split method works with notequal sampled data. Check that the interval can be passed via an array of double. Check that the split method works with 'timeshift' option.	Check that the split method works with notequal sampled data. Check that the interval can be passed via an array of double. Check that the split method works with 'timeshift' option.	pass
		1) Check the outputs 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
16	Check that the split method works with time intervals which doesn' match the sample frequency.	Check that the split method works with time intervals which doesn' match the sample frequency.	pass
		1) Check the outputs 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 132: Unit tests for times/frequ.

ao/sqrt			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/sqrt] method works with a vector of objects as input.	Test that the [ao/sqrt] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/sqrt] method works with a matrix of objects as input.	Test that the [ao/sqrt] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/sqrt] method works with a list of objects as input.	Test that the [ao/sqrt] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/sqrt] method works with a mix of different arrays of objects as input.	Tests that the [ao/sqrt] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/sqrt] method properly applies history.	Test that the result of applying the [ao/sqrt] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/sqrt]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/sqrt] method can modify the input AO.	Test that the [ao/sqrt] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/sqrt			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/sqrt] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/sqrt] method uses the plist to get the axis.	Test that the [ao/sqrt] method uses the plist to get the axis.	pass
		1) Check that the [ao/sqrt] method applies to the x-axis 2) Check that the [ao/sqrt] method applies to the y-axis 3) Check that the [ao/sqrt] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/sqrt] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/sqrt] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/sqrt] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 133: Unit tests for ao/sqrt.

ao/std			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/std] method works with a vector of objects as input.	Test that the [ao/std] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/std] method works with a matrix of objects as input.	Test that the [ao/std] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/std] method works with a list of objects as input.	Test that the [ao/std] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/std] method works with a mix of different arrays of objects as input.	Tests that the [ao/std] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/std] method properly applies history.	Test that the result of applying the [ao/std] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/std]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/std] method can modify the input AO.	Test that the [ao/std] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/std			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/std] value of the copy 4) Check that out and amodi are the same	pass
108	Test that the [ao/std] method uses the plist to get the axis. This is intended to test methods like ao/mean and ao/std which return different data types depending on which axis is selected.	Test that the [ao/std] method uses the plist to get the axis.	pass
		1) Check that the [ao/std] method applies to the x-axis 2) Check that the [ao/std] method applies to the y-axis 3) Check that the [ao/std] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/std] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/std] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/std] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 134: Unit tests for ao/std.

ao/string			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the string method works with a vector of AO objects as input.	Test that the string method works for a vector of AO objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
03	Tests that the string method works with a matrix of AO objects as input.	Test that the string method works for a matrix of AO objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
04	Tests that the string method works with a list of AO objects as input.	Test that the string method works for a list of AO objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
05	Tests that the string method works with a mix of different shaped AO objects as input.	Test that the string method works with an input of matrices and vectors and single AO objects.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
06	Tests that the string method properly applies history.	The method string doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Tests that the string method doesn't work if the AO object have more than one history step.	The method string throws an error because the input object have more than one history step.	pass
			pass

Table 135: Unit tests for ao/string.

ao/sum			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/sum] method works with a vector of objects as input.	Test that the [ao/sum] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/sum] method works with a matrix of objects as input.	Test that the [ao/sum] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/sum] method works with a list of objects as input.	Test that the [ao/sum] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/sum] method works with a mix of different arrays of objects as input.	Tests that the [ao/sum] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/sum] method properly applies history.	Test that the result of applying the [ao/sum] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/sum]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/sum] method can modify the input AO.	Test that the [ao/sum] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/sum			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/sum] value of the copy 4) Check that out and amodi are the same	pass
108	Test that the [ao/sum] method uses the plist to get the axis. This is intended to test methods like ao/mean and ao/std which return different data types depending on which axis is selected.	Test that the [ao/sum] method uses the plist to get the axis.	pass
		1) Check that the [ao/sum] method applies to the x-axis 2) Check that the [ao/sum] method applies to the y-axis 3) Check that the [ao/sum] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/sum] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/sum] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/sum] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 136: Unit tests for ao/sum.

ao/sumjoin			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the sumjoin method works with a vector of AOs as input.	Test that the sumjoin method works for a vector of AOs as input.	pass
		1) Check that the output is exact one AO 2) Check that the output have the correct data. 3) Check the rebuilt object	pass
03	Tests that the sumjoin method works with a matrix of AOs as input.	Tests that the sumjoin method works with a matrix of AOs as input.	pass
		1) Check that the output is exact one AO 2) Check that the output have the correct data.	pass
04	Tests that the sumjoin method works with a list of AOs as input.	Tests that the sumjoin method works with a list of AOs as input.	pass
		1) Check that the output is exact one AO 2) Check that the output have the correct data.	pass
05	Tests that the sumjoin method works with a mix of different shaped AOs as input.	Tests that the sumjoin method works with a mix of different shaped AOs as input.	pass
		1) Check that the output is exact one AO 2) Check that the output have the correct data.	pass
06	Tests that the sumjoin method properly applies history.	Test that the result of applying the sumjoin method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'sumjoin'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the sumjoin method can modify the input AO.	Test that the sumjoin method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/sumjoin			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input have the new data 4) Check that out and amodi are the same	pass
08	Test the shape of the output.	Test that the sumjoin method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass

Table 137: Unit tests for ao/sumjoin.

ao/svd			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the svd method works with a vector of AOs as input.	Test that the svd method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data.	pass
03	Tests that the svd method works with a matrix of AOs as input.	Test that the svd method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the svd method works with a list of AOs as input.	Test that the svd method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the svd method works with a mix of different shaped AOs as input.	Test that the svd method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the svd method properly applies history.	Test that the result of applying the svd method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'svd'. 2) Check that the rebuilt object is the same object as 'out'.	pass
07	Tests that the svd method can modify the input AO.	Test that the svd method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at4' and 'ain' are now different. 2) Check that 'ain' is svd(at4).	pass

ao/svd			
08	Control the method with a plist.	Test that the svd method can modify the single axis controlled by the plist and the resuld can be processed back to an m-file.	pass
		1) Check that the svd method applies with different options. 2) Check that the re-built objects are the same object as 'out[1..2]'.	pass
09	Control the method with a plist.	Test that the svd method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the svd method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 138: Unit tests for ao/svd.

ao/t0			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the t0 method works with a vector of AOs as input.	The t0 method doesn't work with a vector of AOs. Nothing to do	pass
			pass
03	Tests that the t0 method works with a matrix of AOs as input.	The t0 method doesn't work with a matrix of AOs. Nothing to do	pass
			pass
04	Tests that the t0 method works with a list of AOs as input.	The t0 method doesn't work with a list of AOs. Nothing to do	pass
			pass
05	Tests that the t0 method works with a mix of different shaped AOs as input.	The t0 method can only return the t0 value of one AO. Nothing to do	pass
			pass
06	Tests that the t0 method properly applies history.	The t0 method doesn't change the AO, thus will no history added. Nothing to do	pass
			pass
07	Tests that the t0 method works for AOs with different data objects.	Test that the t0 method returns the t0 value for AOs with cdata, fsdata, tsdata and xydata objects.	pass
		1) Check the output.	pass

Table 139: Unit tests for ao/t0.

ao/tan			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/tan] method works with a vector of objects as input.	Test that the [ao/tan] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/tan] method works with a matrix of objects as input.	Test that the [ao/tan] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/tan] method works with a list of objects as input.	Test that the [ao/tan] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/tan] method works with a mix of different arrays of objects as input.	Tests that the [ao/tan] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/tan] method properly applies history.	Test that the result of applying the [ao/tan] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/tan]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/tan] method can modify the input AO.	Test that the [ao/tan] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/tan			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/tan] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/tan] method uses the plist to get the axis.	Test that the [ao/tan] method uses the plist to get the axis.	pass
		1) Check that the [ao/tan] method applies to the x-axis 2) Check that the [ao/tan] method applies to the y-axis 3) Check that the [ao/tan] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/tan] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/tan] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/tan] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 140: Unit tests for ao/tan.

ao/tfe			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the tfe method works with a vector of AOs as input.	Test that the tfe method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the tfe method doesn't work with a matrix of AOs as input.	Test that the tfe method doesn't work for a matrix of AOs as input.	pass
		1) Nothing to check	pass
04	Tests that the tfe method works with a list of AOs as input.	Test that the tfe method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the tfe method doesn't work with a mix of different shaped AOs as input.	Test that the tfe method doesn't work with an input of matrices and vectors and single AOs.	pass
		1) Nothing to check.	pass
06	Tests that the tfe method properly applies history.	Test that the result of applying the tfe method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'tfe'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the tfe method can not modify the input AO.	Test that the tfe method can not modify the input AO. The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
08	Test the shape of the output.	Test that the plus method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the output data doesn't change.	pass

ao/tfe			
09	Check that the tfe method pass back the output objects to a list of output variables or to a single variable.	This test is not longer necessary because the tfe method pass back always only one object.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the tfe method agrees with MATLAB's tfestimate when configured to use the same parameters.	Test that applying tfe works on two AOs.	pass
		1) Check that output agrees with the output of MATLAB's tfestimate.	pass
11	Check that the [ao/tfe] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Tests that differently sized data sets are treated properly	Test that applying tfe works on two AOs.	pass
		1) Check that tfe used the length of the shortest ao.	pass
17	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) tfe of the white noise	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) tfe of the white noise	pass
		1) Check that (calculated tfe yunits) equals [1/Hz]	pass
21	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) tfe of the 2 series,	1) Prepare the test tsdata: white noise from normal distribution + offset 2) tfe of the 2 series, without detrending, random window, set number of averages	pass
		1) Check that calculated navs are identical to those requested	pass
22	Tests the possibility to set the window, of averages rather than setting the Nfft: 1) white noise produced from uniform pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) tfe of the time series, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run tfe again, with the nfft used 6) compare the calculated objects	1) white noise produced from uniform pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) tfe of the time series, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run tfe again, with the nfft used	pass

ao/tfe			
		1) Check that calculated objects T1 and T2 are identical	pass
23	Tests the possibility to set the number of averages rather than setting the Nfft: 1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) tfe of the time series, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run tfe again, with the nfft used 6) compare navs, nfft, tfes	1) white noise produced from normal pdf, with: a given mean value and sigma (distribution's 1st and 2nd order) 2) tfe of the time series, without detrending, random window, random navs 3) get the number of averages 4) get the nfft used 5) run tfe again, with the nfft used 6) run tfe again, with conflicting parameters, and verify it uses nfft rather than navs	pass
		1) Check that calculated objects T1 and T2 are identical 2) Check that T3 used different values	pass
25	Tests handling of units: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 3) tfe of the white noise	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: white noise from normal distribution + offset 4) Assign a random unit 5) tfe of the white noise	pass
		1) Check that (calculated tfe yunits) equals [1/Hz]	pass
30	Tests handling of special cases: 1) white noise produced from normal pdf, with a given mean value and sigma (distribution's 1st and 2nd orders) 2) the same noise series 3) tfe of the white noise series 4) compares the output to unity	1) Prepare the test tsdata: white noise from normal distribution + offset 2) Assign a random unit 3) Prepare the test tsdata: the same data as 1) and 2) 4) tfe of the series	pass
		1) Check that calculated tfe equals 1	pass

Table 141: Unit tests for ao/tfe.

ao/times			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
rule1 (tsdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass

ao/times			
rule1 (cdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (tsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and tsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (fsdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and fsdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (xydata and cdata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule1 (cdata and xydata)	Tests the arithmetic operators rule 1.	Tests the arithmetic operators rule 1 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule1 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single tsdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector tsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector tsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single fsdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector fsdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector fsdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single xydata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector xydata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector xydata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (vector cdata and single cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule2 (single cdata and vector cdata)	Tests the arithmetic operators rule 2.	Tests the arithmetic operators rule 2 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule2 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector tsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector fsdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector xydata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector xydata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule3 (vector cdata and vector cdata)	Tests the arithmetic operators rule 3.	Tests the arithmetic operators rule 3 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule3 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector tsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector tsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector fsdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector fsdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector xydata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector xydata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule4 (vector cdata and vector cdata)	Tests the arithmetic operators rule 4.	Tests the arithmetic operators rule 4 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule4 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single tsdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP tsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP tsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single fsdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP fsdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP fsdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single xydata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP xydata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP xydata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (NxP cdata and single cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule5 (single cdata and NxP cdata)	Tests the arithmetic operators rule 5.	Tests the arithmetic operators rule 5 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule5 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule6 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 6.	Tests the arithmetic operators rule 6 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule6 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule7 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 7.	Tests the arithmetic operators rule 7 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule7 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector tsdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP tsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP tsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector fsdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP fsdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP fsdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector xydata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP xydata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP xydata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (NxP cdata and vector cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule8 (vector cdata and NxP cdata)	Tests the arithmetic operators rule 8.	Tests the arithmetic operators rule 8 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule8 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ tsdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP tsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP tsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ fsdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP fsdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP fsdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ xydata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP xydata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP xydata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxP cdata and NxQ cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule9 (NxQ cdata and NxP cdata)	Tests the arithmetic operators rule 9.	Tests the arithmetic operators rule 9 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule9 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP tsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP tsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP fsdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP fsdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP xydata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP xydata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass

ao/times			
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
rule10 (NxP cdata and NxP cdata)	Tests the arithmetic operators rule 10.	Tests the arithmetic operators rule 10 for each data type: xydata, fsdata, tsdata, cdata and useful combinations.	pass
		Here we test element-wise operator rule10 as in S2-AEI-TN-3059. 1) Check the data type of the resulting object. 2) Check the resulting object contains the correct values. 3) Check the error propagation. 4) Check the units of the output object. 5) Check the resulting object can be rebuilt.	pass
times			pass
			pass
tests (fsdata and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (AO no data and tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (tsdata and AO no data)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different fs in tsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (different x values in fsdata)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass
tests (negative test)	Tests all arithmetic operations which are not allowed.	Tests all arithmetic operations which are not allowed.	pass
		Here we test some negative cases.	pass

ao/times			

Table 142: Unit tests for ao/times.

ao/timeshift			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the timeshift method works with a vector of AOs as input.	Test that the timeshift method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the timeshift method works with a matrix of AOs as input.	Test that the timeshift method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the timeshift method works with a list of AOs as input.	Test that the timeshift method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the timeshift method works with a mix of different shaped AOs as input.	Test that the timeshift method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the timeshift method properly applies history.	Test that the result of applying the timeshift method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timeshift'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/timeshift			
07	Tests that the timeshift method can modify the input AO.	Test that the timeshift method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is timeshift(at1). 3) Check the algorithm	pass
08	Control the method with a plist.	Test that the timeshift method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the timeshift method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/timeshift] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 143: Unit tests for ao/timeshift.

ao/transpose			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/transpose] method works with a vector of objects as input.	Test that the [ao/transpose] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/transpose] method works with a matrix of objects as input.	Test that the [ao/transpose] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/transpose] method works with a list of objects as input.	Test that the [ao/transpose] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/transpose] method works with a mix of different arrays of objects as input.	Tests that the [ao/transpose] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/transpose] method properly applies history.	Test that the result of applying the [ao/transpose] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/transpose]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the transpose method can modify the input AO.	Test that the transpose method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/transpose			
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is transpose(at1).	pass
09	Control the method with a plist.	Test that the abs method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
10	Check that the [ao/transpose] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/transpose] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 144: Unit tests for ao/transpose.

ao/type			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the type method works with a vector of AO objects as input.	Test that the type method works for a vector of AO objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the type method works with a matrix of AO objects as input.	Test that the type method works for a matrix of AO objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the type method works with a list of AO objects as input.	Test that the type method works for a list of AO objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the type method works with a mix of different shaped AO objects as input.	Test that the type method works with an input of matrices and vectors and single AO objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the type method properly applies history.	The method type doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 145: Unit tests for ao/type.

ao/uminus			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/uminus] method works with a vector of objects as input.	Test that the [ao/uminus] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/uminus] method works with a matrix of objects as input.	Test that the [ao/uminus] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/uminus] method works with a list of objects as input.	Test that the [ao/uminus] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/uminus] method works with a mix of different arrays of objects as input.	Tests that the [ao/uminus] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/uminus] method properly applies history.	Test that the result of applying the [ao/uminus] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/uminus]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/uminus] method can modify the input AO.	Test that the [ao/uminus] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/uminus			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/uminus] value of the copy 4) Check that out and amodi are the same	pass
08	Test that the [ao/uminus] method uses the plist to get the axis.	Test that the [ao/uminus] method uses the plist to get the axis.	pass
		1) Check that the [ao/uminus] method applies to the x-axis 2) Check that the [ao/uminus] method applies to the y-axis 3) Check that the [ao/uminus] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/uminus] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/uminus] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/uminus] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 146: Unit tests for ao/uminus.

ao/unwrap			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the unwrap method works with a vector of AOs as input.	Test that the unwrap method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'avec' 2) Check that each ouput corrects the phase angles	pass
03	Tests that the unwrap method works with a matrix of AOs as input.	Tests that the unwrap method works with a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'amat' 2) Check that each ouput corrects the phase angles	pass
04	Tests that the unwrap method works with a list of AOs as input.	Tests that the unwrap method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input 2) Check that each ouput corrects the phase angles	pass
05	Tests that the unwrap method works with a mix of different shaped AOs as input.	Tests that the unwrap method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input 2) Check that each ouput corrects the phase angles	pass
06	Tests that the unwrap method properly applies history.	Test that the result of applying the unwrap method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'unwrap'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the unwrap method can modify the input AO.	Test that the unwrap method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/unwrap			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the unwraped value of the copy 4) Check that out and amodi are the same	pass
08	Test that the unwrap method uses the plist to get the axis.	Test that the unwrap method uses the plist to get the axis.	pass
		1) Check that the unwrap method applies to the x-axis 2) Check that the unwrap method applies to the y-axis 3) Check that the unwrap method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the output.	Test that the unwrap method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the unwrap method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/unwrap] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 147: Unit tests for ao/unwrap.

ao/upsample			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the upsample method works with a vector of AOs as input.	Test that the upsample method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the upsample method works with a matrix of AOs as input.	Test that the upsample method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the upsample method works with a list of AOs as input.	Test that the upsample method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the upsample method works with a mix of different shaped AOs as input.	Test that the upsample method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the upsample method properly applies history.	Test that the result of applying the upsample method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'upsample'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/upsample			
07	Tests that the upsample method can modify the input AO.	Test that the upsample method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is upsample(at1). 3) Check the algorithm	pass
08	Control the method with a plist.	Test that the upsample method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the upsample method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/upsample] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 148: Unit tests for ao/upsample.

ao/var			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ao/var] method works with a vector of objects as input.	Test that the [ao/var] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ao/var] method works with a matrix of objects as input.	Test that the [ao/var] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ao/var] method works with a list of objects as input.	Test that the [ao/var] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ao/var] method works with a mix of different arrays of objects as input.	Tests that the [ao/var] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ao/var] method properly applies history.	Test that the result of applying the [ao/var] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ao/var]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the [ao/var] method can modify the input AO.	Test that the [ao/var] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/var			
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the [ao/var] value of the copy 4) Check that out and amodi are the same	pass
108	Test that the [ao/var] method uses the plist to get the axis. This is intended to test methods like ao/mean and ao/std which return different data types depending on which axis is selected.	Test that the [ao/var] method uses the plist to get the axis.	pass
		1) Check that the [ao/var] method applies to the x-axis 2) Check that the [ao/var] method applies to the y-axis 3) Check that the [ao/var] method applies to both axes 4) Check that the re-built object is the same as in 'out[1..3]'.	pass
09	Test the shape of the data in AOs.	Test that the [ao/var] method keeps the data shape of the input object. The input AO data must be an array with row data and/or column data.	pass
		1) Check that the shape of the data doesn't change.	pass
10	Check that the [ao/var] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/var] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 149: Unit tests for ao/var.

ao/whiten1D			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the whiten1D method works with a vector of AOs as input.	Test that the whiten1D method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atvec' 2) Check that each output AO contains the correct data. 3) Check that each output AO contains empty yunits	pass
03	Tests that the whiten1D method works with a matrix of AOs as input.	Test that the whiten1D method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data. 3) Check that each output AO contains empty yunits	pass
04	Tests that the whiten1D method works with a list of AOs as input.	Test that the whiten1D method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data. 3) Check that each output AO contains empty yunits	pass
05	Tests that the whiten1D method works with a mix of different shaped AOs as input.	Test that the whiten1D method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data. 3) Check that each output AO contains empty yunits	pass
06	Tests that the whiten1D method properly applies history.	Test that the result of applying the whiten1D method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'whiten1D'. 2) Check that the re-built object is the same object as the input.	pass

ao/whiten1D			
07	Tests that the whiten1D method can modify the input AO.	Test that the whiten1D method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is whiten1D(at1).	pass
08	Test the shape of the output.	Test that the whiten1D method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change. 2) Check that the output AOs have empty yunits	pass
09	Check that the whiten1D method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the whiten1D method is capable to increase spectral flatness in case of no model input	Generate a fixed series of noise data, apply a filter to colour data and run the method with a certain number of parameters.	pass
		1) Check that the output spectrum is flatter than input colored spectrum 2) Check that the output AOs have empty yunits	pass
11	Check that the whiten1D method is capable to increase spectral flatness when a model is input	Generate a fixed series of noise data, apply a filter to colour data and run the method with a certain number of parameters.	pass
		1) Check that the output spectrum is flatter than input colored spectrum	pass

Table 150: Unit tests for ao/whiten1D.

ao/whiten2D			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the whiten2D method works with a vector of AOs as input.	Test that the whiten2D method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'acv2' 2) Check that each output AO contains the correct data.	pass
03	Tests that the whiten2D method works with a matrix of AOs as input.	Test that the whiten2D method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'atmat' 2) Check that each output AO contains the correct data.	pass
04	Tests that the whiten2D method works with a list of AOs as input.	Test that the whiten2D method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the whiten2D method works with a mix of different shaped AOs as input.	Test that the whiten1D method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the whiten2D method properly applies history.	Test that the result of applying the whiten2D method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'whiten2D'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the whiten2D method can modify the input AO.	Test that the whiten2D method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Nothing to do.	pass

ao/whiten2D			
08	Test the shape of the output.	Test that the whiten2D method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the noisegen2D method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 151: Unit tests for ao/whiten2D.

ao/x			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the x method works with a vector of AOs as input.	The x method doesn't work with a vector of AOs. Nothing to do	pass
			pass
03	Tests that the x method works with a matrix of AOs as input.	The x method doesn't work with a matrix of AOs. Nothing to do	pass
			pass
04	Tests that the x method works with a list of AOs as input.	The x method doesn't work with a list of AOs. Nothing to do	pass
			pass
05	Tests that the x method works with a mix of different shaped AOs as input.	The x method can only return the x values of one AO. Nothing to do	pass
			pass
06	Tests that the x method properly applies history.	The x method doesn't change the AO, thus will no history added. Nothing to do	pass
			pass
07	Tests that the x method works for AOs with different data objects.	Test that the x method returns the x values for AOs with cdata, fsdata, tsdata and xydata objects.	pass
		1) Check the output.	pass
08	Tests that the x method returns the x values of the data object	Test that the x method returns the x values in a column vector independent form the shape of the values in the data object.	pass
		1) Check that 'x1' and 'x2' are column vectors.	pass

Table 152: Unit tests for ao/x.

ao/xcorr			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the xcorr method works with a vector with 2 AOs as input.	Test that the xcorr method works for a vector with 2 AOs as input.	pass
		1) Check that the numbers of outputs is 1 2) Check that the output AO contains the correct data. 3) Check that the units are properly handled	pass
03	Tests that the xcorr method does not work with a matrix of AOs as input.	Tests that the xcorr method does not work with a matrix of AOs as input.	pass
		1) Nothing to check	pass
04	Tests that the xcorr method works with a list of 2 AOs as input.	Tests that the xcorr method works with a list of 2 AOs as input.	pass
		1) Check that the numbers of outputs is 1 2) Check that the output AO contains the correct data. 3) Check that the units are properly handled	pass
05	Tests that the xcorr method does not work with a mix of different shaped AOs as input.	Tests that the xcorr method does not work with a mix of different shaped AOs as input.	pass
		1) Nothing to check	pass
06	Tests that the xcorr method properly applies history.	Test that the result of applying the xcorr method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'xcorr'. 2) Check that the rebuilt object is the same object as the input.	pass
07	Test the shape of the output.	Test that the xcorr method keeps the data shape of the input object. In this case the first AO defines the data shape.	pass
		1) Check that the shape of the data doesn't change.	pass

ao/xcorr			
08	Check that the xcorr method pass back the output objects to a list of output variables or to a single variable. Does not make sense anymore, given the structure double input -> single output	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output. Additionaly check that the units are handled properly. Additionaly check that the symmetry is preserved.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'. 3) Check that the units are handled properly. 4) Check that the symmetry is preserved.	pass
09	Check that the xcorr method re-sample all input AOs to the highest frequency.	Check that the xcorr method re-sample all input AOs to the highest frequency.	pass
		1) Check that the output contains the right number of objects 2) Check that the first AO is resampled to '200' 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the xcorr method truncate all input AOs to the same length.	Check that the xcorr method truncate all input AOs to the same length.	pass
		1) Check that the output contains the right number of objects 2) Check that the second AO is truncated to 10 seconds 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/xcorr] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass
12	Check that the xcorr uses different lag ranges [-maxlags:maxlags] Check that the xcorr accepts different scales options Check that the xcorr handles units properly	1) Check that the xcorr uses different lag ranges [-maxlags:maxlags] 2) Check that the xcorr accepts different scales options	pass
		1) Check that the output have the range [-maxlags:maxlags] 2) Check that the output have correct units	pass

ao/xcorr			
13	Check that the xcorr uses different lag ranges [-maxlags:maxlags]	Check that the xcorr uses different lag ranges [-maxlags:maxlags]	pass
		1) Check that the output have the range [-maxlags:maxlags]	pass

Table 153: Unit tests for ao/xcorr.

ao/xunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the xunits method works with a vector of AOs as input.	Tests that the xunits method works with a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output unit object contains the correct values.	pass
03	Tests that the xunits method works with a matrix of AOs as input.	Tests that the xunits method works with a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output unit object contains the correct values.	pass
04	Tests that the xunits method works with a list of AOs as input.	Tests that the xunits method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output unit object contains the correct values.	pass
05	Tests that the xunits method works with a mix of different shaped AOs as input.	Tests that the xunits method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output unit object contains the correct values.	pass
06	Tests that the xunits method properly applies history.	The xunits method doesn't change the AO, thus will no history added. Nothing to do	pass
			pass
07	Tests that the xunits method works for AOs with different data objects.	Test that the xunits method returns the xunits values for AOs with fsdata, tsdata and xydata objects, and empty units for AOs with cdata.	pass
		1) Check the output.	pass

ao/xunits			

Table 154: Unit tests for ao/xunits.

ao/y			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the y method works with a vector of AOs as input.	The y method doesn't work with a vector of AOs. Nothing to do	pass
			pass
03	Tests that the y method works with a matrix of AOs as input.	The y method doesn't work with a matrix of AOs. Nothing to do	pass
			pass
04	Tests that the y method works with a list of AOs as input.	The y method doesn't work with a list of AOs. Nothing to do	pass
			pass
05	Tests that the y method works with a mix of different shaped AOs as input.	The y method can only return the y values of one AO. Nothing to do	pass
			pass
06	Tests that the y method properly applies history.	The y method doesn't change the AO, thus will no history added. Nothing to do	pass
			pass
07	Tests that the y method works for AOs with different data objects.	Test that the y method returns the y values for AOs with cdata, fsdata, tsdata and xydata objects.	pass
		1) Check the output.	pass
08	Tests that the y method returns the y values of the data object	Test that the y method returns the y values in a column vector independent form the shape of the values in the data object.	pass
		1) Check that 'y1' and 'y2' are column vectors.	pass

Table 155: Unit tests for ao/y.

ao/yunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the yunits method works with a vector of AOs as input.	Tests that the yunits method works with a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output unit object contains the correct values.	pass
03	Tests that the yunits method works with a matrix of AOs as input.	Tests that the yunits method works with a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output unit object contains the correct values.	pass
04	Tests that the yunits method works with a list of AOs as input.	Tests that the yunits method works with a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output unit object contains the correct values.	pass
05	Tests that the yunits method works with a mix of different shaped AOs as input.	Tests that the yunits method works with a mix of different shaped AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input. 2) Check that each output unit object contains the correct values.	pass
06	Tests that the yunits method properly applies history.	The yunits method doesn't change the AO, thus will no history added. Nothing to do	pass
			pass
07	Tests that the yunits method works for AOs with different data objects.	Test that the yunits method returns the yunits values for AOs with cdata, fsdata, tsdata and xydata objects.	pass
		1) Check the output.	pass

Table 156: Unit tests for ao/yunits.

ao/zDomainFit			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the zDomainFit method works with a vector of AOs as input.	Test that the zDomainFit method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'av' 2) Check that each output AO contains the correct data.	pass
03	Tests that the zDomainFit method works with a matrix of AOs as input.	Test that the zDomainFit method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'am' 2) Check that each output AO contains the correct data.	pass
04	Tests that the zDomainFit method works with a list of AOs as input.	Test that the zDomainFit method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the zDomainFit method works with a mix of different shaped AOs as input.	Test that the zDomainFit method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the zDomainFit method properly applies history.	Test that the result of applying the zDomainFit method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'zDomainFit'. 2) Check that the re-built object is the same object as the input.	pass
07	zDomainFit cannot modify the input AO.	Test that the sDomainFit method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

ao/zDomainFit			
		1) Nothing to do.	pass
09	Check that the zDomainFit method pass back the output objects to a single variable correctly.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the zDomainFit method return the correct coefficients	Test that the zDomainFit method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that output contains the correct coefficients.	pass

Table 157: Unit tests for ao/zDomainFit.

ao/zeropad			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the zeropad method works with a vector of AOs as input.	Test that the zeropad method works for a vector of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
03	Tests that the zeropad method works with a matrix of AOs as input.	Test that the zeropad method works for a matrix of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
04	Tests that the zeropad method works with a list of AOs as input.	Test that the zeropad method works for a list of AOs as input.	pass
		1) Check that the number of elements in 'out' is the square of the number in the input. 2) Check that each output AO contains the correct data.	pass
05	Tests that the zeropad method works with a mix of different shaped AOs as input.	Test that the zeropad method works with an input of matrices and vectors and single AOs.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output AO contains the correct data.	pass
06	Tests that the zeropad method properly applies history.	Test that the result of applying the zeropad method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'zeropad'. 2) Check that the re-built object is the same object as 'out'.	pass

ao/zeropad			
07	Tests that the zeropad method can modify the input AO.	Test that the zeropad method can modify the input AO by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'at1' and 'ain' are now different. 2) Check that 'ain' is zeropad(at1).	pass
08	Check the data shape of the output against the input.	Test that the zeropad method keeps the data shape of the input object. The input AO must be an AO with row data and an AO with column data.	pass
		1) Check that the shpe of the data doesn't change.	pass
09	Check that the zeropad method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [ao/zeropad] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 158: Unit tests for ao/zeropad.

collection/collection			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [collection/collection] method works with a vector of objects as input.	Test that the [collection/collection] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [collection/collection] method works with a matrix of objects as input.	Test that the [collection/collection] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [collection/collection] method works with a list of objects as input.	Test that the [collection/collection] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [collection/collection] method works with a mix of different arrays of objects as input.	Tests that the [collection/collection] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [collection/collection] method properly applies history.	Test that the result of applying the [collection/collection] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[collection/collection]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the collection method properly applies history to the constructor with different numbers of inputs.	Test that the output can be processed back with the rebuild method.	pass

collection/collection			
		1) Check that the last entry in the history of 'out' corresponds to 'collection'. 2) Check that the rebuilt objects are the same	pass
08	Tests the collection constructor with different inputs.	Tests the collection constructor with different inputs.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'collection'. 2) Check that the rebuilt objects are the same	pass
60	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
61	Tests that the constructor properly applies history to the read XML-file constructor.	Tests that the constructor properly applies history to the read XML-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
62	Tests that the constructor properly applies history in the struct constructor.	Tests that the constructor properly applies history in the struct constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
64	Tests that the contructor properly applies history to the plist(filename) constructor.	Tests that the contructor properly applies history to the plist(filename) constructor.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
65	Tests that the contructed object can be submitted and retrieved.	Tests that the contructed object can be submitted and retrieved.	pass

collection/collection			
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
68	Tests that the constructor properly applies history to the conn+Id constructor.	Tests that the constructor properly applies history to the conn+Id constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
70	Tests that the constructor properly applies history to the plist(<plist-object>) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 159: Unit tests for collection/collection.

collection/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 160: Unit tests for collection/copy.

collection/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 161: Unit tests for collection/loadobj.

filterbank/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 162: Unit tests for filterbank/copy.

filterbank/filterbank			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [filterbank/filterbank] method works with a vector of objects as input.	Test that the [filterbank/filterbank] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [filterbank/filterbank] method works with a matrix of objects as input.	Test that the [filterbank/filterbank] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [filterbank/filterbank] method works with a list of objects as input.	Test that the [filterbank/filterbank] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [filterbank/filterbank] method works with a mix of different arrays of objects as input.	Tests that the [filterbank/filterbank] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [filterbank/filterbank] method properly applies history.	Test that the result of applying the [filterbank/filterbank] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[filterbank/filterbank]'. 2) Check that the re-built object is the same object as the input.	pass
60	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	pass

filterbank/filterbank			
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
61	Tests that the constructor properly applies history to the read XML-file constructor.	Tests that the constructor properly applies history to the read XML-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
62	Tests that the constructor properly applies history in the struct constructor.	Tests that the constructor properly applies history in the struct constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
64	Tests that the contructor properly applies history to the plist(filename) constructor.	Tests that the contructor properly applies history to the plist(filename) constructor.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
65	Tests that the contructed object can be submitted and retrieved.	Tests that the contructed object can be submitted and retrieved.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
68	Tests that the constructor properly applies history to the conn+Id constructor.	Tests that the constructor properly applies history to the conn+Id constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 163: Unit tests for filterbank/filterbank.

filterbank/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 164: Unit tests for filterbank/loadobj.

matrix/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of MATRIX objects as input.	Test that the char method works for a vector of MATRIX objects as input.	pass
		1) Check that the output contain at least the character set (char-method) of all inner object	pass
03	Tests that the char method works with a matrix of MATRIX objects as input.	Test that the char method works for a matrix of MATRIX objects as input.	pass
		1) Check that the output contain at least the character set (char-method) of all inner object	pass
04	Tests that the char method works with a list of MATRIX objects as input.	Test that the char method works for a list of MATRIX objects as input.	pass
		1) Check that the output contain at least the character set (char-method) of all inner object	pass
05	Tests that the char method works with a mix of different shaped MATRIX objects as input.	Test that the char method works with an input of matrices and vectors and single MATRIX objects.	pass
		1) Check that the output contain at least the character set (char-method) of all inner object	pass
06	Tests that the char method properly applies history.	The method char doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 165: Unit tests for matrix/char.

matrix/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 166: Unit tests for matrix/copy.

matrix/ctranspose			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ctranspose method works with a vector of MATRICES as input.	Test that the ctranspose method works for a vector of MATRICES as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mav' 2) Check that each output contains the correct data.	pass
03	Tests that the ctranspose method works with a matrix of MATRICES as input.	Test that the ctranspose method works for a matrix of MATRICES as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mam' 2) Check that each output contains the correct data.	pass
04	Tests that the ctranspose method works with a list of MATRICES as input.	Test that the ctranspose method works for a list of MATRICES as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the ctranspose method works with a mix of different shaped MATRICES as input.	Test that the ctranspose method works with an input of matrices and vectors and single MATRICES.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the ctranspose method properly applies history.	Test that the result of applying the ctranspose method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'ctranspose'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the ctranspose method can modify the input MATRIX.	Test that the ctranspose method can modify the input MATRIX by calling with no output.	pass

matrix/ctranspose			
		1) Check that 'out' and 'obj_eq' are now different. 2) Check that 'obj_eq' is not changed 3) Check that the modified input is the ctranspose value of the copy 4) Check that out and amodi are the same	pass
08	Tests that the ctranspose method accepts plist as an input.	Tests that the ctranspose method accepts plist as an input.	pass
		1) Check that the correct data 2) Check that the re-built object is the same object as 'out'.	pass
11	Check that the [matrix/ctranspose] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 167: Unit tests for matrix/ctranspose.

matrix/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of MATRIX objects as input.	Test that the display method works for a vector of MATRIX objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of MATRIX objects as input.	Test that the display method works for a matrix of MATRIX objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of MATRIX objects as input.	Test that the display method works for a list of MATRIX objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped MATRIX objects as input.	Test that the display method works with an input of matrices and vectors and single MATRIX objects as.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the display method properly applies history.	The method display doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 168: Unit tests for matrix/display.

matrix/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 169: Unit tests for matrix/loadobj.

matrix/matrix			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [matrix/matrix] method works with a vector of objects as input.	Test that the [matrix/matrix] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [matrix/matrix] method works with a matrix of objects as input.	Test that the [matrix/matrix] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [matrix/matrix] method works with a list of objects as input.	Test that the [matrix/matrix] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [matrix/matrix] method works with a mix of different arrays of objects as input.	Tests that the [matrix/matrix] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [matrix/matrix] method properly applies history.	Test that the result of applying the [matrix/matrix] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[matrix/matrix]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the matrix method properly applies history to the constructor with different numbers of inputs.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'matrix'. 2) Check that the re-built objects are the same	pass

matrix/matrix			
60	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
61	Tests that the constructor properly applies history to the read XML-file constructor.	Tests that the constructor properly applies history to the read XML-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
62	Tests that the constructor properly applies history in the struct constructor.	Tests that the constructor properly applies history in the struct constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
64	Tests that the contructor properly applies history to the plist(filename) constructor.	Tests that the contructor properly applies history to the plist(filename) constructor.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
65	Tests that the contructed object can be submitted and retrieved.	Tests that the contructed object can be submitted and retrieved.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
68	Tests that the constructor properly applies history to the conn+Id constructor.	Tests that the constructor properly applies history to the conn+Id constructor.	pass

matrix/matrix			
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
70	Tests that the constructor properly applies history to the plist(<plist-object>) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 170: Unit tests for matrix/matrix.

matrix/ncols			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ncols method works with a vector of MATRICES as input.	The ncols method doesn't work with a vector of MATRICES. Nothing to do	pass
			pass
03	Tests that the ncols method works with a matrix of MATRICES as input.	The ncols method doesn't work with a matrix of MATRICES. Nothing to do.	pass
			pass
04	Tests that the ncols method works with a list of MATRICES as input.	The ncols method doesn't work with a list of MATRICES. Nothing to do.	pass
			pass
05	Tests that the ncols method works with a mix of different shaped MATRICES as input.	The ncols method can only return the ncols values of one MATRIX. Nothing to do	pass
			pass
06	Tests that the ncols method properly applies history.	The ncols method doesn't change the MATRIX, thus will no history added. Nothing to do	pass
			pass
07	Tests that the ncols method returns the number of columns of the inner objects.	Tests that the ncols method returns the number of columns of the inner objects.	pass
		1) Check that n1 and n2 are correct.	pass

Table 171: Unit tests for matrix/ncols.

matrix/nrows			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the nrows method works with a vector of MATRICES as input.	The nrows method doesn't work with a vector of MATRICES. Nothing to do	pass
			pass
03	Tests that the nrows method works with a matrix of MATRICES as input.	The nrows method doesn't work with a matrix of MATRICES. Nothing to do.	pass
			pass
04	Tests that the nrows method works with a list of MATRICES as input.	The nrows method doesn't work with a list of MATRICES. Nothing to do.	pass
			pass
05	Tests that the nrows method works with a mix of different shaped MATRICES as input.	The nrows method can only return the nrows values of one MATRIX. Nothing to do	pass
			pass
06	Tests that the nrows method properly applies history.	The nrows method doesn't change the MATRIX, thus will no history added. Nothing to do	pass
			pass
07	Tests that the nrows method returns the number of columns of the inner objects.	Tests that the nrows method returns the number of columns of the inner objects.	pass
		1) Check that n1 and n2 are correct.	pass

Table 172: Unit tests for matrix/nrows.

matrix/osize			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the osize method works with a vector of MATRICES as input.	The osize method doesn't work with a vector of MATRICES. Nothing to do	pass
			pass
03	Tests that the osize method works with a matrix of MATRICES as input.	The osize method doesn't work with a matrix of MATRICES. Nothing to do.	pass
			pass
04	Tests that the osize method works with a list of MATRICES as input.	The osize method doesn't work with a list of MATRICES. Nothing to do.	pass
			pass
05	Tests that the osize method works with a mix of different shaped MATRICES as input.	The osize method can only return the osize values of one MATRIX. Nothing to do	pass
			pass
06	Tests that the osize method properly applies history.	The osize method doesn't change the MATRIX, thus will no history added. Nothing to do	pass
			pass
07	Tests that the osize method returns the number of columns of the inner objects.	Tests that the osize method returns the number of columns of the inner objects.	pass
		1) Check that n1 and n2 are correct.	pass

Table 173: Unit tests for matrix/osize.

matrix/setObjs			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [matrix/setObjs] method works with a vector of objects as input.	Test that the [matrix/setObjs] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [matrix/setObjs] method works with a matrix of objects as input.	Test that the [matrix/setObjs] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [matrix/setObjs] method works with a list of objects as input.	Test that the [matrix/setObjs] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [matrix/setObjs] method works with a mix of different arrays of objects as input.	Tests that the [matrix/setObjs] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [matrix/setObjs] method properly applies history.	Test that the result of applying the [matrix/setObjs] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[matrix/setObjs]'. 2) Check that the re-built object is the same object as the input.	pass
07	Tests that the setObjs method can modify the input MATRIX.	Test that the setObjs method can modify the input MATRIX by calling with no output.	pass

matrix/setObjs			
		1) Check that 'out' and 'obj_eq' are now different. 2) Check that 'obj_eq' is not changed 3) Check that the modified input is the setObjs value of the copy 4) Check that out and amodi are the same	pass
08	Tests that the setObjs method can set the property without a plist.	Test that the setObjs method can modify the property 'objs' without a plist.	pass
		1) Check that 'out' has the correct objs field 2) Check that the re-built object is the same object as 'out'.	pass
09	Check that the setObjs method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the [matrix/setObjs] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 174: Unit tests for matrix/setObjs.

matrix/transpose			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the transpose method works with a vector of MATRICES as input.	Test that the transpose method works for a vector of MATRICES as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mav' 2) Check that each output contains the correct data.	pass
03	Tests that the transpose method works with a matrix of MATRICES as input.	Test that the transpose method works for a matrix of MATRICES as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mam' 2) Check that each output contains the correct data.	pass
04	Tests that the transpose method works with a list of MATRICES as input.	Test that the transpose method works for a list of MATRICES as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the transpose method works with a mix of different shaped MATRICES as input.	Test that the transpose method works with an input of matrices and vectors and single MATRICES.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the transpose method properly applies history.	Test that the result of applying the transpose method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'transpose'. 2) Check that the re-built object is the same object as 'out'.	pass
07	Tests that the transpose method can modify the input MATRIX.	Test that the transpose method can modify the input MATRIX by calling with no output.	pass

matrix/transpose			
		1) Check that 'out' and 'obj_eq' are now different. 2) Check that 'obj_eq' is not changed 3) Check that the modified input is the transpose value of the copy 4) Check that out and amodi are the same	pass
08	Tests that the transpose method accepts plist as an input.	Tests that the transpose method accepts plist as an input.	pass
		1) Check that the correct data 2) Check that the re-built object is the same object as 'out'.	pass
11	Check that the [matrix/transpose] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the same plotinfo plist	pass

Table 175: Unit tests for matrix/transpose.

mfir/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of MFIR objects as input.	Test that the char method works for a vector of MFIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the char method works with a matrix of MFIR objects as input.	Test that the char method works for a matrix of MFIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the char method works with a list of MFIR objects as input.	Test that the char method works for a list of MFIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the char method works with a mix of different shaped MFIR objects as input.	Test that the char method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the char method properly applies history.	The method char doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 176: Unit tests for mfir/char.

mfir/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 177: Unit tests for mfir/copy.

mfir/created			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the created method works with a vector of MFIR objects as input.	Test that the created method works for a vector of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firv' 2) Check that each output contains the correct data.	pass
03	Tests that the created method works with a matrix of MFIR objects as input.	Test that the created method works for a matrix of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firm' 2) Check that each output contains the correct data.	pass
04	Tests that the created method works with a list of MFIR objects as input.	Test that the created method works for a list of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the created method works with a mix of different shaped MFIR objects as input.	Test that the created method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the created method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the created method can be used with the modify command.	Tests that the created method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the created method retruns always a well defined time object even for an empty input object.	Test that the created method with an empty 'MFIR object	pass
		1) Check that the output is a time object with a ell defined time.	pass

mfir/created			

Table 178: Unit tests for mfir/created.

mfir/creator			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the creator method works with a vector of MFIR objects as input.	Test that the creator method works for a vector of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firv' 2) Check that each output contains the correct data.	pass
03	Tests that the creator method works with a matrix of MFIR objects as input.	Test that the creator method works for a matrix of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firm' 2) Check that each output contains the correct data.	pass
04	Tests that the creator method works with a list of MFIR objects as input.	The creator method doesn't work for a list of MFIR objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the creator method works with a mix of different shaped MFIR objects as input.	The creator method doesn't work with different shaped input objects.	pass
		1) Nothing to test	pass
06	Tests that the creator method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the creator method can be used with the modify command.	Tests that the creator method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the creator method retruns all creator(s)/modifier(s) which are in the history.	Test that the creator method uses the option 'all' direct or in a plist. The test file must have the modifier 'first', 'second' and 'third'	pass
		1) Check that out1 contains only one creator 2) Check that out2 contain more creator/modifier	pass
09	Tests the negative case for the option 'all'.	Test that the creator method throws an error if the option 'all' is used in connection with a matrix/vector of MFIR objects.	pass

mfir/creator			
		1) Nothing to test.	pass

Table 179: Unit tests for mfir/creator.

mfir/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of MFIR objects as input.	Test that the display method works for a vector of MFIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of MFIR objects as input.	Test that the display method works for a matrix of MFIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of MFIR objects as input.	Test that the display method works for a list of MFIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped MFIR objects as input.	Test that the display method works with an input of matrices and vectors and single MFIR objects as.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the display method properly applies history.	The method display doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 180: Unit tests for mfir/display.

mfir/eq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eq method works with a vector of MFIR objects as input.	Test that the eq method works for a vector of MFIR objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
03	Tests that the eq method works with a matrix of MFIR objects as input.	Test that the eq method works for a matrix of MFIR objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
04	Tests that the eq method works with a list of MFIR objects as input.	The eq method doesn't works for a list of MFIR objects as input. Nothing to do.	pass
			pass
05	Tests that the eq method works with a mix of different shaped MFIR objects as input.	The eq method doesn't works for a list of MFIR objects as input. Nothing to do.	pass
			pass
06	Tests that the eq method properly applies history.	The eq method doesn't change the MFIR object, thus will no history added. Nothing to do	pass
			pass
07	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because fir is created at an other time.	pass
		1) Check the output.	pass
08	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'histout'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because fir is created at an other time.	pass
		1) Check the output.	pass

mfir/eq			
09	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'iunits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because fir is created at an other time.	pass
		1) Check the output.	pass
10	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'ounits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because fir is created at an other time.	pass
		1) Check the output.	pass
11	Test the eq method with an exception list which is in a plist.	Test that the eq method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 181: Unit tests for mfir/eq.

mfir/get			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the get method of the mfir class.	Test that the get returns returns the value of the specified property. Do this for all properties of the MFIR object.	pass
		1) Check the correct value of the output	pass
03	Tests that the get method works with a plist.	Test that the get returns returns the value of the specified property which is defined in a plist.	pass
		1) Check the correct value of the output	pass
04	Tests the get method of the mfir class.	Test that the get throws an error if the input are more than one MFIR object.	pass
		1) Nothing to test	pass

Table 182: Unit tests for mfir/get.

mfir/index			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the index method works with a vector of MFIR objects as input.	Test that the index method works for a vector of MFIR objects as input. The following indexing should work: I = [1 2 3] or (I/J) = [(1,1), (1,2), (1,3)]	pass
		1) Check that the index method selects the correct object.	pass
03	Tests that the index method works with a matrix of MFIR objects as input.	Test that the index method works for a matrix of MFIR objects as input. The following indexing should work: I = [1 3 5] or (I/J) = [(1,1), (1,2), (1,3)] [2 4 6] [(2,1), (2,2), (2,3)]	pass
		1) Check that the index method selects the correct object.	pass
04	Tests that the index method works with a list of MFIR objects as input.	The index method doesn't work for a list of MFIR objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the index method properly applies history.	Test that the result of index have an additional history step.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'index'.	pass
06	Tests that the index method works for the modifier command.	Tests that the index method works for the modifier command.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
07	Control the method with a plist.	Test that the index method can be controled with a plist.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
08	Test that the index method selects more objects if I have more indices.	Test that the index method selects more objects if I have more indices.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass

mfir/index			

Table 183: Unit tests for mfir/index.

mfir/isprop			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isprop method works with a vector of MFIR objects as input.	Test that the isprop method works for a vector of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firv' 2) Check that each output contains the correct data.	pass
03	Tests that the isprop method works with a matrix of MFIR objects as input.	Test that the isprop method works for a matrix of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firm' 2) Check that each output contains the correct data.	pass
04	Tests that the isprop method works with a list of MFIR objects as input.	Test that the isprop method works for a list of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the isprop method works with a mix of different shaped MFIR objects as input.	Test that the isprop method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the isprop method properly applies history.	The method isprop doesn't change the object, thus it is not necessary to apply history.	pass
			pass
07	Tests that the isprop method works for each property.	Test that the isprop method works for the properties: 'gd', 'ntaps', 'fs', 'infile', 'a', 'histout', 'iunits', 'ounits', 'hist', 'name'	pass
		1) Check that each output contains the correct data.	pass
08	Test the negative case and the not function command.	Test that the isprop method retrun false for a unknown property and for methods of the object.	pass

mfir/isprop			
		1) Check that each output contains the correct data.	pass

Table 184: Unit tests for mfir/isprop.

mfir/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 185: Unit tests for mfir/loadobj.

mfir/mfir			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the mfir method works with a vector of MFIR objects as input.	Test that the mfir method works with a vector of MFIR objects as input.	pass
		1) Check that the shape of the output MFIRs is the same as the input shape. 2) Check that each output MFIR is a copy of the input MFIR. 3) Check that the copy have an additional history step.	pass
03	Tests that the mfir method works with a matrix of MFIR objects as input.	Test that the mfir method works with a matrix of MFIR objects as input.	pass
		1) Check that the shape of the output MFIRs is the same as the input shape. 2) Check that each output MFIR is a copy of the input MFIR. 3) Check that the copy have an additional history step.	pass
04	Tests that the mfir method works with a list of MFIR objects as input.	Test that the mfir method works with a list of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output MFIR is a copy of the input MFIR. 3) Check that the copy have an additional history step.	pass
05	Tests that the mfir method works with a mix of different shaped MFIRs as input.	Test that the mfir method works with a mix of different shaped MFIRs as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output MFIR is a copy of the input MFIR. 3) Check that the copy have an additional history step.	pass
06	Tests that the mfir method properly applies history.	Test that the result of applying the mfir method can be processed back.	pass

mfir/mfir			
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the mfir method properly applies history to the copy constructor.	Test that the output can be processed back with the 'rebuild' method. Test the constructor with a different number of inputs.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the original objects are not changed by the setter function 3) Check that the method rebuild produces the same object as 'out'.	pass
08	Tests that the mfir method properly applies history to the read MAT-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Tests that the mfir method properly applies history to the read XML-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check the shape 2) Check that the last entry in the history of 'out' corresponds to 'mfir'. 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the mfir method properly applies history to the AO constructor.	Test that the output can be processed back with the 'rebuild' method. Use the default values (method = 'frequency-sampling').	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Tests that the mfir method properly applies history to the struct constructor.	Test that the output can be processed back with the 'rebuild' method.	pass

mfir/mfir			
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Tests that the mfir method properly applies history to the pzmodel constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Tests that the mfir method properly applies history to the plist(filename) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Tests that the MFIR method properly applies history to the plist(conn) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Tests that the MFIR method properly applies history to the plist(type) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
16	Tests that the MFIR method properly applies history to the plist(pzmodel) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
17	Tests that the MFIR method properly applies history to the plist(<plist-object>) constructor.	Test that the output can be processed back with the rebuild method.	pass

mfir/mfir			
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
18	Tests that the MFIR method properly applies history to the a constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
19	Tests that the mfir method properly applies history to the plist(AO) constructor.	Test that the output can be processed back with the 'rebuild' method. Use the method 'frequency-sampling'.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
20	Tests that the mfir method properly applies history to the plist(AO) constructor.	Test that the output can be processed back with the 'rebuild' method. Use the method 'least-squares'.	pass
		1) At the moment throws this method an error.	pass
21	Tests that the mfir method properly applies history to the plist(AO) constructor.	Test that the output can be processed back with the 'rebuild' method. Use the method 'Parks-McClellan'.	pass
		1) At the moment throws this method an error.	pass
22	Tests that the mfir method properly applies history to the AO + plist constructor.	Test that the output can be processed back with the 'rebuild' method. Use the method 'frequency-sampling'.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
23	Tests that the MFIR method properly applies history to the conn+Id constructor.	Test that the output can be processed back with the rebuild method.	pass

mfir/mfir			
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that re-running the 'test.m' file produces the same object as 'out'.	pass
24	Tests that the MFIR method properly applies history to the pole/zero model + plist object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
25	Tests that the MFIR method properly applies history to the a + fs object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mfir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 186: Unit tests for mfir/mfir.

mfir/ne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ne method works with a vector of MFIR objects as input.	Test that the ne method works for a vector of MFIR objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
03	Tests that the ne method works with a matrix of MFIR objects as input.	Test that the ne method works for a matrix of MFIR objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
04	Tests that the ne method works with a list of MFIR objects as input.	The ne method doesn't works for a list of MFIR objects as input. Nothing to do.	pass
			pass
05	Tests that the ne method works with a mix of different shaped MFIR objects as input.	The ne method doesn't works for a list of MFIR objects as input. Nothing to do.	pass
			pass
06	Tests that the ne method properly applies history.	The ne method doesn't change the MFIR object, thus will no history added. Nothing to do	pass
			pass
07	Test the ne method with an exception list. The function mfir/ne use the function mfir/eq so it is not necessary to check all possibilities of the exception list.	Test the ne method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because fir is created at an other time.	pass
		1) Check that each output contains the correct data.	pass
08	Test the ne method with an exception list which is in a plist.	Test that the ne method uses the exception list in a plist.	pass
		1) Check that each output contains the correct data.	pass

Table 187: Unit tests for mfir/ne.

mfir/rebuild			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rebuild method works with a vector of MFIR objects as input.	Test that the rebuild method works for a vector of MFIR objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the rebuild method works with a matrix of MFIR objects as input.	Test that the rebuild method works for a matrix of MFIR objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the rebuild method works with a list of MFIR objects as input.	Test that the rebuild method works for a list of MFIR objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the rebuild method works with a mix of different shaped MFIR objects as input.	Test that the rebuild method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the rebuild method properly applies history.	The method rebuild doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Check that the rebuild method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 188: Unit tests for mfir/rebuild.

mfir/redesign			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the redesign method works with a vector of MFIR objects as input.	Test that the redesign method works for a vector of MFIR objects as input. To keep this UTP simple use for the vector only one filter object. The different filters will be tested in an other UTP.	pass
		1) Check that the number of elements in 'out' is the same as in 'firvec' 2) Check that each output MFIR contains the correct data.	pass
03	Tests that the redesign method works with a matrix of MFIR objects as input.	Test that the redesign method works for a matrix of MFIR objects as input. To keep this UTP simple use for the matrix only one filter object. The different filters will be tested in an other UTP.	pass
		1) Check that the number of elements in 'out' is the same as in 'firmat' 2) Check that each output MFIR contains the correct data.	pass
04	Tests that the redesign method works with a list of MFIR objects as input.	Test that the redesign method works for a list of MFIR objects as input. To keep this UTP simple use for the list only one filter object. The different filters will be tested in an other UTP.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output MFIR contains the correct data.	pass
05	Tests that the redesign method works with a mix of different shaped MFIR objects as input.	Test that the redesign method works with an input of matrices and vectors and single MFIR objects. To keep this UTP simple use for the vector only one filter object. The different filters will be tested in an other UTP.	pass

mfir/redesign			
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output MFIR contains the correct data.	pass
06	Tests that the redesign method properly applies history.	Test that the result of applying the redesign method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'redesign'. 2) Check that rebuilt object is the same object as the input.	pass
07	Tests that the redesign method redesigns a standard filter type.	Tests that the redesign method redesigns a standard filter type.	pass
		1) Check the output 2) Check the rebuilt object	pass
08	Tests that the redesign method redesigns a pzmodel filter type.	Tests that the redesign method redesigns a pzmodel filter type.	pass
		1) Check the output 2) Check the rebuilt object	pass

Table 189: Unit tests for mfir/redesign.

mfir/resp			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the resp method works with a vector of MFIR objects as input.	Test that the resp method works for a vector of MFIR objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'firv' 3) Check that each output MFIR contains the correct data.	pass
03	Tests that the resp method works with a matrix of MFIR objects as input.	Test that the resp method works for a matrix of MFIR objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'firmat' 3) Check that each output MFIR contains the correct data.	pass
04	Tests that the resp method works with a list of MFIR objects as input.	Test that the resp method works for a list of MFIR objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'firmat' 3) Check that each output MFIR contains the correct data.	pass
05	Tests that the resp method works with a mix of different shaped MFIR objects as input.	Test that the resp method works with an input of matrices and vectors and single MFIR objects. Test the method with an output and with no output (a diagram must appear)	pass

mfir/resp			
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'firmat' 3) Check that each output MFIR contains the correct data.	pass
06	Tests that the resp method properly applies history.	Test that the result of applying the resp method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'resp'. 2) Check that re-built object is the same object as the input.	pass
07	Tests that modify command plots the response into a diagram.	Tests that modify command plots the response into a diagram.	pass
		1) Check the response diagram.	pass
08	Test the shape of the output.	Test that the output AO of the resp method keeps the shape of the used input f vector.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the resp method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the resp method uses the x-data of an input AO for f-vector.	Call the method with different method to pass an AO in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the resp method uses the specified f-vector to compute the response.	Call the method with different method to pass an f-vector in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass

mfir/resp			
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Check that the resp method the response of a serial filter bank.	Check that the resp method the response of a serial filter bank.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Check that the resp method the response of a parallel filter bank.	Check that the resp method the response of a parallel filter bank.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 190: Unit tests for mfir/resp.

mfir/save			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the save method works with a vector of MFIR objects as input.	Test that the save method works for a vector of MFIR objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'firv' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
03	Tests that the save method works with a matrix of MFIR objects as input.	Test that the save method works for a matrix of MFIR objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'firm' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
04	Tests that the save method works with a list of MFIR objects as input.	Test that the save method works for a list of MFIR objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in the list 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
05	Tests that the save method works with a mix of different shaped MFIR objects as input.	Test that the save method works with an input of matrices and vectors and single MFIR objects. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output MFIR object contains the correct data.	pass

mfir/save			
06	Tests that the save method properly applies history.	Test that the result of applying the save method can be processed back to an m-file. Do this for both extensions 'mat' and 'xml'	pass
		1) Check that the history applies to the output object. Check that save doesn't add a history step to the input object. 2) Check that the read object doesn't contain the save + load history steps. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the save method works with the modify command.	Use the save method with the modifier command.	pass
		1) Check that the save method applies the history. 2) Check the output against the input.	pass
08	Control the method with a plist.	Test that the save method uses the filename which is stored in a plist.	pass
		1) Check the output	pass
09	Test the save method with standard MFIR objects.	Save all standard MFIR objects with both extensions.	pass
		1) Check the output	pass
10	Test the save method with MFIR object which is created from a pole/zero model	Save MFIR object which is created from a pzmodel.	pass
		1) Check the output	pass
11	Test the save method with MFIR object which is created from an analysis model.	Save MFIR object which is created from an analysis model.	pass
		1) Check the output	pass

Table 191: Unit tests for mfir/save.

mfir/setHistout			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setHistout method works with a vector of MFIR objects as input.	The setHistout should set the output history (histout) of each input.	pass
		1) Check the histout has the correct values	pass
03	Tests that the setHistout method works with a matrix of MFIR objects as input.	The setHistout should set the output history (histout) of each input.	pass
		1) Check the histout has the correct values	pass
04	Tests that the setHistout method works with a list of MFIR objects as input.	The setHistout should set the output history (histout) of each input.	pass
		1) Check the histout has the correct values	pass
05	Tests that the setHistout method works with a mix of different shaped MFIR objects as input.	The setHistout method is not designed for this call, for that reason must this call fail.	pass
		1) Check the histout has the correct values	pass
06	Tests that the setHistout method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setHistout method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setHistout'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setHistout'. 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
07	Tests that the setHistout method can modify the input MFIR object.	Test that the setHistout method can modify the input MFIR object by calling with no output.	pass
		1) Check that 'firhp' and 'ain' are now different. 2) Check that 'ain' has the correct histout field	pass
08	Tests that the setHistout method can set the property with a plist.	Test that the setHistout method can modify the property 'histout' with a value in a plist.	pass

mfir/setHistout			
		1) Check that 'ain' has the correct histout field 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 192: Unit tests for mfir/setHistout.

mfir/setIunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setIunits method works with a vector of MFIR objects as input.	Test that the setIunits method works for a vector of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firv' 2) Check that each output contains the correct data.	pass
03	Tests that the setIunits method works with a matrix of MFIR objects as input.	Test that the setIunits method works for a matrix of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firm' 2) Check that each output contains the correct data.	pass
04	Tests that the setIunits method works with a list of MFIR objects as input.	Test that the setIunits method works for a list of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setIunits method works with a mix of different shaped MFIR objects as input.	Test that the setIunits method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setIunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setIunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setIunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setIunits'. 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
07	Tests that the setIunits method can modify the input MFIR object.	Test that the setIunits method can modify the input MFIR object by calling with no output.	pass

mfir/setIunits			
		1) Check that 'firhp' and 'ain' are now different. 2) Check that 'ain' has the correct iunits field	pass
08	Tests that the setIunits method can set the property with a plist.	Test that the setIunits method can modify the property 'iunits' with a value in a plist.	pass
		1) Check that 'ain' has the correct iunits field 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Check that the setIunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 193: Unit tests for mfir/setIunits.

mfir/setName			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setName method works with a vector of MFIR objects as input.	Test that the setName method works for a vector of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firv' 2) Check that each output contains the correct data.	pass
03	Tests that the setName method works with a matrix of MFIR objects as input.	Test that the setName method works for a matrix of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firm' 2) Check that each output contains the correct data.	pass
04	Tests that the setName method works with a list of MFIR objects as input.	Test that the setName method works for a list of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setName method works with a mix of different shaped MFIR objects as input.	Test that the setName method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setName method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setName method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setName'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setName'. 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
07	Tests that the setName method can modify the input MFIR object.	Test that the setName method can modify the input MFIR object by calling with no output.	pass

mfir/setName			
		1) Check that 'firhp' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
08	Tests that the setName method can set the property with a plist.	Test that the setName method can modify the property 'name' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Check that the setName method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 194: Unit tests for mfir/setName.

mfir/setOunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setOunits method works with a vector of MFIR objects as input.	Test that the setOunits method works for a vector of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firv' 2) Check that each output contains the correct data.	pass
03	Tests that the setOunits method works with a matrix of MFIR objects as input.	Test that the setOunits method works for a matrix of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'firm' 2) Check that each output contains the correct data.	pass
04	Tests that the setOunits method works with a list of MFIR objects as input.	Test that the setOunits method works for a list of MFIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setOunits method works with a mix of different shaped MFIR objects as input.	Test that the setOunits method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setOunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setOunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setOunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setOunits'. 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
07	Tests that the setOunits method can modify the input MFIR object.	Test that the setOunits method can modify the input MFIR object by calling with no output.	pass

mfir/setOunits			
		1) Check that 'firhp' and 'ain' are now different. 2) Check that 'ain' has the correct ounits field	pass
08	Tests that the setOunits method can set the property with a plist.	Test that the setOunits method can modify the property 'ounits' with a value in a plist.	pass
		1) Check that 'ain' has the correct ounits field 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Check that the setOunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 195: Unit tests for mfir/setOunits.

mfir/string			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the string method works with a vector of MFIR objects as input.	Test that the string method works for a vector of MFIR objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
03	Tests that the string method works with a matrix of MFIR objects as input.	Test that the string method works for a matrix of MFIR objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
04	Tests that the string method works with a list of MFIR objects as input.	Test that the string method works for a list of MFIR objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
05	Tests that the string method works with a mix of different shaped MFIR objects as input.	Test that the string method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
06	Tests that the string method properly applies history.	The method string doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Tests that the string method doesn't work if the MFIR object have more than one history step.	The method string throws an error because the input object have more than one history step.	pass
			pass

Table 196: Unit tests for mfir/string.

mfir/type			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the type method works with a vector of MFIR objects as input.	Test that the type method works for a vector of MFIR objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the type method works with a matrix of MFIR objects as input.	Test that the type method works for a matrix of MFIR objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the type method works with a list of MFIR objects as input.	Test that the type method works for a list of MFIR objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the type method works with a mix of different shaped MFIR objects as input.	Test that the type method works with an input of matrices and vectors and single MFIR objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the type method properly applies history.	The method type doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 197: Unit tests for mfir/type.

miir/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of MIIR objects as input.	Test that the char method works for a vector of MIIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the char method works with a matrix of MIIR objects as input.	Test that the char method works for a matrix of MIIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the char method works with a list of MIIR objects as input.	Test that the char method works for a list of MIIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the char method works with a mix of different shaped MIIR objects as input.	Test that the char method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the char method properly applies history.	The method char doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 198: Unit tests for miir/char.

miir/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 199: Unit tests for miir/copy.

miir/created			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the created method works with a vector of MIIR objects as input.	Test that the created method works for a vector of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirv' 2) Check that each output contains the correct data.	pass
03	Tests that the created method works with a matrix of MIIR objects as input.	Test that the created method works for a matrix of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirm' 2) Check that each output contains the correct data.	pass
04	Tests that the created method works with a list of MIIR objects as input.	Test that the created method works for a list of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the created method works with a mix of different shaped MIIR objects as input.	Test that the created method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the created method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the created method can be used with the modify command.	Tests that the created method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the created method retruns always a well defined time object even for an empty input object.	Test that the created method with an empty 'MIIR object	pass
		1) Check that the output is a time object with a ell defined time.	pass

miir/created			

Table 200: Unit tests for miir/created.

miir/creator			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the creator method works with a vector of MIIR objects as input.	Test that the creator method works for a vector of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirv' 2) Check that each output contains the correct data.	pass
03	Tests that the creator method works with a matrix of MIIR objects as input.	Test that the creator method works for a matrix of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirm' 2) Check that each output contains the correct data.	pass
04	Tests that the creator method works with a list of MIIR objects as input.	The creator method doesn't work for a list of MIIR objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the creator method works with a mix of different shaped MIIR objects as input.	The creator method doesn't work with different shaped input objects.	pass
		1) Nothing to test	pass
06	Tests that the creator method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the creator method can be used with the modify command.	Tests that the creator method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the creator method retruns all creator(s)/modifier(s) which are in the history.	Test that the creator method uses the option 'all' direct or in a plist. The test file must have the modifier 'first', 'second' and 'third'	pass
		1) Check that out1 contains only one creator 2) Check that out2 contain more creator/modifier	pass
09	Tests the negative case for the option 'all'.	Test that the creator method throws an error if the option 'all' is used in connection with a matrix/vector of MIIR objects.	pass

miir/creator			
		1) Nothing to test.	pass

Table 201: Unit tests for miir/creator.

miir/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of MIIR objects as input.	Test that the display method works for a vector of MIIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of MIIR objects as input.	Test that the display method works for a matrix of MIIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of MIIR objects as input.	Test that the display method works for a list of MIIR objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped MIIR objects as input.	Test that the display method works with an input of matrices and vectors and single MIIR objects as.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the display method properly applies history.	The method display doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 202: Unit tests for miir/display.

miir/eq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eq method works with a vector of MIIR objects as input.	Test that the eq method works for a vector of MIIR objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
03	Tests that the eq method works with a matrix of MIIR objects as input.	Test that the eq method works for a matrix of MIIR objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
04	Tests that the eq method works with a list of MIIR objects as input.	The eq method doesn't works for a list of MIIR objects as input. Nothing to do.	pass
			pass
05	Tests that the eq method works with a mix of different shaped MIIR objects as input.	The eq method doesn't works for a list of MIIR objects as input. Nothing to do.	pass
			pass
06	Tests that the eq method properly applies history.	The eq method doesn't change the MIIR object, thus will no history added. Nothing to do	pass
			pass
07	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because iir is created at an other time.	pass
		1) Check the output.	pass
08	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'histin'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because iir is created at an other time.	pass
		1) Check the output.	pass

miir/eq			
09	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'histout'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because iir is created at an other time.	pass
		1) Check the output.	pass
10	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'iunits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because iir is created at an other time.	pass
		1) Check the output.	pass
11	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'ounits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because iir is created at an other time.	pass
		1) Check the output.	pass
12	Test the eq method with an exception list which is in a plist.	Test that the eq method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 203: Unit tests for miir/eq.

miir/get			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the get method of the miir class.	Test that the get returns returns the value of the specified property. Do this for all properties of the MIIR object.	pass
		1) Check the correct value of the output	pass
03	Tests that the get method works with a plist.	Test that the get returns returns the value of the specified property which is defined in a plist.	pass
		1) Check the correct value of the output	pass
04	Tests the get method of the miir class.	Test that the get throws an error if the input are more than one MIIR object.	pass
		1) Nothing to test	pass

Table 204: Unit tests for miir/get.

miir/index			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the index method works with a vector of MIIR objects as input.	Test that the index method works for a vector of MIIR objects as input. The following indexing should work: I = [1 2 3] or (I/J) = [(1,1), (1,2), (1,3)]	pass
		1) Check that the index method selects the correct object.	pass
03	Tests that the index method works with a matrix of MIIR objects as input.	Test that the index method works for a matrix of MIIR objects as input. The following indexing should work: I = [1 3 5] or (I/J) = [(1,1), (1,2), (1,3)] [2 4 6] [(2,1), (2,2), (2,3)]	pass
		1) Check that the index method selects the correct object.	pass
04	Tests that the index method works with a list of MIIR objects as input.	The index method doesn't work for a list of MIIR objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the index method properly applies history.	Test that the result of index have an additional history step.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'index'.	pass
06	Tests that the index method works for the modifier command.	Tests that the index method works for the modifier command.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
07	Control the method with a plist.	Test that the index method can be controled with a plist.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
08	Test that the index method selects more objects if I have more indices.	Test that the index method selects more objects if I have more indices.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass

miir/index			

Table 205: Unit tests for miir/index.

miir/isprop			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isprop method works with a vector of MIIR objects as input.	Test that the isprop method works for a vector of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirv' 2) Check that each output contains the correct data.	pass
03	Tests that the isprop method works with a matrix of MIIR objects as input.	Test that the isprop method works for a matrix of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirm' 2) Check that each output contains the correct data.	pass
04	Tests that the isprop method works with a list of MIIR objects as input.	Test that the isprop method works for a list of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the isprop method works with a mix of different shaped MIIR objects as input.	Test that the isprop method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the isprop method properly applies history.	The method isprop doesn't change the object, thus it is not necessary to apply history.	pass
			pass
07	Tests that the isprop method works for each property.	Test that the isprop method works for the properties: 'b', 'histin', 'ntaps', 'fs', 'infile', 'a', 'histout', 'iunits', 'ounits', 'hist', 'name'	pass
		1) Check that each output contains the correct data.	pass

miir/isprop			
08	Test the negative case and the not function command.	Test that the isprop method retrun false for a unknown property and for methods of the object.	pass
		1) Check that each output contains the correct data.	pass

Table 206: Unit tests for miir/isprop.

miir/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 207: Unit tests for miir/loadobj.

miir/miir			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the miir method works with a vector of MIIR objects as input.	Test that the miir method works with a vector of MIIR objects as input.	pass
		1) Check that the shape of the output MIIRs is the same as the input shape. 2) Check that each output MIIR is a copy of the input MIIR. 3) Check that the copy have an additional history step.	pass
03	Tests that the miir method works with a matrix of MIIR objects as input.	Test that the miir method works with a matrix of MIIR objects as input.	pass
		1) Check that the shape of the output MIIRs is the same as the input shape. 2) Check that each output MIIR is a copy of the input MIIR. 3) Check that the copy have an additional history step.	pass
04	Tests that the miir method works with a list of MIIR objects as input.	Test that the miir method works with a list of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output MIIR is a copy of the input MIIR. 3) Check that the copy have an additional history step.	pass
05	Tests that the miir method works with a mix of different shaped MIIRs as input.	Test that the miir method works with a mix of different shaped MIIRs as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output MIIR is a copy of the input MIIR. 3) Check that the copy have an additional history step.	pass
06	Tests that the miir method properly applies history.	Test that the result of applying the miir method can be processed back.	pass

miir/miir			
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the miir method properly applies history to the copy constructor.	Test that the output can be processed back with the 'rebuild' method. Test the constructor with a different number of inputs.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the original objects are not changed by the setter function 3) Check that the method rebuild produces the same object as 'out'.	pass
08	Tests that the miir method properly applies history to the read MAT-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Tests that the miir method properly applies history to the read XML-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the miir method properly applies history to the read FIL-file constructor.	Read the FIL file which is created from LISO. Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Tests that the miir method properly applies history to the struct constructor.	Test that the output can be processed back with the 'rebuild' method.	pass

miir/miir			
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Tests that the miir method properly applies history to the parfrac constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check the correct number of outputs. 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Tests that the miir method properly applies history to the pzmodel constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Tests that the miir method properly applies history to the plist(filename) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Tests that the MIIR method properly applies history to the plist(conn) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
16	Tests that the MIIR method properly applies history to the plist(type) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
17	Tests that the MIIR method properly applies history to the plist(pzmodel) constructor.	Test that the output can be processed back with the rebuild method.	pass

miir/miir			
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
18	Tests that the MIIR method properly applies history to the plist(parfrac) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
19	Tests that the MIIR method properly applies history to the plist(<plist-object>) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
20	Tests that the MIIR method properly applies history to the a,b constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
21	Tests that the MIIR method properly applies history to the pole/zero model + plist object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
22	Tests that the MIIR method properly applies history to the partial fraction model + plist object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
23	Tests that the MIIR method properly applies history to the conn+Id constructor.	Test that the output can be processed back with the rebuild method.	pass

miir/miir			
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
24	Tests that the MIIR method properly applies history to the a + b + fs object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'miir'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 208: Unit tests for miir/miir.

miir/ne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ne method works with a vector of MIIR objects as input.	Test that the ne method works for a vector of MIIR objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
03	Tests that the ne method works with a matrix of MIIR objects as input.	Test that the ne method works for a matrix of MIIR objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
04	Tests that the ne method works with a list of MIIR objects as input.	The ne method doesn't works for a list of MIIR objects as input. Nothing to do.	pass
			pass
05	Tests that the ne method works with a mix of different shaped MIIR objects as input.	The ne method doesn't works for a list of MIIR objects as input. Nothing to do.	pass
			pass
06	Tests that the ne method properly applies history.	The ne method doesn't change the MIIR object, thus will no history added. Nothing to do	pass
			pass
07	Test the ne method with an exception list. The function miir/ne use the function miir/eq so it is not necessary to check all possibilities of the exception list.	Test the ne method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because iir is created at an other time.	pass
		1) Check that each output contains the correct data.	pass
08	Test the ne method with an exception list which is in a plist.	Test that the ne method uses the exception list in a plist.	pass
		1) Check that each output contains the correct data.	pass

Table 209: Unit tests for miir/ne.

miir/rebuild			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rebuild method works with a vector of MIIR objects as input.	Test that the rebuild method works for a vector of MIIR objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the rebuild method works with a matrix of MIIR objects as input.	Test that the rebuild method works for a matrix of MIIR objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the rebuild method works with a list of MIIR objects as input.	Test that the rebuild method works for a list of MIIR objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the rebuild method works with a mix of different shaped MIIR objects as input.	Test that the rebuild method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the rebuild method properly applies history.	The method rebuild doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Check that the rebuild method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 210: Unit tests for miir/rebuild.

miir/redesign			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the redesign method works with a vector of MIIR objects as input.	Test that the redesign method works for a vector of MIIR objects as input. To keep this UTP simple use for the vector only one filter object. The different filters will be tested in an other UTP.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirvec' 2) Check that each output MIIR contains the correct data.	pass
03	Tests that the redesign method works with a matrix of MIIR objects as input.	Test that the redesign method works for a matrix of MIIR objects as input. To keep this UTP simple use for the matrix only one filter object. The different filters will be tested in an other UTP.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirmat' 2) Check that each output MIIR contains the correct data.	pass
04	Tests that the redesign method works with a list of MIIR objects as input.	Test that the redesign method works for a list of MIIR objects as input. To keep this UTP simple use for the list only one filter object. The different filters will be tested in an other UTP.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output MIIR contains the correct data.	pass
05	Tests that the redesign method works with a mix of different shaped MIIR objects as input.	Test that the redesign method works with an input of matrices and vectors and single MIIR objects. To keep this UTP simple use for the vector only one filter object. The different filters will be tested in an other UTP.	pass

miir/redesign			
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output MIIR contains the correct data.	pass
06	Tests that the redesign method properly applies history.	Test that the result of applying the redesign method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'redesign'. 2) Check that rebuilt object is the same object as the input.	pass
07	Tests that the redesign method redesigns a standard filter type.	Tests that the redesign method redesigns a standard filter type.	pass
		1) Check the output 2) Check the rebuilt object	pass
08	Tests that the redesign method redesigns a pzmodel filter type.	Tests that the redesign method redesigns a pzmodel filter type.	pass
		1) Check the output 2) Check the rebuilt object	pass
09	Tests that the redesign method redesigns a parfrac filter type.	Tests that the redesign method redesigns a parfrac filter type.	pass
		1) Check the output 2) Check the rebuilt object	pass

Table 211: Unit tests for miir/redesign.

miir/resp			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the resp method works with a vector of MIIR objects as input.	Test that the resp method works for a vector of MIIR objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'iirv' 3) Check that each output MIIR contains the correct data.	pass
03	Tests that the resp method works with a matrix of MIIR objects as input.	Test that the resp method works for a matrix of MIIR objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'iirmat' 3) Check that each output MIIR contains the correct data.	pass
04	Tests that the resp method works with a list of MIIR objects as input.	Test that the resp method works for a list of MIIR objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'iirmat' 3) Check that each output MIIR contains the correct data.	pass
05	Tests that the resp method works with a mix of different shaped MIIR objects as input.	Test that the resp method works with an input of matrices and vectors and single MIIR objects. Test the method with an output and with no output (a diagram must appear)	pass

miir/resp			
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'iirmat' 3) Check that each output MIIR contains the correct data.	pass
06	Tests that the resp method properly applies history.	Test that the result of applying the resp method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'resp'. 2) Check that re-built object is the same object as the input.	pass
07	Tests that modify command plots the response into a diagram.	Tests that modify command plots the response into a diagram.	pass
		1) Check the response diagram.	pass
08	Test the shape of the output.	Test that the output AO of the resp method keeps the shape of the used input f vector.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the resp method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the resp method uses the x-data of an input AO for f-vector.	Call the method with different method to pass an AO in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the resp method uses the specified f-vector to compute the response.	Call the method with different method to pass an f-vector in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass

miir/resp			
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Check that the resp method the response of a serial filter bank.	Check that the resp method the response of a serial filter bank.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Check that the resp method the response of a parallel filter bank.	Check that the resp method the response of a parallel filter bank.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 212: Unit tests for miir/resp.

miir/save			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the save method works with a vector of MIIR objects as input.	Test that the save method works for a vector of MIIR objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'iirv' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
03	Tests that the save method works with a matrix of MIIR objects as input.	Test that the save method works for a matrix of MIIR objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'iirm' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
04	Tests that the save method works with a list of MIIR objects as input.	Test that the save method works for a list of MIIR objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in the list 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
05	Tests that the save method works with a mix of different shaped MIIR objects as input.	Test that the save method works with an input of matrices and vectors and single MIIR objects. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output MIIR object contains the correct data.	pass

miir/save			
06	Tests that the save method properly applies history.	Test that the result of applying the save method can be processed back to an m-file. Do this for both extensions 'mat' and 'xml'	pass
		1) Check that the history applies to the output object. Check that save doesn't add a history step to the input object. 2) Check that the read object doesn't contain the save + load history steps. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the save method works with the modify command.	Use the save method with the modifier command.	pass
		1) Check that the save method doesn't apply the history. 2) Check the output against the input.	pass
08	Control the method with a plist.	Test that the save method uses the filename which is stored in a plist.	pass
		1) Check the output	pass
09	Test the save method with standard MIIR objects.	Save all standard MIIR objects with both extensions.	pass
		1) Check the output	pass
10	Test the save method with MIIR object which is created from a pole/zero model	Save MIIR object which is created from a pzmodel.	pass
		1) Check the output	pass

Table 213: Unit tests for miir/save.

miir/setHistin			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setHistin method works with a vector of MIIR objects as input.	The setHistin method is not designed for vectors, for that reason must this call fail.	pass
		1) Nothing to check	pass
03	Tests that the setHistin method works with a matrix of MIIR objects as input.	The setHistin method is not designed for matrices, for that reason must this call fail.	pass
		1) Nothing to check	pass
04	Tests that the setHistin method works with a list of MIIR objects as input.	The setHistin method is not designed for input lists, for that reason must this call fail.	pass
		1) Nothing to check	pass
05	Tests that the setHistin method works with a mix of different shaped MIIR objects as input.	The setHistin method is not designed for this call, for that reason must this call fail.	pass
		1) Nothing to check	pass
06	Tests that the setHistin method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setHistin method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setHistin'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setHistin'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setHistin method can modify the input MIIR object.	Test that the setHistin method can modify the input MIIR object by calling with no output.	pass
		1) Check that 'iirhp' and 'ain' are now different. 2) Check that 'ain' has the correct histin field	pass
08	Tests that the setHistin method can set the property with a plist.	Test that the setHistin method can modify the property 'histin' with a value in a plist.	pass
		1) Check that 'ain' has the correct histin field 2) Check that the method rebuild produces the same object as 'out'.	pass

Table 214: Unit tests for miir/setHistin.

miir/setHistout			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setHistout method works with a vector of MIIR objects as input.	The setHistout should set the output history (histout) of each input.	pass
		1) Check the histout has the correct values	pass
03	Tests that the setHistout method works with a matrix of MIIR objects as input.	The setHistout should set the output history (histout) of each input.	pass
		1) Check the histout has the correct values	pass
04	Tests that the setHistout method works with a list of MIIR objects as input.	The setHistout should set the output history (histout) of each input.	pass
		1) Check the histout has the correct values	pass
05	Tests that the setHistout method works with a mix of different shaped MIIR objects as input.	The setHistout method is not designed for this call, for that reason must this call fail.	pass
		1) Check the histout has the correct values	pass
06	Tests that the setHistout method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setHistout method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setHistout'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setHistout'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setHistout method can modify the input MIIR object.	Test that the setHistout method can modify the input MIIR object by calling with no output.	pass
		1) Check that 'iirhp' and 'ain' are now different. 2) Check that 'ain' has the correct histout field	pass
08	Tests that the setHistout method can set the property with a plist.	Test that the setHistout method can modify the property 'histout' with a value in a plist.	pass

miir/setHistout			
		1) Check that 'ain' has the correct histout field 2) Check that the method rebuild produces the same object as 'out'.	pass

Table 215: Unit tests for miir/setHistout.

miir/setIunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setIunits method works with a vector of MIIR objects as input.	Test that the setIunits method works for a vector of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirv' 2) Check that each output contains the correct data.	pass
03	Tests that the setIunits method works with a matrix of MIIR objects as input.	Test that the setIunits method works for a matrix of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirm' 2) Check that each output contains the correct data.	pass
04	Tests that the setIunits method works with a list of MIIR objects as input.	Test that the setIunits method works for a list of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setIunits method works with a mix of different shaped MIIR objects as input.	Test that the setIunits method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setIunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setIunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setIunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setIunits'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setIunits method can modify the input MIIR object.	Test that the setIunits method can modify the input MIIR object by calling with no output.	pass

miir/setIunits			
		1) Check that 'iirhp' and 'ain' are now different. 2) Check that 'ain' has the correct iunits field	pass
08	Tests that the setIunits method can set the property with a plist.	Test that the setIunits method can modify the property 'iunits' with a value in a plist.	pass
		1) Check that 'ain' has the correct iunits field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setIunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 216: Unit tests for miir/setIunits.

miir/setName			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setName method works with a vector of MIIR objects as input.	Test that the setName method works for a vector of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirv' 2) Check that each output contains the correct data.	pass
03	Tests that the setName method works with a matrix of MIIR objects as input.	Test that the setName method works for a matrix of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirm' 2) Check that each output contains the correct data.	pass
04	Tests that the setName method works with a list of MIIR objects as input.	Test that the setName method works for a list of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setName method works with a mix of different shaped MIIR objects as input.	Test that the setName method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setName method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setName method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setName'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setName'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setName method can modify the input MIIR object.	Test that the setName method can modify the input MIIR object by calling with no output.	pass

miir/setName			
		1) Check that 'iirhp' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
08	Tests that the setName method can set the property with a plist.	Test that the setName method can modify the property 'name' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setName method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 217: Unit tests for miir/setName.

miir/setOunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setOunits method works with a vector of MIIR objects as input.	Test that the setOunits method works for a vector of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirv' 2) Check that each output contains the correct data.	pass
03	Tests that the setOunits method works with a matrix of MIIR objects as input.	Test that the setOunits method works for a matrix of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'iirm' 2) Check that each output contains the correct data.	pass
04	Tests that the setOunits method works with a list of MIIR objects as input.	Test that the setOunits method works for a list of MIIR objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setOunits method works with a mix of different shaped MIIR objects as input.	Test that the setOunits method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setOunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setOunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setOunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setOunits'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setOunits method can modify the input MIIR object.	Test that the setOunits method can modify the input MIIR object by calling with no output.	pass

miir/setOunits			
		1) Check that 'iirhp' and 'ain' are now different. 2) Check that 'ain' has the correct ounits field	pass
08	Tests that the setOunits method can set the property with a plist.	Test that the setOunits method can modify the property 'ounits' with a value in a plist.	pass
		1) Check that 'ain' has the correct ounits field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setOunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 218: Unit tests for miir/setOunits.

miir/string			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the string method works with a vector of MIIR objects as input.	Test that the string method works for a vector of MIIR objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
03	Tests that the string method works with a matrix of MIIR objects as input.	Test that the string method works for a matrix of MIIR objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
04	Tests that the string method works with a list of MIIR objects as input.	Test that the string method works for a list of MIIR objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
05	Tests that the string method works with a mix of different shaped MIIR objects as input.	Test that the string method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
06	Tests that the string method properly applies history.	The method string doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Tests that the string method doesn't work if the MIIR object have more than one history step.	The method string throws an error because the input object have more than one history step.	pass
			pass

Table 219: Unit tests for miir/string.

miir/type			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the type method works with a vector of MIIR objects as input.	Test that the type method works for a vector of MIIR objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the type method works with a matrix of MIIR objects as input.	Test that the type method works for a matrix of MIIR objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the type method works with a list of MIIR objects as input.	Test that the type method works for a list of MIIR objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the type method works with a mix of different shaped MIIR objects as input.	Test that the type method works with an input of matrices and vectors and single MIIR objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the type method properly applies history.	The method type doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 220: Unit tests for miir/type.

parfrac/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of PARFRAC objects as input.	Test that the char method works for a vector of PARFRAC objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the char method works with a matrix of PARFRAC objects as input.	Test that the char method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the char method works with a list of PARFRAC objects as input.	Test that the char method works for a list of PARFRAC objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the char method works with a mix of different shaped PARFRAC objects as input.	Test that the char method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the char method properly applies history.	The method char doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 221: Unit tests for parfrac/char.

parfrac/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 222: Unit tests for parfrac/copy.

parfrac/created			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the created method works with a vector of PARFRAC objects as input.	Test that the created method works for a vector of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfv' 2) Check that each output contains the correct data.	pass
03	Tests that the created method works with a matrix of PARFRAC objects as input.	Test that the created method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfm' 2) Check that each output contains the correct data.	pass
04	Tests that the created method works with a list of PARFRAC objects as input.	Test that the created method works for a list of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the created method works with a mix of different shaped PARFRAC objects as input.	Test that the created method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the created method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the created method can be used with the modify command.	Tests that the created method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the created method retruns always a well defined time object even for an empty input object.	Test that the created method with an empty 'PARFRAC object	pass

parfrac/created			
		1) Check that the output is a time object with a ell defined time.	pass

Table 223: Unit tests for parfrac/created.

parfrac/creator			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the creator method works with a vector of PARFRAC objects as input.	Test that the creator method works for a vector of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfv' 2) Check that each output contains the correct data.	pass
03	Tests that the creator method works with a matrix of PARFRAC objects as input.	Test that the creator method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfm' 2) Check that each output contains the correct data.	pass
04	Tests that the creator method works with a list of PARFRAC objects as input.	The creator method doesn't work for a list of PARFRAC objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the creator method works with a mix of different shaped PARFRAC objects as input.	The creator method doesn't work with different shaped input objects.	pass
		1) Nothing to test	pass
06	Tests that the creator method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the creator method can be used with the modify command.	Tests that the creator method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the creator method retruns all creator(s)/modifier(s) which are in the history.	Test that the creator method uses the option 'all' direct or in a plist. The test file must have the modifier 'first', 'second' and 'third'	pass
		1) Check that out1 contains only one creator 2) Check that out2 contain more creator/modifier	pass

parfrac/creator			
09	Tests the negative case for the option 'all'.	Test that the creator method throws an error if the option 'all' is used in connection with a matrix/vector of PARFRAC objects.	pass
		1) Nothing to test.	pass

Table 224: Unit tests for parfrac/creator.

parfrac/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of PARFRAC objects as input.	Test that the display method works for a vector of PARFRAC objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of PARFRAC objects as input.	Test that the display method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of PARFRAC objects as input.	Test that the display method works for a list of PARFRAC objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped PARFRAC objects as input.	Test that the display method works with an input of matrices and vectors and single PARFRAC objects as.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the display method properly applies history.	The method display doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 225: Unit tests for parfrac/display.

parfrac/eq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eq method works with a vector of PARFRAC objects as input.	Test that the eq method works for a vector of PARFRAC objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
03	Tests that the eq method works with a matrix of PARFRAC objects as input.	Test that the eq method works for a matrix of PARFRAC objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
04	Tests that the eq method works with a list of PARFRAC objects as input.	The eq method doesn't works for a list of PARFRAC objects as input. Nothing to do.	pass
			pass
05	Tests that the eq method works with a mix of different shaped PARFRAC objects as input.	The eq method doesn't works for a list of PARFRAC objects as input. Nothing to do.	pass
			pass
06	Tests that the eq method properly applies history.	The eq method doesn't change the PARFRAC object, thus will no history added. Nothing to do	pass
			pass
07	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because 'pa' is created at an other time.	pass
		1) Check the output.	pass
08	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'iunits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because 'pa' is created at an other time.	pass
		1) Check the output.	pass

parfrac/eq			
09	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'ounits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because 'pa' is created at an other time.	pass
		1) Check the output.	pass
10	Test the eq method with an exception list which is in a plist.	Test that the eq method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 226: Unit tests for parfrac/eq.

parfrac/get			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the get method of the parfrac class.	Test that the get returns returns the value of the specified property. Do this for all properties of the PARFRAC object.	pass
		1) Check the correct value of the output	pass
03	Tests that the get method works with a plist.	Test that the get returns returns the value of the specified property which is defined in a plist.	pass
		1) Check the correct value of the output	pass
04	Tests the get method of the parfrac class.	Test that the get throws an error if the input are more than one PARFRAC object.	pass
		1) Nothing to test	pass

Table 227: Unit tests for parfrac/get.

parfrac/getlowerFreq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the getlowerFreq method of the parfrac class.	Test that the getlowerFreq returns the lowest frequence of the pole in the parfrac object.	pass
		1) Check the output	pass
03	Tests the getlowerFreq method of the parfrac class.	Test that the getlowerFreq throws an error if the input are more than one parfrac.	pass
		1) Nothing to test	pass

Table 228: Unit tests for parfrac/getlowerFreq.

parfrac/getupperFreq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the getupperFreq method of the parfrac class.	Test that the getupperFreq returns the lowest frequence of the pole in the parfrac object.	pass
		1) Check the output	pass
03	Tests the getupperFreq method of the parfrac class.	Test that the getupperFreq throws an error if the input are more than one parfrac.	pass
		1) Nothing to test	pass

Table 229: Unit tests for parfrac/getupperFreq.

parfrac/index			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the index method works with a vector of PARFRAC objects as input.	Test that the index method works for a vector of PARFRAC objects as input. The following indexing should work: I = [1 2 3] or (I/J) = [(1,1), (1,2), (1,3)]	pass
		1) Check that the index method selects the correct object.	pass
03	Tests that the index method works with a matrix of PARFRAC objects as input.	Test that the index method works for a matrix of PARFRAC objects as input. The following indexing should work: I = [1 3 5] or (I/J) = [(1,1), (1,2), (1,3)] [2 4 6] [(2,1), (2,2), (2,3)]	pass
		1) Check that the index method selects the correct object.	pass
04	Tests that the index method works with a list of PARFRAC objects as input.	The index method doesn't work for a list of PARFRAC objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the index method properly applies history.	Test that the result of index have an additional history step.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'index'.	pass
06	Tests that the index method works for the modifier command.	Tests that the index method works for the modifier command.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
07	Control the method with a plist.	Test that the index method can be controled with a plist.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
08	Test that the index method selects more objects if I have more indices.	Test that the index method selects more objects if I have more indices.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass

parfrac/index			

Table 230: Unit tests for parfrac/index.

parfrac/isprop			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isprop method works with a vector of PARFRAC objects as input.	Test that the isprop method works for a vector of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfv' 2) Check that each output contains the correct data.	pass
03	Tests that the isprop method works with a matrix of PARFRAC objects as input.	Test that the isprop method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfm' 2) Check that each output contains the correct data.	pass
04	Tests that the isprop method works with a list of PARFRAC objects as input.	Test that the isprop method works for a list of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the isprop method works with a mix of different shaped PARFRAC objects as input.	Test that the isprop method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the isprop method properly applies history.	The method isprop doesn't change the object, thus it is not necessary to apply history.	pass
			pass
07	Tests that the isprop method works for each property.	Test that the isprop method works for the properties: 'res', 'poles', 'pmul', 'dir', 'iunits', 'ounits', 'hist', 'name'	pass
		1) Check that each output contains the correct data.	pass
08	Test the negative case and the not function command.	Test that the isprop method retrun false for a unknown property and for methods of the object.	pass

parfrac/isprop			
		1) Check that each output contains the correct data.	pass

Table 231: Unit tests for parfrac/isprop.

parfrac/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 232: Unit tests for parfrac/loadobj.

parfrac/ne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ne method works with a vector of PARFRAC objects as input.	Test that the ne method works for a vector of PARFRAC objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
03	Tests that the ne method works with a matrix of PARFRAC objects as input.	Test that the ne method works for a matrix of PARFRAC objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
04	Tests that the ne method works with a list of PARFRAC objects as input.	The ne method doesn't works for a list of PARFRAC objects as input. Nothing to do.	pass
			pass
05	Tests that the ne method works with a mix of different shaped PARFRAC objects as input.	The ne method doesn't works for a list of PARFRAC objects as input. Nothing to do.	pass
			pass
06	Tests that the ne method properly applies history.	The ne method doesn't change the PARFRAC object, thus will no history added. Nothing to do	pass
			pass
07	Test the ne method with an exception list. The function parfrac/ne use the function parfrac/eq so it is not necessary to check all possibilities of the exception list.	Test the ne method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because pf is created at an other time.	pass
		1) Check that each output contains the correct data.	pass
08	Test the ne method with an exception list which is in a plist.	Test that the ne method uses the exception list in a plist.	pass
		1) Check that each output contains the correct data.	pass

Table 233: Unit tests for parfrac/ne.

parfrac/parfrac			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the parfrac method works with a vector of PARFRAC objects as input.	Test that the parfrac method works with a vector of PARFRAC objects as input.	pass
		1) Check that the shape of the output PARFRACs is the same as the input shape. 2) Check that each output PARFRAC is a copy of the input PARFRAC. 3) Check that the copy have an additional history step.	pass
03	Tests that the parfrac method works with a matrix of PARFRAC objects as input.	Test that the parfrac method works with a matrix of PARFRAC objects as input.	pass
		1) Check that the shape of the output PARFRACs is the same as the input shape. 2) Check that each output PARFRAC is a copy of the input PARFRAC. 3) Check that the copy have an additional history step.	pass
04	Tests that the parfrac method works with a list of PARFRAC objects as input.	Test that the parfrac method works with a list of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output PARFRAC is a copy of the input PARFRAC. 3) Check that the copy have an additional history step.	pass
05	Tests that the parfrac method works with a mix of different shaped PARFRACs as input.	Test that the parfrac method works with a mix of different shaped PARFRACs as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output PARFRAC is a copy of the input PARFRAC. 3) Check that the copy have an additional history step.	pass
06	Tests that the parfrac method properly applies history.	Test that the result of applying the parfrac method can be processed back.	pass

parfrac/parfrac			
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the parfrac method properly applies history to the copy constructor.	Test that the output can be processed back with the 'rebuild' method. Test the constructor with a different number of inputs.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the original objects are not changed by the setter function 3) Check that the method rebuild produces the same object as 'out'.	pass
08	Tests that the parfrac method properly applies history to the read MAT-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the loaded object is the same as the saved object. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Tests that the parfrac method properly applies history to the read XML-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the loaded object is the same as the saved object. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the parfrac method properly doesn't apply history to the struct constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Tests that the parfrac method properly applies history to the rational constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

parfrac/parfrac			
12	Tests that the parfrac method properly applies history to the pzmodel constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Tests that the parfrac method properly applies history to the plist(filename) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Tests that the PARFRAC method properly applies history to the plist(conn) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Tests that the PARFRAC method properly applies history to the plist(res || poles || dir) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
16	Tests that the PARFRAC method properly applies history to the plist(pzmodel) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
17	Tests that the PARFRAC method properly applies history to the plist(parfrac) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
18	Tests that the PARFRAC method properly applies history to the plist(<plist-object>) constructor.	Test that the output can be processed back with the rebuild method.	pass

parfrac/parfrac			
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
19	Tests that the PARFRAC method properly applies history to the conn+Id constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
20	Tests that the PARFRAC method properly applies history to the res + poles + direct terms object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
21	Tests that the PARFRAC method properly applies history to the res + poles + direct terms + name object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
22	Tests that the PARFRAC method properly applies history to the res + poles + direct terms + name + iunits + ounits object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'parfrac'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 234: Unit tests for parfrac/parfrac.

parfrac/rebuild			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rebuild method works with a vector of PARFRAC objects as input.	Test that the rebuild method works for a vector of PARFRAC objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the rebuild method works with a matrix of PARFRAC objects as input.	Test that the rebuild method works for a matrix of PARFRAC objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the rebuild method works with a list of PARFRAC objects as input.	Test that the rebuild method works for a list of PARFRAC objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the rebuild method works with a mix of different shaped PARFRAC objects as input.	Test that the rebuild method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the rebuild method properly applies history.	The method rebuild doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Check that the rebuild method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 235: Unit tests for parfrac/rebuild.

parfrac/resp			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the resp method works with a vector of PARFRAC objects as input.	Test that the resp method works for a vector of PARFRAC objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'pfv' 3) Check that each output PARFRAC contains the correct data.	pass
03	Tests that the resp method works with a matrix of PARFRAC objects as input.	Tests that the resp method works with a matrix of PARFRAC objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'pfm' 3) Check that each output PARFRAC contains the correct data.	pass
04	Tests that the resp method works with a list of PARFRAC objects as input.	Tests that the resp method works with a list of PARFRAC objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'rain' 3) Check that each output PARFRAC contains the correct data.	pass
05	Tests that the resp method works with a mix of different shaped PARFRAC objects as input.	Tests that the resp method works with a mix of different shaped PARFRAC objects as input. Test the method with an output and with no output (a diagram must appear)	pass

parfrac/resp			
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'rain' 3) Check that each output PARFRAC contains the correct data.	pass
06	Tests that the resp method properly applies history.	Test that the result of applying the resp method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'resp'. 2) Check that re-built object is the same object as the input.	pass
07	Tests that modify command plots the response into a diagram.	Tests that modify command plots the response into a diagram.	pass
		1) Check the response diagram.	pass
08	Test the shape of the output.	Test that the output AO of the resp method keeps the shape of the used input f vector.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the resp method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the resp method uses the x-data of an input AO for f-vector.	Call the method with different method to pass an AO in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the resp method uses the specified f-vector to compute the response.	Call the method with different method to pass an f-vector in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass

parfrac/resp			
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 236: Unit tests for parfrac/resp.

parfrac/save			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the save method works with a vector of PARFRAC objects as input.	Test that the save method works for a vector of PARFRAC objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'pfv' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
03	Tests that the save method works with a matrix of PARFRAC objects as input.	Test that the save method works for a matrix of PARFRAC objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'pfm' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
04	Tests that the save method works with a list of PARFRAC objects as input.	Test that the save method works for a list of PARFRAC objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in the list 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
05	Tests that the save method works with a mix of different shaped PARFRAC objects as input.	Test that the save method works with an input of matrices and vectors and single PARFRAC objects. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output PARFRAC object contains the correct data.	pass

parfrac/save			
06	Tests that the save method properly applies history.	Test that the result of applying the save method can be processed back to an m-file. Do this for both extensions 'mat' and 'xml'	pass
		1) Check that the history applies to the output object. Check that save doesn't add a history step to the input object. 2) Check that the read object doesn't contain the save + load history steps. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the save method works with the modify command.	Use the save method with the modifier command.	pass
		1) Check that the save method doesn't apply the history. 2) Check the output against the input. 3) Check the history of the output against the input.	pass
08	Control the method with a plist.	Test that the save method uses the filename which is stored in a plist.	pass
		1) Check the output	pass
09	Test the save method with different complex PARFRAC objects	Test the save method with different complex PARFRAC objects	pass
		1) Check the output	pass

Table 237: Unit tests for parfrac/save.

parfrac/setIunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setIunits method works with a vector of PARFRAC objects as input.	Test that the setIunits method works for a vector of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfv' 2) Check that each output contains the correct data.	pass
03	Tests that the setIunits method works with a matrix of PARFRAC objects as input.	Test that the setIunits method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfm' 2) Check that each output contains the correct data.	pass
04	Tests that the setIunits method works with a list of PARFRAC objects as input.	Test that the setIunits method works for a list of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setIunits method works with a mix of different shaped PARFRAC objects as input.	Test that the setIunits method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setIunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setIunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setIunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setIunits'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setIunits method can modify the input PARFRAC object.	Test that the setIunits method can modify the input PARFRAC object by calling with no output.	pass

parfrac/setIunits			
		1) Check that 'pf3' and 'ain' are now different. 2) Check that 'ain' has the correct iunits field	pass
08	Tests that the setIunits method can set the property with a plist.	Test that the setIunits method can modify the property 'iunits' with a value in a plist.	pass
		1) Check that 'ain' has the correct iunits field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setIunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 238: Unit tests for parfrac/setIunits.

parfrac/setName			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setName method works with a vector of PARFRAC objects as input.	Test that the setName method works for a vector of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfv' 2) Check that each output contains the correct data.	pass
03	Tests that the setName method works with a matrix of PARFRAC objects as input.	Test that the setName method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfm' 2) Check that each output contains the correct data.	pass
04	Tests that the setName method works with a list of PARFRAC objects as input.	Test that the setName method works for a list of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setName method works with a mix of different shaped PARFRAC objects as input.	Test that the setName method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setName method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setName method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setName'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setName'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setName method can modify the input PARFRAC object.	Test that the setName method can modify the input PARFRAC object by calling with no output.	pass

parfrac/setName			
		1) Check that 'pf3' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
08	Tests that the setName method can set the property with a plist.	Test that the setName method can modify the property 'name' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setName method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 239: Unit tests for parfrac/setName.

parfrac/setOunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setOunits method works with a vector of PARFRAC objects as input.	Test that the setOunits method works for a vector of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfv' 2) Check that each output contains the correct data.	pass
03	Tests that the setOunits method works with a matrix of PARFRAC objects as input.	Test that the setOunits method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pfm' 2) Check that each output contains the correct data.	pass
04	Tests that the setOunits method works with a list of PARFRAC objects as input.	Test that the setOunits method works for a list of PARFRAC objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setOunits method works with a mix of different shaped PARFRAC objects as input.	Test that the setOunits method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setOunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setOunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setOunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setOunits'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setOunits method can modify the input PARFRAC object.	Test that the setOunits method can modify the input PARFRAC object by calling with no output.	pass

parfrac/setOunits			
		1) Check that 'pf3' and 'ain' are now different. 2) Check that 'ain' has the correct ounits field	pass
08	Tests that the setOunits method can set the property with a plist.	Test that the setOunits method can modify the property 'ounits' with a value in a plist.	pass
		1) Check that 'ain' has the correct ounits field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setOunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 240: Unit tests for parfrac/setOunits.

parfrac/string			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the string method works with a vector of PARFRAC objects as input.	Test that the string method works for a vector of PARFRAC objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
03	Tests that the string method works with a matrix of PARFRAC objects as input.	Test that the string method works for a matrix of PARFRAC objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
04	Tests that the string method works with a list of PARFRAC objects as input.	Test that the string method works for a list of PARFRAC objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
05	Tests that the string method works with a mix of different shaped PARFRAC objects as input.	Test that the string method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
06	Tests that the string method properly applies history.	The method string doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Tests that the string method doesn't work if the PARFRAC object have more than one history step.	The method string throws an error because the input object have more than one history step.	pass
			pass

Table 241: Unit tests for parfrac/string.

parfrac/type			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the type method works with a vector of PARFRAC objects as input.	Test that the type method works for a vector of PARFRAC objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the type method works with a matrix of PARFRAC objects as input.	Test that the type method works for a matrix of PARFRAC objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the type method works with a list of PARFRAC objects as input.	Test that the type method works for a list of PARFRAC objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the type method works with a mix of different shaped PARFRAC objects as input.	Test that the type method works with an input of matrices and vectors and single PARFRAC objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the type method properly applies history.	The method type doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 242: Unit tests for parfrac/type.

pest/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 243: Unit tests for pest/copy.

pest/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 244: Unit tests for pest/loadobj.

pest/pest			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [pest/pest] method works with a vector of objects as input.	Test that the [pest/pest] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [pest/pest] method works with a matrix of objects as input.	Test that the [pest/pest] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [pest/pest] method works with a list of objects as input.	Test that the [pest/pest] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [pest/pest] method works with a mix of different arrays of objects as input.	Tests that the [pest/pest] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [pest/pest] method properly applies history.	Test that the result of applying the [pest/pest] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/pest]'. 2) Check that the re-built object is the same object as the input.	pass
60	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	pass

pest/pest			
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
61	Tests that the constructor properly applies history to the read XML-file constructor.	Tests that the constructor properly applies history to the read XML-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
62	Tests that the constructor properly applies history in the struct constructor.	Tests that the constructor properly applies history in the struct constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
64	Tests that the contructor properly applies history to the plist(filename) constructor.	Tests that the contructor properly applies history to the plist(filename) constructor.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
65	Tests that the contructed object can be submitted and retrieved.	Tests that the contructed object can be submitted and retrieved.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
68	Tests that the constructor properly applies history to the conn+Id constructor.	Tests that the constructor properly applies history to the conn+Id constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 245: Unit tests for pest/pest.

pest/setChain			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setChain] method works for a list of objects as input.	Tests that the [pest/setChain] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setChain] method works for a list of objects as input.	Tests that the [pest/setChain] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setChain] method works for a list of objects as input.	Tests that the [pest/setChain] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setChain] method properly applies history.	Test that the result of applying the [pest/setChain] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setChain]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setChain			
genericModify	Tests that the [pest/setChain] method can modify the input AO.	Test that the [pest/setChain] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setChain] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 246: Unit tests for pest/setChain.

pest/setChi2			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setChi2] method works for a list of objects as input.	Tests that the [pest/setChi2] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setChi2] method works for a list of objects as input.	Tests that the [pest/setChi2] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setChi2] method works for a list of objects as input.	Tests that the [pest/setChi2] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setChi2] method properly applies history.	Test that the result of applying the [pest/setChi2] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setChi2]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setChi2			
genericModify	Tests that the [pest/setChi2] method can modify the input AO.	Test that the [pest/setChi2] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setChi2] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 247: Unit tests for pest/setChi2.

pest/setCorr			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setCorr] method works for a list of objects as input.	Tests that the [pest/setCorr] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setCorr] method works for a list of objects as input.	Tests that the [pest/setCorr] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setCorr] method works for a list of objects as input.	Tests that the [pest/setCorr] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setCorr] method properly applies history.	Test that the result of applying the [pest/setCorr] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setCorr]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setCorr			
genericModify	Tests that the [pest/setCorr] method can modify the input AO.	Test that the [pest/setCorr] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setCorr] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 248: Unit tests for pest/setCorr.

pest/setCov			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setCov] method works for a list of objects as input.	Tests that the [pest/setCov] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setCov] method works for a list of objects as input.	Tests that the [pest/setCov] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setCov] method works for a list of objects as input.	Tests that the [pest/setCov] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setCov] method properly applies history.	Test that the result of applying the [pest/setCov] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setCov]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setCov			
genericModify	Tests that the [pest/setCov] method can modify the input AO.	Test that the [pest/setCov] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setCov] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 249: Unit tests for pest/setCov.

pest/setDof			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setDof] method works for a list of objects as input.	Tests that the [pest/setDof] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setDof] method works for a list of objects as input.	Tests that the [pest/setDof] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setDof] method works for a list of objects as input.	Tests that the [pest/setDof] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setDof] method properly applies history.	Test that the result of applying the [pest/setDof] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setDof]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setDof			
genericModify	Tests that the [pest/setDof] method can modify the input AO.	Test that the [pest/setDof] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setDof] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 250: Unit tests for pest/setDof.

pest/setDy			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setDy] method works for a list of objects as input.	Tests that the [pest/setDy] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setDy] method works for a list of objects as input.	Tests that the [pest/setDy] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setDy] method works for a list of objects as input.	Tests that the [pest/setDy] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setDy] method properly applies history.	Test that the result of applying the [pest/setDy] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setDy]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setDy			
genericModify	Tests that the [pest/setDy] method can modify the input AO.	Test that the [pest/setDy] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setDy] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 251: Unit tests for pest/setDy.

pest/setModels			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setModels] method works for a list of objects as input.	Tests that the [pest/setModels] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setModels] method works for a list of objects as input.	Tests that the [pest/setModels] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setModels] method works for a list of objects as input.	Tests that the [pest/setModels] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setModels] method properly applies history.	Test that the result of applying the [pest/setModels] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setModels]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setModels			
genericModify	Tests that the [pest/setModels] method can modify the input AO.	Test that the [pest/setModels] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setModels] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 252: Unit tests for pest/setModels.

pest/setNames			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setNames] method works for a list of objects as input.	Tests that the [pest/setNames] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setNames] method works for a list of objects as input.	Tests that the [pest/setNames] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setNames] method works for a list of objects as input.	Tests that the [pest/setNames] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setNames] method properly applies history.	Test that the result of applying the [pest/setNames] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setNames]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setNames			
genericModify	Tests that the [pest/setNames] method can modify the input AO.	Test that the [pest/setNames] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setNames] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 253: Unit tests for pest/setNames.

pest/setPdf			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setPdf] method works for a list of objects as input.	Tests that the [pest/setPdf] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setPdf] method works for a list of objects as input.	Tests that the [pest/setPdf] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setPdf] method works for a list of objects as input.	Tests that the [pest/setPdf] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setPdf] method properly applies history.	Test that the result of applying the [pest/setPdf] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setPdf]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setPdf			
genericModify	Tests that the [pest/setPdf] method can modify the input AO.	Test that the [pest/setPdf] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setPdf] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 254: Unit tests for pest/setPdf.

pest/setY			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setY] method works for a list of objects as input.	Tests that the [pest/setY] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setY] method works for a list of objects as input.	Tests that the [pest/setY] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setY] method works for a list of objects as input.	Tests that the [pest/setY] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setY] method properly applies history.	Test that the result of applying the [pest/setY] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setY]'. 2) Check that the re-built object is the same object as the input.	pass

pest/setY			
genericModify	Tests that the [pest/setY] method can modify the input AO.	Test that the [pest/setY] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setY] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 255: Unit tests for pest/setY.

pest/setYunits			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShapeInternal			pass
			pass
genericList	Tests that the [pest/setYunits] method works for a list of objects as input.	Tests that the [pest/setYunits] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setYunits] method works for a list of objects as input.	Tests that the [pest/setYunits] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [pest/setYunits] method works for a list of objects as input.	Tests that the [pest/setYunits] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [pest/setYunits] method properly applies history.	Test that the result of applying the [pest/setYunits] method can be processed back.	pass

pest/setYunits			
		1) Check that the last entry in the history of 'out' corresponds to '[pest/setYunits]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [pest/setYunits] method can modify the input AO.	Test that the [pest/setYunits] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [pest/setYunits] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 256: Unit tests for pest/setYunits.

plist/append			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the append method works with a vector of PLIST objects as input.	Test that the append method appends all input PLIST objects to one object.	pass
		1) Check that the output is one PLIST object. 2) Check that the output PLIST contains all key/value pairs.	pass
03	Tests that the append method works with a matrix of PLIST objects as input.	Test that the append method appends all input PLIST objects to one object.	pass
		1) Check that the output is one PLIST object. 2) Check that the output PLIST contains all key/value pairs.	pass
04	Tests that the append method works with a list of PLIST objects as input.	Test that the append method appends all input PLIST objects to one object.	pass
		1) Check that the output is one PLIST object. 2) Check that the output PLIST contains all key/value pairs.	pass
05	Tests that the append method works with a mix of different shaped PLIST objects as input.	Test that the append method appends all input PLIST objects to one object.	pass
		1) Check that the output is one PLIST object. 2) Check that the output PLIST contains all key/value pairs.	pass
06	Tests that the append method applies the modify command	Test that the append method can modify the input PLIST by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'pl1' and 'plin' are now different. 2) Check that 'ain' append the key/value pair.	pass
07	Test the append method with different input variants for the key/value pair.	Test that the append method accepts plists-objects, param-objects or direct key/value pairs as an input.	pass
		1) Check that the output have all key/value pairs.	pass

plist/append			
08	Test that the append method appends the key always in capital letter	Test that the append method appends the key always in capital letter	pass
		1) Check that the key is in capital letters	pass
09	Test the append method in a negative case.	The append method throwns an error if a user tries to append a key which already exist.	pass
		1) Check that the negative case doesn't change the input object.	pass

Table 257: Unit tests for plist/append.

plist/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of PLIST objects as input.	Test that the char method works for a vector of PLIST objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the char method works with a matrix of PLIST objects as input.	Test that the char method works for a matrix of PLIST objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the char method works with a list of PLIST objects as input.	Test that the char method works for a list of PLIST objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the char method works with a mix of different shaped PLIST objects as input.	Test that the char method works with an input of matrices and vectors and single PLIST objects.	pass
		1) Check that the output contain at least each object name	pass

Table 258: Unit tests for plist/char.

plist/combine			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the combine method works with a vector of PLIST objects as input.	Test that the combine method combines all input PLIST objects to one object.	pass
		1) Check that the output is one PLIST object. 2) Check that the output PLIST contains all key/value pairs.	pass
03	Tests that the combine method works with a matrix of PLIST objects as input.	Test that the combine method combines all input PLIST objects to one object.	pass
		1) Check that the output is one PLIST object. 2) Check that the output PLIST contains all key/value pairs.	pass
04	Tests that the combine method works with a list of PLIST objects as input.	Test that the combine method combines all input PLIST objects to one object.	pass
		1) Check that the output is one PLIST object. 2) Check that the output PLIST contains all key/value pairs.	pass
05	Tests that the combine method works with a mix of different shaped PLIST objects as input.	Test that the combine method combines all input PLIST objects to one object.	pass
		1) Check that the output is one PLIST object. 2) Check that the output PLIST contains all key/value pairs.	pass
06	Tests that the combine method applies the modify command	Test that the combine method can modify the input PLIST by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'pl1' and 'plin' are now different. 2) Check that 'ain' combine the key/value pair.	pass
07	Test the combine method doen't overwrite existing keys.	Duplicate parameters which are given priority in the order in which they appear in the input.	pass
		1) Check that the output have all key/value pairs in the order they appear.	pass

plist/combine			

Table 259: Unit tests for plist/combine.

plist/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 260: Unit tests for plist/copy.

plist/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of PLIST objects as input.	Test that the display method works for a vector of PLIST objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of PLIST objects as input.	Test that the display method works for a matrix of PLIST objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of PLIST objects as input.	Test that the display method works for a list of PLIST objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped PLIST objects as input.	Test that the display method works with an input of matrices and vectors and single PLIST objects as.	pass
		1) Check that the output contain at least each object name	pass

Table 261: Unit tests for plist/display.

plist/eq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eq method works with a vector of PLIST objects as input.	Test that the eq method works for a vector of PLIST objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
03	Tests that the eq method works with a matrix of PLIST objects as input.	Test that the eq method works for a matrix of PLIST objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
04	Tests that the eq method works with a list of PLIST objects as input.	The eq method doesn't works for a list of PLIST objects as input. Nothing to do.	pass
			pass
05	Tests that the eq method works with a mix of different shaped PLIST objects as input.	The eq method doesn't works for a list of PLIST objects as input. Nothing to do.	pass
			pass
06	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'name'. It is necessary to add 'created' to the exception list because pl is created at an other time.	pass
		1) Check the output.	pass
07	Test the eq method with an exception list which is in a plist.	Test that the eq method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 262: Unit tests for plist/eq.

plist/find			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the find method applies the modify command.	Test that it is possible to use modify command and show that this method ignoring case.	pass
		1) Check that out1..3 have all the same object.	pass
03	Test that the find method accepts a plist.	Test that the find method take the 'key' from a 'search' plist.	pass
		1) Check that the output have the correct value.	pass
04	Test the find method returns an empty array if the 'key' doesn't exist in the PLIST.	Test the find method returns an empty array if the 'key' doesn't exist in the PLIST.	pass
		1) Check that the output is an empty array.	pass

Table 263: Unit tests for plist/find.

plist/get			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the get method of the plist class.	Test that the get returns returns the value of the specified property. Do this for all properties of the PLIST object.	pass
		1) Check the correct value of the output	pass
03	Tests that the get method works with a plist.	Test that the get returns returns the value of the specified property which is defined in a plist.	pass
		1) Check the correct value of the output	pass
04	Tests the get method of the plist class.	Test that the get throws an error if the input are more than one PLIST object.	pass
		1) Nothing to test	pass

Table 264: Unit tests for plist/get.

plist/isparam			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isparam method works with a vector of PLIST objects as input.	Tests that the isparam method works with a vector of PLIST objects as input.	pass
		1) Check that the output have the same size as the input. 2) Check the output.	pass
03	Tests that the isparam method works with a matrix of PLIST objects as input.	Tests that the isparam method works with a matrix of PLIST objects as input.	pass
		1) Check that the output have the same size as the input. 2) Check the output.	pass
04	Tests that the isparam method works with a list of PLIST objects as input.	Tests that the isparam method works with a list of PLIST objects as input.	pass
		1) Check that the output have the same size as the input. 2) Check the output.	pass
05	Tests that the isparam method works with a mix of different shaped PLIST objects as input.	Tests that the isparam method works with a mix of different shaped PLIST objects as input.	pass
		1) Check that the output have the same size as the input. 2) Check the output.	pass
06	Tests that the isparam method applies the modify command	Test that the isparam method can used in the modifier form.	pass
		1) Test that plin doesn't change.	pass
07	Test the isparam method with a positiv match.	Test the isparam method with different input.	pass
		1) Check the output	pass

Table 265: Unit tests for plist/isparam.

plist/isprop			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isprop method works with a vector of PLIST objects as input.	Test that the isprop method works for a vector of PLIST objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'plv' 2) Check that each output contains the correct data.	pass
03	Tests that the isprop method works with a matrix of PLIST objects as input.	Test that the isprop method works for a matrix of PLIST objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'plm' 2) Check that each output contains the correct data.	pass
04	Tests that the isprop method works with a list of PLIST objects as input.	Test that the isprop method works for a list of PLIST objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the isprop method works with a mix of different shaped PLIST objects as input.	Test that the isprop method works with an input of matrices and vectors and single PLIST objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the isprop method works for each property.	Test that the isprop method works for the properties: 'params', 'created', 'creator', 'name'	pass
		1) Check that each output contains the correct data.	pass
07	Test the negative case and the not function command.	Test that the isprop method retrun false for a unknown property and for methods of the object.	pass
		1) Check that each output contains the correct data.	pass

Table 266: Unit tests for plist/isprop.

plist/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 267: Unit tests for plist/loadobj.

plist/ne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ne method works with a vector of PLIST objects as input.	Test that the ne method works for a vector of PLIST objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
03	Tests that the ne method works with a matrix of PLIST objects as input.	Test that the ne method works for a matrix of PLIST objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
04	Tests that the ne method works with a list of PLIST objects as input.	The ne method doesn't works for a list of PLIST objects as input. Nothing to do.	pass
			pass
05	Tests that the ne method works with a mix of different shaped PLIST objects as input.	The ne method doesn't works for a list of PLIST objects as input. Nothing to do.	pass
			pass
06	Tests that the ne method properly applies history.	The ne method doesn't change the PLIST object, thus will no history added. Nothing to do	pass
			pass
07	Test the ne method with an exception list. The function plist/ne use the function plist/eq so it is not necessary to check all possibilities of the exception list.	Test the ne method with the exception 'name'. It is necessary to add 'created' to the exception list because pl is created at an other time.	pass
		1) Check that each output contains the correct data.	pass
08	Test the ne method with an exception list which is in a plist.	Test that the ne method uses the exception list in a plist.	pass
		1) Check that each output contains the correct data.	pass

Table 268: Unit tests for plist/ne.

plist/nparams			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the nparams method works with a vector of PLIST objects as input.	Test that the nparams method returns the number of PARAM objects in the PLIST objects.	pass
		1) Check that the number of outputs is the same as the number of input PLIST objects. 2) Check the output.	pass
03	Tests that the nparams method works with a vector of PLIST objects as input.	Test that the nparams method returns the number of PARAM objects in the PLIST objects.	pass
		1) Check that the number of outputs is the same as the number of input PLIST objects. 2) Check the output.	pass
04	Tests that the nparams method works with a list of PLIST objects as input.	Test that the nparams method returns the number of PARAM objects in the PLIST objects.	pass
		1) Check that the number of outputs is the same as the number of input PLIST objects. 2) Check the output.	pass
05	Tests that the nparams method works with a mix of different shaped PLIST objects as input.	Test that the nparams method returns the number of PARAM objects in the PLIST objects.	pass
		1) Check that the number of outputs is the same as the number of input PLIST objects. 2) Check the output.	pass
06	Tests that the nparams method applies the modify command	Test that the nparams method can used in the modifier style.	pass
		1) Check the output.	pass

Table 269: Unit tests for plist/nparams.

plist/parse			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the parse method works with a vector of PLIST objects as input.	The parse method is an internal method and it doesn't work for vector of PLIST objects as an input.	pass
		1) Nothing to test.	pass
03	Tests that the parse method works with a matrix of PLIST objects as input.	The parse method is an internal method and it doesn't work for matrix of PLIST objects as an input.	pass
		1) Nothing to test	pass
04	Tests that the parse method works with a list of PLIST objects as input.	The parse method is an internal method and it doesn't work for list of PLIST objects as an input.	pass
		1) Nothing to check.	pass
05	Tests that the parse method works with a mix of different shaped PLIST objects as input.	The parse method is an internal method and it doesn't work for different shaped PLIST objects as input.	pass
		1) Nothing to test.	pass
06	Tests that the parse method with non-dependent elements.	Test that the parse method can convert non-dependent elements.	pass
		1) Check the right number of parameter in out. 2) Check that the correct values are converted.	pass
07	Tests that the parse method with dependent elements.	Test that the parse method can convert dependent elements.	pass
		1) Check the right number of parameter in out. 2) Check that the correct values are converted.	pass
08	Tests that upper/lower case doesn't matter in the parse method.	Tests that upper/lower case doesn't matter in the parse method.	pass
		1) Check the right number of parameter in out. 2) Check that the correct values are converted.	pass

Table 270: Unit tests for plist/parse.

plist/plist			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the plist method works with a vector of PLIST objects as input.	Test that the plist method works with a vector of PLIST objects as input.	pass
		1) Check that the shape of the output PLISTs is the same as the input shape. 2) Check that each output PLIST is a copy of the input PLIST.	pass
03	Tests that the plist method works with a matrix of PLIST objects as input.	Test that the plist method works with a matrix of PLIST objects as input.	pass
		1) Check that the shape of the output PLISTs is the same as the input shape. 2) Check that each output PLIST is a copy of the input PLIST.	pass
04	Tests that the plist method works with a list of PLIST objects as input.	Test that the plist method works with a list of PLIST objects as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output PLIST is a copy of the input PLIST.	pass
05	Tests that the plist method works with a mix of different shaped PLISTs as input.	Test that the plist method works with a mix of different shaped PLISTs as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output PLIST is a copy of the input PLIST.	pass
06	Tests the MAT-file constructor.	Tests the MAT-file constructor.	pass
		1) Check that the saved object is the same as the loaded object.	pass
07	Tests the XML-file constructor.	Tests the XML-file constructor.	pass
		1) Check that the saved object is the same as the loaded object.	pass
08	Tests that the plist method for a struct as an input.	Tests that the plist method for a struct as an input.	pass
		2) Check the built object.	pass
09	Tests the param constructor.	Tests the param constructor.	pass

plist/plist			
		1) Check that the output contains the input parameter objects	pass
10	Tests the plist(filename) constructor.	Tests the plist(filename) constructor.	pass
		1) Check that the loaded object is the same as the saved object	pass
11	Tests the plist(conn) constructor.	Tests the plist(conn) constructor.	pass
		1) Check the retrieved object against the submitted object.	pass
12	Tests the conn+Id constructor.	Tests the conn+Id constructor.	pass
		1) Check the retrieved object against the submitted object.	pass
13	Tests the key/value constructor	Tests the key/value constructor	pass
		1) Check the right number of parameter in the PLIST object 2) Check the parameter in the PLIST object 3) Check that the key is in capital letter.	pass

Table 271: Unit tests for plist/plist.

plist/plist2cmds			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the plist2cmds method works with a vector of PLIST objects as input.	The plist2cmds method doesn't works for a vector of PLIST objects as input.	pass
		1) Nothing to test.	pass
03	Tests that the plist2cmds method works with a matrix of PLIST objects as input.	The plist2cmds method doesn't works for a matrix of PLIST objects as input.	pass
		1) Nothing to test.	pass
04	Tests that the plist2cmds method works with a list of PLIST objects as input.	The plist2cmds method doesn't works for a list of PLIST objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the plist2cmds method works with a mix of different shaped PLIST objects as input.	The plist2cmds method doesn't works with an input of matrices and vectors and single PLIST objects.	pass
		1) Nothing to test	pass
06	Tests that the plist2cmds method accepts different objects for the 'val' property.	Create a plist with all possible objects for the 'val' property.	pass
		1) Check that the output is a executable plist2cmds	pass

Table 272: Unit tests for plist/plist2cmds.

plist/pset			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the pset method works with a vector of PLIST objects as input.	Test that the pset method set or add a key/value pair to all PLIST objects in the vector.	pass
		1) Check that the shape of the output is the same as the shape of the input 2) Check that the output PLIST contains the new key/value pair.	pass
03	Tests that the pset method works with a matrix of PLIST objects as input.	Test that the pset method set or add a key/value pair to all PLIST objects in the matrix.	pass
		1) Check that the shape of the output is the same as the shape of the input 2) Check that the output PLIST contains the new key/value pair.	pass
04	Tests that the pset method works with a list of PLIST objects as input.	Test that the pset method set or add a key/value pair to all PLIST objects in the input.	pass
		1) Check that the shape of the output is the same as the shape of the input 2) Check that the output PLIST contains the new key/value pair.	pass
05	Tests that the pset method works with a mix of different shaped PLIST objects as input.	Test that the pset method set or add a key/value pair to all PLIST objects in the input.	pass
		1) Check that the shape of the output is the same as the shape of the input 2) Check that the output PLIST contains the new key/value pair.	pass
06	Tests that the pset method applies the modify command	Test that the pset method can modify the input PLIST by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

plist/pset			
		1) Check that 'out' and 'pleq' are now different. 2) Check that 'plmo' and 'out' are the same. 3) Check that 'out' and 'plmo' have the new key/value pair 4) Check that pleq don't have the new key/value pair	pass
07	Test the pset method works for different input variants for the key/value pair.	Test that the pset method accepts param-objects or direct key/value pairs as an input.	pass
		1) Check that the output have all key/value pairs.	pass
08	Test the pset method in the setting and appending case.	Create an example where the pset method set a key to a new value and an example where pset append the new key/value pair	pass
		1) Check the number of parametes in the output. 2) Check the new key/value pair	pass
09	Test that the pset method sets the key always in capital letter	Test that the pset method sets the key always in capital letter	pass
		1) Check that the key is in capital letters	pass

Table 273: Unit tests for plist/pset.

plist/remove			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the remove method works with a vector of PLIST objects as input.	Test that the remove method remove the 'key' from all PLIST objects in the vector.	pass
		1) Check that the shape of the output is the same as the shape of the input 2) Check that the output PLIST doesn't contains the key.	pass
03	Tests that the remove method works with a matrix of PLIST objects as input.	Test that the remove method remove the 'key' from all PLIST objects in the matrix.	pass
		1) Check that the shape of the output is the same as the shape of the input 2) Check that the output PLIST doesn't contains the key.	pass
04	Tests that the remove method works with a list of PLIST objects as input.	Test that the remove method remove the 'key' from all PLIST objects of the input.	pass
		1) Check that the shape of the output is the same as the shape of the input 2) Check that the output PLIST doesn't contains the key.	pass
05	Tests that the remove method works with a mix of different shaped PLIST objects as input.	Test that the remove method remove the 'key' from all PLIST objects of the input.	pass
		1) Check that the shape of the output is the same as the shape of the input 2) Check that the output PLIST doesn't contains the key.	pass
06	Tests that the remove method applies the modify command	Test that the remove method can modify the input PLIST by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

plist/remove			
		1) Check that 'out' and 'pleq' are now different. 2) Check that 'plmo' and 'out' are the same. 3) Check that 'out' and 'plmo' don't have the key 4) Check that pleq doesn't have the key	pass
07	Test the remove method that it removes the parameter which is defined as an index or as a key	Test the remove method that it removes the parameter which is defined as an index or as a key	pass
		1) Check the output.	pass
08	Test the remove method in a negative case that the key is not in the paramter list.	The remove method doesn't throwns an error if the key doesn't exist in the parameter list.	pass
		1) Nothing to test.	pass

Table 274: Unit tests for plist/remove.

plist/save			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the save method works with a vector of PLIST objects as input.	Test that the save method works for a vector of PLIST objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'plv' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
03	Tests that the save method works with a matrix of PLIST objects as input.	Test that the save method works for a matrix of PLIST objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'plm' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
04	Tests that the save method works with a list of PLIST objects as input.	Test that the save method works for a list of PLIST objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in the list 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
05	Tests that the save method works with a mix of different shaped PLIST objects as input.	Test that the save method works with an input of matrices and vectors and single PLIST objects. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output PLIST object contains the correct data.	pass

plist/save			
06	Tests that the save method works with the modify command.	Use the save method with the modifier command.	pass
		1) Check that the save method doesn't change the input PLIST object. 2) Check the output against the input except.	pass
07	Control the method with a plist.	Test that the save method uses the filename which is stored in a plist.	pass
		1) Check the output	pass

Table 275: Unit tests for plist/save.

plist/setName			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setName method works with a vector of PLIST objects as input.	Test that the setName method works for a vector of PLIST objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'plv' 2) Check that each output contains the correct data.	pass
03	Tests that the setName method works with a matrix of PLIST objects as input.	Test that the setName method works for a matrix of PLIST objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'plm' 2) Check that each output contains the correct data.	pass
04	Tests that the setName method works with a list of PLIST objects as input.	Test that the setName method works for a list of PLIST objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setName method works with a mix of different shaped PLIST objects as input.	Test that the setName method works with an input of matrices and vectors and single PLIST objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setName method can modify the input PLIST object.	Test that the setName method can modify the input PLIST object by calling with no output.	pass
		1) Check that 'pl2' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
07	Tests that the setName method can set the property with a plist.	Test that the setName method can modify the property 'name' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field	pass
08	Check that the setName method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable.	pass

plist/setName			
		1) Check that the output contains the right number of objects	pass

Table 276: Unit tests for plist/setName.

plist/string			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the string method works with a vector of PLIST objects as input.	Test that the string method works for a vector of PLIST objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
03	Tests that the string method works with a matrix of PLIST objects as input.	Test that the string method works for a matrix of PLIST objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
04	Tests that the string method works with a list of PLIST objects as input.	Test that the string method works for a list of PLIST objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
05	Tests that the string method works with a mix of different shaped PLIST objects as input.	Test that the string method works with an input of matrices and vectors and single PLIST objects.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
06	Tests that the string method doesn't work if the PLIST object have more than one history step.	The method string throws an error because the input object have more than one history step.	pass
			pass
07	Tests that the string method accepts different objects for the 'val' property.	Create a plist with all possible objects for the 'val' property.	pass
		1) Check that the output is a executable string	pass

Table 277: Unit tests for plist/string.

pzmodel/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of PZMODEL objects as input.	Test that the char method works for a vector of PZMODEL objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the char method works with a matrix of PZMODEL objects as input.	Test that the char method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the char method works with a list of PZMODEL objects as input.	Test that the char method works for a list of PZMODEL objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the char method works with a mix of different shaped PZMODEL objects as input.	Test that the char method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the char method properly applies history.	The method char doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 278: Unit tests for pzmodel/char.

pzmodel/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 279: Unit tests for pzmodel/copy.

pzmodel/created			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the created method works with a vector of PZMODEL objects as input.	Test that the created method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check that each output contains the correct data.	pass
03	Tests that the created method works with a matrix of PZMODEL objects as input.	Test that the created method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check that each output contains the correct data.	pass
04	Tests that the created method works with a list of PZMODEL objects as input.	Test that the created method works for a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the created method works with a mix of different shaped PZMODEL objects as input.	Test that the created method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the created method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the created method can be used with the modify command.	Tests that the created method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the created method retruns always a well defined time object even for an empty input object.	Test that the created method with an empty 'PZMODEL object	pass

pzmodel/created			
		1) Check that the output is a time object with a ell defined time.	pass

Table 280: Unit tests for pzmodel/created.

pzmodel/creator			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the creator method works with a vector of PZMODEL objects as input.	Test that the creator method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check that each output contains the correct data.	pass
03	Tests that the creator method works with a matrix of PZMODEL objects as input.	Test that the creator method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check that each output contains the correct data.	pass
04	Tests that the creator method works with a list of PZMODEL objects as input.	The creator method doesn't work for a list of PZMODEL objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the creator method works with a mix of different shaped PZMODEL objects as input.	The creator method doesn't work with different shaped input objects.	pass
		1) Nothing to test	pass
06	Tests that the creator method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the creator method can be used with the modify command.	Tests that the creator method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the creator method retruns all creator(s)/modifier(s) which are in the history.	Test that the creator method uses the option 'all' direct or in a plist. The test file must have the modifier 'first', 'second' and 'third'	pass
		1) Check that out1 contains only one creator 2) Check that out2 contain more creator/modifier	pass

pzmodel/creator			
09	Tests the negative case for the option 'all'.	Test that the creator method throws an error if the option 'all' is used in connection with a matrix/vector of PZMODEL objects.	pass
		1) Nothing to test.	pass

Table 281: Unit tests for pzmodel/creator.

pzmodel/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of PZMODEL objects as input.	Test that the display method works for a vector of PZMODEL objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of PZMODEL objects as input.	Test that the display method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of PZMODEL objects as input.	Test that the display method works for a list of PZMODEL objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped PZMODEL objects as input.	Test that the display method works with an input of matrices and vectors and single PZMODEL objects as.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the display method properly applies history.	The method display doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 282: Unit tests for pzmodel/display.

pzmodel/eq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eq method works with a vector of PZMODEL objects as input.	Test that the eq method works for a vector of PZMODEL objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
03	Tests that the eq method works with a matrix of PZMODEL objects as input.	Test that the eq method works for a matrix of PZMODEL objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
04	Tests that the eq method works with a list of PZMODEL objects as input.	The eq method doesn't works for a list of PZMODEL objects as input. Nothing to do.	pass
			pass
05	Tests that the eq method works with a mix of different shaped PZMODEL objects as input.	The eq method doesn't works for a list of PZMODEL objects as input. Nothing to do.	pass
			pass
06	Tests that the eq method properly applies history.	The eq method doesn't change the PZMODEL object, thus will no history added. Nothing to do	pass
			pass
07	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because pzm is created at an other time.	pass
		1) Check the output.	pass
08	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'iunits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because pzm is created at an other time.	pass
		1) Check the output.	pass

pzmodel/eq			
09	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'ounits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because pzm is created at an other time.	pass
		1) Check the output.	pass
10	Test the eq method with an exception list which is in a plist.	Test that the eq method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 283: Unit tests for pzmodel/eq.

pzmodel/get			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the get method of the pzmodel class.	Test that the get returns returns the value of the specified property. Do this for all properties of the PZMODEL object.	pass
		1) Check the correct value of the output	pass
03	Tests that the get method works with a plist.	Test that the get returns returns the value of the specified property which is defined in a plist.	pass
		1) Check the correct value of the output	pass
04	Tests the get method of the pzmodel class.	Test that the get throws an error if the input are more than one PZMODEL object.	pass
		1) Nothing to test	pass

Table 284: Unit tests for pzmodel/get.

pzmodel/getlowerFreq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the getlowerFreq method of the pzmodel class.	Test that the getlowerFreq returns the lowest frequence of the lowest pole or zero in the model.	pass
		1) Check the output	pass
03	Tests the getlowerFreq method of the pzmodel class.	Test that the getlowerFreq throws an error if the input are more than one pzmodel.	pass
		1) Nothing to test	pass

Table 285: Unit tests for pzmodel/getlowerFreq.

pzmodel/getupperFreq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the getupperFreq method of the pzmodel class.	Test that the getupperFreq returns the lowest frequence of the lowest pole or zero in the model.	pass
		1) Check the output	pass
03	Tests the getupperFreq method of the pzmodel class.	Test that the getupperFreq throws an error if the input are more than one pzmodel.	pass
		1) Nothing to test	pass

Table 286: Unit tests for pzmodel/getupperFreq.

pzmodel/index			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the index method works with a vector of PZMODEL objects as input.	Test that the index method works for a vector of PZMODEL objects as input. The following indexing should work: I = [1 2 3] or (I/J) = [(1,1), (1,2), (1,3)]	pass
		1) Check that the index method selects the correct object.	pass
03	Tests that the index method works with a matrix of PZMODEL objects as input.	Test that the index method works for a matrix of PZMODEL objects as input. The following indexing should work: I = [1 3 5] or (I/J) = [(1,1), (1,2), (1,3)] [2 4 6] [(2,1), (2,2), (2,3)]	pass
		1) Check that the index method selects the correct object.	pass
04	Tests that the index method works with a list of PZMODEL objects as input.	The index method doesn't work for a list of PZMODEL objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the index method properly applies history.	Test that the result of index have an additional history step.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'index'.	pass
06	Tests that the index method works for the modifier command.	Tests that the index method works for the modifier command.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
07	Control the method with a plist.	Test that the index method can be controled with a plist.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
08	Test that the index method selects more objects if I have more indices.	Test that the index method selects more objects if I have more indices.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass

pzmodel/index			

Table 287: Unit tests for pzmodel/index.

pzmodel/isprop			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isprop method works with a vector of PZMODEL objects as input.	Test that the isprop method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check that each output contains the correct data.	pass
03	Tests that the isprop method works with a matrix of PZMODEL objects as input.	Test that the isprop method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check that each output contains the correct data.	pass
04	Tests that the isprop method works with a list of PZMODEL objects as input.	Test that the isprop method works for a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the isprop method works with a mix of different shaped PZMODEL objects as input.	Test that the isprop method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the isprop method properly applies history.	The method isprop doesn't change the object, thus it is not necessary to apply history.	pass
			pass
07	Tests that the isprop method works for each property.	Test that the isprop method works for the properties: 'gain', 'poles', 'zeros', 'iunits', 'ounits', 'hist', 'name'	pass
		1) Check that each output contains the correct data.	pass
08	Test the negative case and the not function command.	Test that the isprop method retrun false for a unknown property and for methods of the object.	pass

pzmodel/isprop			
		1) Check that each output contains the correct data.	pass

Table 288: Unit tests for pzmodel/isprop.

pzmodel/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 289: Unit tests for pzmodel/loadobj.

pzmodel/mrdivide			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the mrdivide method works with a vector of PZMODELs as input.	Test that the mrdivide method works for a vector of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
03	Tests that the mrdivide method works with a matrix of PZMODELs as input.	Tests that the rdivie method works with a matrix of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
04	Tests that the mrdivide method works with a list of PZMODELs as input.	Tests that the mrdivide method works with a list of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
05	Tests that the mrdivide method works with a mix of different shaped PZMODELs as input.	Tests that the mrdivide method works with a mix of different shaped PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output 5) Check the rebuilt object	pass
06	Tests that the mrdivide method properly applies history.	Test that the result of applying the mrdivide method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'mrdivide'. 2) Check that rebuilt object is the same object as the input.	pass

pzmodel/mrdivide			
07	Check that the mrdivide method only divide PZMODELs with the same output units.	Check that the mrdivide method only divide PZMODELs with the same output units. Check also the negative case.	pass
		1) Check the I-/O-units	pass

Table 290: Unit tests for pzmodel/mrdivide.

pzmodel/mtimes			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the mtimes method works with a vector of PZMODELs as input.	Test that the mtimes method works for a vector of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
03	Tests that the mtimes method works with a matrix of PZMODELs as input.	Tests that the mtimes method works with a matrix of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
04	Tests that the mtimes method works with a list of PZMODELs as input.	Tests that the mtimes method works with a list of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
05	Tests that the mtimes method works with a mix of different shaped PZMODELs as input.	Tests that the mtimes method works with a mix of different shaped PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output 5) Check the rebuilt object	pass
06	Tests that the mtimes method properly applies history.	Test that the result of applying the mtimes method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'times'. 2) Check that re-built object is the same object as the input.	pass

pzmodel/mtimes			
07	Check that the mtimes method only multiply PZMODELs with the same output units.	Check that the mtimes method only multiply PZMODELs with the same output units. Check also the negative case.	pass
		1) Check the I-/O-units	pass

Table 291: Unit tests for pzmodel/mtimes.

pzmodel/ne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ne method works with a vector of PZMODEL objects as input.	Test that the ne method works for a vector of PZMODEL objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
03	Tests that the ne method works with a matrix of PZMODEL objects as input.	Test that the ne method works for a matrix of PZMODEL objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
04	Tests that the ne method works with a list of PZMODEL objects as input.	The ne method doesn't works for a list of PZMODEL objects as input. Nothing to do.	pass
			pass
05	Tests that the ne method works with a mix of different shaped PZMODEL objects as input.	The ne method doesn't works for a list of PZMODEL objects as input. Nothing to do.	pass
			pass
06	Tests that the ne method properly applies history.	The ne method doesn't change the PZMODEL object, thus will no history added. Nothing to do	pass
			pass
07	Test the ne method with an exception list. The function pzmodel/ne use the function pzmodel/eq so it is not necessary to check all possibilities of the exception list.	Test the ne method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because pzm is created at an other time.	pass
		1) Check that each output contains the correct data.	pass
08	Test the ne method with an exception list which is in a plist.	Test that the ne method uses the exception list in a plist.	pass
		1) Check that each output contains the correct data.	pass

Table 292: Unit tests for pzmodel/ne.

pzmodel/pzmodel			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the pzmodel method works with a vector of PZMODEL objects as input.	Test that the pzmodel method works with a vector of PZMODEL objects as input.	pass
		1) Check that the shape of the output PZMODELs is the same as the input shape. 2) Check that each output PZMODEL is a copy of the input PZMODEL. 3) Check that the copy have an additional history step.	pass
03	Tests that the pzmodel method works with a matrix of PZMODEL objects as input.	Test that the pzmodel method works with a matrix of PZMODEL objects as input.	pass
		1) Check that the shape of the output PZMODELs is the same as the input shape. 2) Check that each output PZMODEL is a copy of the input PZMODEL. 3) Check that the copy have an additional history step.	pass
04	Tests that the pzmodel method works with a list of PZMODEL objects as input.	Test that the pzmodel method works with a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output PZMODEL is a copy of the input PZMODEL. 3) Check that the copy have an additional history step.	pass
05	Tests that the pzmodel method works with a mix of different shaped PZMODELs as input.	Test that the pzmodel method works with a mix of different shaped PZMODELs as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output PZMODEL is a copy of the input PZMODEL. 3) Check that the copy have an additional history step.	pass
06	Tests that the pzmodel method properly applies history.	Test that the result of applying the pzmodel method can be processed back.	pass

pzmodel/pzmodel			
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the pzmodel method properly applies history to the copy constructor.	Test that the output can be processed back with the 'rebuild' method. Test the constructor with a different number of inputs.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the original objects are not changed by the setter function 3) Check that the method rebuild produces the same object as 'out'.	pass
08	Tests that the pzmodel method properly applies history to the read MAT-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the loaded object is the same as the saved object. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Tests that the pzmodel method properly applies history to the read XML-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the loaded object is the same as the saved object. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the pzmodel method properly applies history to the read FIL-file constructor.	Read the FIL file which is created from LISO. Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Tests that the pzmodel method properly applies history to the struct constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

pzmodel/pzmodel			
12	Tests that the pzmodel from a constant properly applies history.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Tests that the pzmodel method properly applies history to the rational constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Tests that the pzmodel method properly applies history to the plist(filename) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Tests that the PZMODEL method properly applies history to the plist(conn) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
16	Tests that the PZMODEL method properly applies history to the plist(rational) constructor	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check the algorithm 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
17	Tests that the PZMODEL method properly applies history to the plist(gain|poles|zeros) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

pzmodel/pzmodel			
18	Tests that the PZMODEL method properly applies history to the plist(<plist-object>) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
19	Tests that the PZMODEL method properly applies history to the conn+Id constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
20	Tests that the PZMODEL method properly applies history to the gain + poles + zeros constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
21	Tests that the PZMODEL method properly applies history to the gain + poles + zeros + name constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
22	Tests that the PZMODEL method properly applies history to the gain + poles + zeros + iunits + ounits constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'pzmodel'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 293: Unit tests for pzmodel/pzmodel.

pzmodel/rdivide			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rdivide method works with a vector of PZMODELs as input.	Test that the rdivide method works for a vector of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
03	Tests that the rdivide method works with a matrix of PZMODELs as input.	Tests that the rdivie method works with a matrix of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
04	Tests that the rdivide method works with a list of PZMODELs as input.	Tests that the rdivide method works with a list of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
05	Tests that the rdivide method works with a mix of different shaped PZMODELs as input.	Tests that the rdivide method works with a mix of different shaped PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output 5) Check the rebuilt object	pass
06	Tests that the rdivide method properly applies history.	Test that the result of applying the rdivide method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rdivide'. 2) Check that rebuilt object is the same object as the input.	pass

pzmodel/rdivide			
07	Check that the rdivide method only divide PZMODELs with the same output units.	Check that the rdivide method only divide PZMODELs with the same output units. Check also the negative case.	pass
		1) Check the I-/O-units	pass

Table 294: Unit tests for pzmodel/rdivide.

pzmodel/rebuild			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rebuild method works with a vector of PZMODEL objects as input.	Test that the rebuild method works for a vector of PZMODEL objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the rebuild method works with a matrix of PZMODEL objects as input.	Test that the rebuild method works for a matrix of PZMODEL objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the rebuild method works with a list of PZMODEL objects as input.	Test that the rebuild method works for a list of PZMODEL objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the rebuild method works with a mix of different shaped PZMODEL objects as input.	Test that the rebuild method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the rebuild method properly applies history.	The method rebuild doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Check that the rebuild method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 295: Unit tests for pzmodel/rebuild.

pzmodel/resp			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the resp method works with a vector of PZMODEL objects as input.	Test that the resp method works for a vector of PZMODEL objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'pzv' 3) Check that each output PZMODEL contains the correct data.	pass
03	Tests that the resp method works with a matrix of PZMODEL objects as input.	Test that the resp method works for a matrix of PZMODEL objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'pzmat' 3) Check that each output PZMODEL contains the correct data.	pass
04	Tests that the pzmat method works with a list of PZMODEL objects as input.	Test that the resp method works for a list of PZMODEL objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'pzmat' 3) Check that each output PZMODEL contains the correct data.	pass
05	Tests that the resp method works with a mix of different shaped PZMODEL objects as input.	Test that the resp method works with an input of matrices and vectors and single PZMODEL objects. Test the method with an output and with no output (a diagram must appear)	pass

pzmodel/resp			
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'pzmat' 3) Check that each output PZMODEL contains the correct data.	pass
06	Tests that the resp method properly applies history.	Test that the result of applying the resp method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'resp'. 2) Check that re-built object is the same object as the input.	pass
07	Tests that modify command plots the response into a diagram.	Tests that modify command plots the response into a diagram.	pass
		1) Check the response diagram.	pass
08	Test the shape of the output.	Test that the output AO of the resp method keeps the shape of the used input f vector.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the resp method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the resp method uses the x-data of an input AO for f-vector.	Call the method with different method to pass an AO in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the resp method uses the specified f-vector to compute the response.	Call the method with different method to pass an f-vector in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass

pzmodel/resp			
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 296: Unit tests for pzmodel/resp.

pzmodel/save			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the save method works with a vector of PZMODEL objects as input.	Test that the save method works for a vector of PZMODEL objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'pzv' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
03	Tests that the save method works with a matrix of PZMODEL objects as input.	Test that the save method works for a matrix of PZMODEL objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'pzm' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
04	Tests that the save method works with a list of PZMODEL objects as input.	Test that the save method works for a list of PZMODEL objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in the list 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
05	Tests that the save method works with a mix of different shaped PZMODEL objects as input.	Test that the save method works with an input of matrices and vectors and single PZMODEL objects. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output PZMODEL object contains the correct data.	pass

pzmodel/save			
06	Tests that the save method properly applies history.	Test that the result of applying the save method can be processed back to an m-file. Do this for both extensions 'mat' and 'xml'	pass
		1) Check that the history applies to the output object. Check that save doesn't add a history step to the input object. 2) Check that the read object doesn't contain the save + load history steps. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the save method works with the modify command.	Use the save method with the modifier command.	pass
		1) Check that the save method doesn't apply the history. 2) Check the output against the input. 3) Check the history of the output against the input.	pass
08	Control the method with a plist.	Test that the save method uses the filename which is stored in a plist.	pass
		1) Check the output	pass
09	Test the save method with common PZMODEL objects.	Save all common PZMODEL objects with both extensions.	pass
		1) Check the output	pass

Table 297: Unit tests for pzmodel/save.

pzmodel/setDelay			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setDelay method works with a vector of PZMODEL objects as input.	Test that the setDelay method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check that each output contains the correct data.	pass
03	Tests that the setDelay method works with a matrix of PZMODEL objects as input.	Test that the setDelay method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check that each output contains the correct data.	pass
04	Tests that the setDelay method works with a list of PZMODEL objects as input.	Test that the setDelay method works for a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setDelay method works with a mix of different shaped PZMODEL objects as input.	Test that the setDelay method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setDelay method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setDelay method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setDelay'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setDelay'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setDelay method can modify the input PZMODEL object.	Test that the setDelay method can modify the input PZMODEL object by calling with no output.	pass

pzmodel/setDelay			
		1) Check that 'pz5' and 'ain' are now different. 2) Check that 'ain' has the correct delay field	pass
08	Tests that the setDelay method can set the property with a plist.	Test that the setDelay method can modify the property 'delay' with a value in a plist.	pass
		1) Check that 'ain' has the correct delay field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setDelay method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 298: Unit tests for pzmodel/setDelay.

pzmodel/setIunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setIunits method works with a vector of PZMODEL objects as input.	Test that the setIunits method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check that each output contains the correct data.	pass
03	Tests that the setIunits method works with a matrix of PZMODEL objects as input.	Test that the setIunits method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check that each output contains the correct data.	pass
04	Tests that the setIunits method works with a list of PZMODEL objects as input.	Test that the setIunits method works for a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setIunits method works with a mix of different shaped PZMODEL objects as input.	Test that the setIunits method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setIunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setIunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setIunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setIunits'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setIunits method can modify the input PZMODEL object.	Test that the setIunits method can modify the input PZMODEL object by calling with no output.	pass

pzmodel/setIunits			
		1) Check that 'pz5' and 'ain' are now different. 2) Check that 'ain' has the correct iunits field	pass
08	Tests that the setIunits method can set the property with a plist.	Test that the setIunits method can modify the property 'iunits' with a value in a plist.	pass
		1) Check that 'ain' has the correct iunits field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setIunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 299: Unit tests for pzmodel/setIunits.

pzmodel/setName			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setName method works with a vector of PZMODEL objects as input.	Test that the setName method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check that each output contains the correct data.	pass
03	Tests that the setName method works with a matrix of PZMODEL objects as input.	Test that the setName method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check that each output contains the correct data.	pass
04	Tests that the setName method works with a list of PZMODEL objects as input.	Test that the setName method works for a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setName method works with a mix of different shaped PZMODEL objects as input.	Test that the setName method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setName method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setName method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setName'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setName'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setName method can modify the input PZMODEL object.	Test that the setName method can modify the input PZMODEL object by calling with no output.	pass

pzmodel/setName			
		1) Check that 'pz5' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
08	Tests that the setName method can set the property with a plist.	Test that the setName method can modify the property 'name' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setName method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 300: Unit tests for pzmodel/setName.

pzmodel/setOunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setOunits method works with a vector of PZMODEL objects as input.	Test that the setOunits method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check that each output contains the correct data.	pass
03	Tests that the setOunits method works with a matrix of PZMODEL objects as input.	Test that the setOunits method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check that each output contains the correct data.	pass
04	Tests that the setOunits method works with a list of PZMODEL objects as input.	Test that the setOunits method works for a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setOunits method works with a mix of different shaped PZMODEL objects as input.	Test that the setOunits method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setOunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setOunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setOunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setOunits'. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the setOunits method can modify the input PZMODEL object.	Test that the setOunits method can modify the input PZMODEL object by calling with no output.	pass

pzmodel/setOunits			
		1) Check that 'pz5' and 'ain' are now different. 2) Check that 'ain' has the correct ounits field	pass
08	Tests that the setOunits method can set the property with a plist.	Test that the setOunits method can modify the property 'ounits' with a value in a plist.	pass
		1) Check that 'ain' has the correct ounits field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setOunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 301: Unit tests for pzmodel/setOunits.

pzmodel/simplify			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the simplify method works with a vector of PZMODELs as input.	Test that the simplify method works for a vector of PZMODELs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check the poles of the output 3) Check the zeros of the output	pass
03	Tests that the simplify method works with a matrix of PZMODELs as input.	Tests that the simplify method works with a matrix of PZMODELs as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check the poles of the output 3) Check the zeros of the output	pass
04	Tests that the simplify method works with a list of PZMODELs as input.	Tests that the simplify method works with a list of PZMODELs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input 2) Check the poles of the output 3) Check the zeros of the output	pass
05	Tests that the simplify method works with a mix of different shaped PZMODELs as input.	Tests that the simplify method works with a mix of different shaped PZMODELs as input.	pass
		1) Check that the number of elements in 'out' is the same as in the input 2) Check the poles of the output 3) Check the zeros of the output 4) Check the re-built object	pass
06	Tests that the simplify method properly applies history.	Test that the result of applying the simplify method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'simplify'. 2) Check that re-built object is the same object as the input.	pass

pzmodel/simplify			
07	Tests that the simplify method can modify the input PZMODEL.	Test that the simplify method can modify the input PZMODEL by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that 'out' and 'aeq' are now different. 2) Check that 'aeq' is not changed 3) Check that the modified input is the simplified value of the copy 4) Check that out and amodi are the same	pass
08	Check that the simplify method cancle poles/zeros at different positions.	Check that the simplify method cancle poles/zeros at different positions.	pass
		1) Check the poles and peros of the output.	pass

Table 302: Unit tests for pzmodel/simplify.

pzmodel/string			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the string method works with a vector of PZMODEL objects as input.	Test that the string method works for a vector of PZMODEL objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
03	Tests that the string method works with a matrix of PZMODEL objects as input.	Test that the string method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
04	Tests that the string method works with a list of PZMODEL objects as input.	Test that the string method works for a list of PZMODEL objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
05	Tests that the string method works with a mix of different shaped PZMODEL objects as input.	Test that the string method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
06	Tests that the string method properly applies history.	The method string doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Tests that the string method doesn't work if the PZMODEL object have more than one history step.	The method string throws an error because the input object have more than one history step.	pass
			pass

Table 303: Unit tests for pzmodel/string.

pzmodel/times			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the times method works with a vector of PZMODELs as input.	Test that the times method works for a vector of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
03	Tests that the times method works with a matrix of PZMODELs as input.	Tests that the times method works with a matrix of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
04	Tests that the times method works with a list of PZMODELs as input.	Tests that the times method works with a list of PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output	pass
05	Tests that the times method works with a mix of different shaped PZMODELs as input.	Tests that the times method works with a mix of different shaped PZMODELs as input.	pass
		1) Check that the output is exact one PZMODEL object. 2) Check the gain of the output 3) Check the poles of the output 4) Check the zeros of the output 5) Check the rebuilt object	pass
06	Tests that the times method properly applies history.	Test that the result of applying the times method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'times'. 2) Check that re-built object is the same object as the input.	pass

pzmodel/times			
07	Check that the times method only multiply PZMODELs with the same output units.	Check that the times method only multiply PZMODELs with the same output units. Check also the negative case.	pass
		1) Check the I-/O-units	pass

Table 304: Unit tests for pzmodel/times.

pzmodel/tomfir			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the tomfir method works with a vector of PZMODEL objects as input.	Test that the tomfir method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv' 2) Check that each output PZMODEL object contains the correct data.	pass
03	Tests that the tomfir method works with a matrix of PZMODEL objects as input.	Test that the tomfir method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm' 2) Check that each output PZMODEL object contains the correct data.	pass
04	Tests that the tomfir method works with a list of PZMODEL objects as input.	Test that the tomfir method works for a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output PZMODEL object contains the correct data.	pass
05	Tests that the tomfir method works with a mix of different shaped PZMODEL objects as input.	Test that the tomfir method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output PZMODEL object contains the correct data.	pass
06	Tests that the tomfir method properly applies history.	Test that the result of applying the tomfir method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'tomfir'. 2) Check that re-built object is the same object as the input.	pass

pzmodel/tomfir			
07	Tests that the tomfir method can modify the input PZMODEL object.	Test that the tomfir method can not modify the input PZMODEL object . The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass
08	Control the method with a plist.	Test that the tomfir method use the different values in a plist.	pass
		1) Check the output 2) Check the re-built object	pass
09	Check that the tomfir method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 305: Unit tests for pzmodel/tomfir.

pzmodel/tomiir			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the tomiir method works with a vector of PZMODEL objects as input.	Test that the tomiir method works for a vector of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzv'	pass
03	Tests that the tomiir method works with a matrix of PZMODEL objects as input.	Test that the tomiir method works for a matrix of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'pzm'	pass
04	Tests that the tomiir method works with a list of PZMODEL objects as input.	Test that the tomiir method works for a list of PZMODEL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output PZMODEL object contains the correct data.	pass
05	Tests that the tomiir method works with a mix of different shaped PZMODEL objects as input.	Test that the tomiir method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output PZMODEL object contains the correct data.	pass
06	Tests that the tomiir method properly applies history.	Test that the result of applying the tomiir method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'tomiir'. 2) Check that re-built object is the same object as the input.	pass
07	Tests that the tomiir method can modify the input PZMODEL object.	Test that the tomiir method can not modify the input PZMODEL object . The method must throw an error for the modifier call.	pass
		1) Nothing to check.	pass

pzmodel/tomiir			
08	Control the method with a plist.	Test that the tomiir method use the different values in a plist.	pass
		1) Check the output 2) Check the re-built object	pass
09	Check that the tomiir method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check indirect the protected 'pzm2ab' method because tomiir uses this method to get the a and b of the iir object.	Create some special pole/zero models to test the 'pzm2ab' method indirect with the tomiir method. Test with complex pole/zero pairs. This UTP define a simple method to get the a-, and b-value from a complex pole/zero pair. It uses the following code: function [a,b] = utp_cpz2ab(pf, pq, zf, zq, fs) wp = pf*2*pi; wp2 = wp$\hat{2}$; wz = zf*2*pi; wz2 = wz$\hat{2}$; k = 4*fs*fs + 2*wp*fs/pq + wp2; a(1) = (4*fs*fs + 2*wz*fs/zq + wz2)/k; a(2) = (2*wz2 - 8*fs*fs)/k; a(3) = (4*fs*fs - 2*wz*fs/zq + wz2)/k; b(1) = 1; b(2) = (2*wp2 - 8*fs*fs)/k; b(3) = (wp2 + 4*fs*fs - 2*wp*fs/pq)/k; g = sum(a) / sum(b); a = a / g; end	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

pzmodel/tomiir			
11	Check indirect the protected 'pzm2ab' method because tomiir uses this method to get the a and b of the iir object.	Create some special pole/zero models to test the 'pzm2ab' method indirect with the tomiir method. Test with complex poles. This UTPs define a simple method to get the a-, and b-value from a complex pole. It uses the following code: function [a,b] = utp_cp2iir(pf, pq, fs) w0 = pf*2*pi; w02 = w0$\hat{2}$; k = (pq*w02 + 4*pq*fs*fs + 2*w0*fs) / (pq*w02); b(1) = 1; b(2) = (2*w02-8*fs*fs) / (k*w02); b(3) = (pq*w02 + 4*pq*fs*fs - 2*w0*fs) / (k*pq*w02); a(1) = 1/k; a(2) = -2/k; a(3) = -1/k; a = a*-2; end	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Check indirect the protected 'pzm2ab' method because tomiir uses this method to get the a and b of the iir object.	Create some special pole/zero models to test the 'pzm2ab' method indirect with the tomiir method. Test with complex zeros. This UTPs define a simple method to get the a-, and b-value from a complex zero. It uses the following code: function [a,b] = utp_cz2iir(zf, zq, fs) w0 = zf*2*pi; w02 = w0$\hat{2}$; a(1) = (-zq*w02/2 - 2*zq*fs*fs - w0*fs) / (zq*w02); a(2) = (-w02+4*fs*fs) / w02; a(3) = (-zq*w02/2 - 2*zq*fs*fs + w0*fs) / (zq*w02); b(1) = 1; b(2) = -2; b(3) = -1; end	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Check indirect the protected 'pzm2ab' method because tomiir uses this method to get the a and b of the iir object.	Create some special pole/zero models to test the 'pzm2ab' method indirect with the tomiir method. Test with real poles. This UTPs define a simple method to get the a-, and b-value from a real pole. It uses the following code: function [a,b] = utp_rp2iir(pf, fs) w0 = pf*2*pi; a(1) = w0 / (2*fs + w0); a(2) = a(1); b(1) = 1; b(2) = (w0-2*fs) / (w0+2*fs); end	pass

pzmodel/tomiir			
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Check indirect the protected 'pzm2ab' method because tomiir uses this method to get the a and b of the iir object.	Create some special pole/zero models to test the 'pzm2ab' method indirect with the tomiir method. Test with real zeros. This UTPs define a simple method to get the a-, and b-value from a real zero. It uses the following code: function [a,b] = utp_rp2iir(zf, fs) w0 = zf*2*pi; a(1) = (2*fs + w0) / w0; a(2) = (-2*fs + w0) / w0; b(1) = 1; b(2) = 1; end	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 306: Unit tests for pzmodel/tomiir.

pzmodel/type			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the type method works with a vector of PZMODEL objects as input.	Test that the type method works for a vector of PZMODEL objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the type method works with a matrix of PZMODEL objects as input.	Test that the type method works for a matrix of PZMODEL objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the type method works with a list of PZMODEL objects as input.	Test that the type method works for a list of PZMODEL objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the type method works with a mix of different shaped PZMODEL objects as input.	Test that the type method works with an input of matrices and vectors and single PZMODEL objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the type method properly applies history.	The method type doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 307: Unit tests for pzmodel/type.

rational/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of RATIONAL objects as input.	Test that the char method works for a vector of RATIONAL objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the char method works with a matrix of RATIONAL objects as input.	Test that the char method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the char method works with a list of RATIONAL objects as input.	Test that the char method works for a list of RATIONAL objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the char method works with a mix of different shaped RATIONAL objects as input.	Test that the char method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the char method properly applies history.	The method char doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 308: Unit tests for rational/char.

rational/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 309: Unit tests for rational/copy.

rational/created			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the created method works with a vector of RATIONAL objects as input.	Test that the created method works for a vector of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'rav' 2) Check that each output contains the correct data.	pass
03	Tests that the created method works with a matrix of RATIONAL objects as input.	Test that the created method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'ram' 2) Check that each output contains the correct data.	pass
04	Tests that the created method works with a list of RATIONAL objects as input.	Test that the created method works for a list of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the created method works with a mix of different shaped RATIONAL objects as input.	Test that the created method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the created method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the created method can be used with the modify command.	Tests that the created method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the created method retruns always a well defined time object even for an empty input object.	Test that the created method with an empty 'RATIONAL object	pass

rational/created			
		1) Check that the output is a time object with a ell defined time.	pass

Table 310: Unit tests for rational/created.

rational/creator			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the creator method works with a vector of RATIONAL objects as input.	Test that the creator method works for a vector of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'rav' 2) Check that each output contains the correct data.	pass
03	Tests that the creator method works with a matrix of RATIONAL objects as input.	Test that the creator method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'ram' 2) Check that each output contains the correct data.	pass
04	Tests that the creator method works with a list of RATIONAL objects as input.	The creator method doesn't work for a list of RATIONAL objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the creator method works with a mix of different shaped RATIONAL objects as input.	The creator method doesn't work with different shaped input objects.	pass
		1) Nothing to test	pass
06	Tests that the creator method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the creator method can be used with the modify command.	Tests that the creator method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the creator method retruns all creator(s)/modifier(s) which are in the history.	Test that the creator method uses the option 'all' direct or in a plist. The test file must have the modifier 'first', 'second' and 'third'	pass
		1) Check that out1 contains only one creator 2) Check that out2 contain more creator/modifier	pass

rational/creator			
09	Tests the negative case for the option 'all'.	Test that the creator method throws an error if the option 'all' is used in connection with a matrix/vector of RATIONAL objects.	pass
		1) Nothing to test.	pass

Table 311: Unit tests for rational/creator.

rational/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of RATIONAL objects as input.	Test that the display method works for a vector of RATIONAL objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of RATIONAL objects as input.	Test that the display method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of RATIONAL objects as input.	Test that the display method works for a list of RATIONAL objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped RATIONAL objects as input.	Test that the display method works with an input of matrices and vectors and single RATIONAL objects as.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the display method properly applies history.	The method display doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 312: Unit tests for rational/display.

rational/eq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eq method works with a vector of RATIONAL objects as input.	Test that the eq method works for a vector of RATIONAL objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
03	Tests that the eq method works with a matrix of RATIONAL objects as input.	Test that the eq method works for a matrix of RATIONAL objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
04	Tests that the eq method works with a list of RATIONAL objects as input.	The eq method doesn't works for a list of RATIONAL objects as input. Nothing to do.	pass
			pass
05	Tests that the eq method works with a mix of different shaped RATIONAL objects as input.	The eq method doesn't works for a list of RATIONAL objects as input. Nothing to do.	pass
			pass
06	Tests that the eq method properly applies history.	The eq method doesn't change the RATIONAL object, thus will no history added. Nothing to do	pass
			pass
07	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because 'pa' is created at an other time.	pass
		1) Check the output.	pass
08	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'iunits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because 'pa' is created at an other time.	pass
		1) Check the output.	pass

rational/eq			
09	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'ounits'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because 'pa' is created at an other time.	pass
		1) Check the output.	pass
10	Test the eq method with an exception list which is in a plist.	Test that the eq method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 313: Unit tests for rational/eq.

rational/get			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the get method of the rational class.	Test that the get returns returns the value of the specified property. Do this for all properties of the RATIONAL object.	pass
		1) Check the correct value of the output	pass
03	Tests that the get method works with a plist.	Test that the get returns returns the value of the specified property which is defined in a plist.	pass
		1) Check the correct value of the output	pass
04	Tests the get method of the rational class.	Test that the get throws an error if the input are more than one RATIONAL object.	pass
		1) Nothing to test	pass

Table 314: Unit tests for rational/get.

rational/getlowerFreq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the getlowerFreq method of the rational class.	Test that the getlowerFreq returns the lowest frequence of the pole in the rational object.	pass
		1) Check the output	pass
03	Tests the getlowerFreq method of the rational class.	Test that the getlowerFreq throws an error if the input are more than one rational.	pass
		1) Nothing to test	pass

Table 315: Unit tests for rational/getlowerFreq.

rational/getupperFreq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the getupperFreq method of the rational class.	Test that the getupperFreq returns the lowest frequence in the rational object.	pass
		1) Check the output	pass
03	Tests the getupperFreq method of the rational class.	Test that the getupperFreq throws an error if the input are more than one rational.	pass
		1) Nothing to test	pass

Table 316: Unit tests for rational/getupperFreq.

rational/index			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the index method works with a vector of RATIONAL objects as input.	Test that the index method works for a vector of RATIONAL objects as input. The following indexing should work: I = [1 2 3] or (I/J) = [(1,1), (1,2), (1,3)]	pass
		1) Check that the index method selects the correct object.	pass
03	Tests that the index method works with a matrix of RATIONAL objects as input.	Test that the index method works for a matrix of RATIONAL objects as input. The following indexing should work: I = [1 3 5] or (I/J) = [(1,1), (1,2), (1,3)] [2 4 6] [(2,1), (2,2), (2,3)]	pass
		1) Check that the index method selects the correct object.	pass
04	Tests that the index method works with a list of RATIONAL objects as input.	The index method doesn't work for a list of RATIONAL objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the index method properly applies history.	Test that the result of index have an additional history step.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'index'.	pass
06	Tests that the index method works for the modifier command.	Tests that the index method works for the modifier command.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
07	Control the method with a plist.	Test that the index method can be controled with a plist.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
08	Test that the index method selects more objects if I have more indices.	Test that the index method selects more objects if I have more indices.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass

rational/index			

Table 317: Unit tests for rational/index.

rational/isprop			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isprop method works with a vector of RATIONAL objects as input.	Test that the isprop method works for a vector of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'rav' 2) Check that each output contains the correct data.	pass
03	Tests that the isprop method works with a matrix of RATIONAL objects as input.	Test that the isprop method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'ram' 2) Check that each output contains the correct data.	pass
04	Tests that the isprop method works with a list of RATIONAL objects as input.	Test that the isprop method works for a list of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the isprop method works with a mix of different shaped RATIONAL objects as input.	Test that the isprop method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the isprop method properly applies history.	The method isprop doesn't change the object, thus it is not necessary to apply history.	pass
			pass
07	Tests that the isprop method works for each property.	Test that the isprop method works for the properties: 'num', 'den', 'iunits', 'ounits', 'hist', 'name'	pass
		1) Check that each output contains the correct data.	pass
08	Test the negative case and the not function command.	Test that the isprop method retrun false for a unknown property and for methods of the object.	pass

rational/isprop			
		1) Check that each output contains the correct data.	pass

Table 318: Unit tests for rational/isprop.

rational/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 319: Unit tests for rational/loadobj.

rational/ne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ne method works with a vector of RATIONAL objects as input.	Test that the ne method works for a vector of RATIONAL objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
03	Tests that the ne method works with a matrix of RATIONAL objects as input.	Test that the ne method works for a matrix of RATIONAL objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
04	Tests that the ne method works with a list of RATIONAL objects as input.	The ne method doesn't works for a list of RATIONAL objects as input. Nothing to do.	pass
			pass
05	Tests that the ne method works with a mix of different shaped RATIONAL objects as input.	The ne method doesn't works for a list of RATIONAL objects as input. Nothing to do.	pass
			pass
06	Tests that the ne method properly applies history.	The ne method doesn't change the RATIONAL object, thus will no history added. Nothing to do	pass
			pass
07	Test the ne method with an exception list. The function rational/ne use the function rational/eq so it is not necessary to check all possibilities of the exception list.	Test the ne method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because ra is created at an other time.	pass
		1) Check that each output contains the correct data.	pass
08	Test the ne method with an exception list which is in a plist.	Test that the ne method uses the exception list in a plist.	pass
		1) Check that each output contains the correct data.	pass

Table 320: Unit tests for rational/ne.

rational/rational			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rational method works with a vector of RATIONAL objects as input.	Test that the rational method works with a vector of RATIONAL objects as input.	pass
		1) Check that the shape of the output RATIONALs is the same as the input shape. 2) Check that each output RATIONAL is a copy of the input RATIONAL. 3) Check that the copy have an additional history step.	pass
03	Tests that the rational method works with a matrix of RATIONAL objects as input.	Test that the rational method works with a matrix of RATIONAL objects as input.	pass
		1) Check that the shape of the output RATIONALs is the same as the input shape. 2) Check that each output RATIONAL is a copy of the input RATIONAL. 3) Check that the copy have an additional history step.	pass
04	Tests that the rational method works with a list of RATIONAL objects as input.	Test that the rational method works with a list of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output RATIONAL is a copy of the input RATIONAL. 3) Check that the copy have an additional history step.	pass
05	Tests that the rational method works with a mix of different shaped RATIONALs as input.	Test that the rational method works with a mix of different shaped RATIONALs as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output RATIONAL is a copy of the input RATIONAL. 3) Check that the copy have an additional history step.	pass
06	Tests that the rational method properly applies history.	Test that the result of applying the rational method can be processed back.	pass

rational/rational			
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the rational method properly applies history to the copy constructor.	Test that the output can be processed back with the 'rebuild' method. Test the constructor with a different number of inputs.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the original objects are not changed by the setter function 3) Check that the method rebuild produces the same object as 'out'.	pass
08	Tests that the rational method properly applies history to the read MAT-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Tests that the rational method properly applies history to the read XML-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the rational method properly doesn't apply history to the struct constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Tests that the rational method properly applies history to the parfrac constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Tests that the rational method properly applies history to the pzmodel constructor.	Test that the output can be processed back with the 'rebuild' method.	pass

rational/rational			
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Tests that the rational method properly applies history to the plist(filename) constructor.	Test that the output can be processed back to an m-file.	pass
		1) Check that the save method doesn't change the input object 2) Check that the last two entries in the history of 'out' doesn't corresponds to 'rational' and 'save' 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
14	Tests that the RATIONAL method properly applies history to the plist(conn) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Tests that the RATIONAL method properly applies history to the plist(den || num) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
16	Tests that the RATIONAL method properly applies history to the plist(pzmodel) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
17	Tests that the RATIONAL method properly applies history to the plist(rational) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
18	Tests that the RATIONAL method properly applies history to the plist(<plist-object>) constructor.	Test that the output can be processed back with the rebuild method.	pass

rational/rational			
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
19	Tests that the RATIONAL method properly applies history to the conn+Id constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
20	Tests that the RATIONAL method properly applies history to the num + den object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
21	Tests that the RATIONAL method properly applies history to the num + den + name object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
22	Tests that the RATIONAL method properly applies history to the num + den + name + iunits + ounits object constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'rational'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 321: Unit tests for rational/rational.

rational/rebuild			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rebuild method works with a vector of RATIONAL objects as input.	Test that the rebuild method works for a vector of RATIONAL objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the rebuild method works with a matrix of RATIONAL objects as input.	Test that the rebuild method works for a matrix of RATIONAL objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the rebuild method works with a list of RATIONAL objects as input.	Test that the rebuild method works for a list of RATIONAL objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the rebuild method works with a mix of different shaped RATIONAL objects as input.	Test that the rebuild method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the rebuild method properly applies history.	The method rebuild doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Check that the rebuild method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 322: Unit tests for rational/rebuild.

rational/resp			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the resp method works with a vector of RATIONAL objects as input.	Test that the resp method works for a vector of RATIONAL objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'rav' 3) Check that each output RATIONAL contains the correct data.	pass
03	Tests that the resp method works with a matrix of RATIONAL objects as input.	Tests that the resp method works with a matrix of RATIONAL objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'ram' 3) Check that each output RATIONAL contains the correct data.	pass
04	Tests that the resp method works with a list of RATIONAL objects as input.	Tests that the resp method works with a list of RATIONAL objects as input. Test the method with an output and with no output (a diagram must appear)	pass
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'rain' 3) Check that each output RATIONAL contains the correct data.	pass
05	Tests that the resp method works with a mix of different shaped RATIONAL objects as input.	Tests that the resp method works with a mix of different shaped RATIONAL objects as input. Test the method with an output and with no output (a diagram must appear)	pass

rational/resp			
		1) Test the right number of lines in the diagram. 2) Check that the number of elements in 'out' is the same as in 'rain' 3) Check that each output RATIONAL contains the correct data.	pass
06	Tests that the resp method properly applies history.	Test that the result of applying the resp method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'resp'. 2) Check that re-built object is the same object as the input.	pass
07	Tests that modify command plots the response into a diagram.	Tests that modify command plots the response into a diagram.	pass
		1) Check the response diagram.	pass
08	Test the shape of the output.	Test that the output AO of the resp method keeps the shape of the used input f vector.	pass
		1) Check that the shape of the data doesn't change.	pass
09	Check that the resp method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the resp method uses the x-data of an input AO for f-vector.	Call the method with different method to pass an AO in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Check that the resp method uses the specified f-vector to compute the response.	Call the method with different method to pass an f-vector in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Check that the resp method uses the specified f1, f2, and nf to compute the response.	Call the method with different method to pass f1, f2, and nf in.	pass

rational/resp			
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 323: Unit tests for rational/resp.

rational/save			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the save method works with a vector of RATIONAL objects as input.	Test that the save method works for a vector of RATIONAL objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'rav' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
03	Tests that the save method works with a matrix of RATIONAL objects as input.	Test that the save method works for a matrix of RATIONAL objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'ram' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
04	Tests that the save method works with a list of RATIONAL objects as input.	Test that the save method works for a list of RATIONAL objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in the list 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
05	Tests that the save method works with a mix of different shaped RATIONAL objects as input.	Test that the save method works with an input of matrices and vectors and single RATIONAL objects. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output RATIONAL object contains the correct data.	pass

rational/save			
06	Tests that the save method properly applies history.	Test that the result of applying the save method can be processed back to an m-file. Do this for both extensions 'mat' and 'xml'	pass
		1) Check that the history applies to the output object. Check that save doesn't add a history step to the input object. 2) Check that the read object doesn't contain the save + load history steps. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the save method works with the modify command.	Use the save method with the modifier command.	pass
		1) Check that the save method doesn't apply the history. 2) Check the output against the input.	pass
08	Control the method with a plist.	Test that the save method uses the filename which is stored in a plist.	pass
		1) Check the output	pass
09	Test the save method with different complex RATIONAL objects	Test the save method with different complex RATIONAL objects	pass
		1) Check the output	pass

Table 324: Unit tests for rational/save.

rational/setIunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setIunits method works with a vector of RATIONAL objects as input.	Test that the setIunits method works for a vector of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'rav' 2) Check that each output contains the correct data.	pass
03	Tests that the setIunits method works with a matrix of RATIONAL objects as input.	Test that the setIunits method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'ram' 2) Check that each output contains the correct data.	pass
04	Tests that the setIunits method works with a list of RATIONAL objects as input.	Test that the setIunits method works for a list of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setIunits method works with a mix of different shaped RATIONAL objects as input.	Test that the setIunits method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setIunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setIunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setIunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setIunits'. 3) Check that the method rebuild produces the same object as 'out'.	pass

rational/setIunits			
07	Tests that the setIunits method can modify the input RATIONAL object.	Test that the setIunits method can modify the input RATIONAL object by calling with no output.	pass
		1) Check that 'ra3' and 'ain' are now different. 2) Check that 'ain' has the correct iunits field	pass
08	Tests that the setIunits method can set the property with a plist.	Test that the setIunits method can modify the property 'iunits' with a value in a plist.	pass
		1) Check that 'ain' has the correct iunits field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setIunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 325: Unit tests for rational/setIunits.

rational/setName			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setName method works with a vector of RATIONAL objects as input.	Test that the setName method works for a vector of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'rav' 2) Check that each output contains the correct data.	pass
03	Tests that the setName method works with a matrix of RATIONAL objects as input.	Test that the setName method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'ram' 2) Check that each output contains the correct data.	pass
04	Tests that the setName method works with a list of RATIONAL objects as input.	Test that the setName method works for a list of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setName method works with a mix of different shaped RATIONAL objects as input.	Test that the setName method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setName method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setName method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setName'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setName'. 3) Check that the method rebuild produces the same object as 'out'.	pass

rational/setName			
07	Tests that the setName method can modify the input RATIONAL object.	Test that the setName method can modify the input RATIONAL object by calling with no output.	pass
		1) Check that 'ra3' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
08	Tests that the setName method can set the property with a plist.	Test that the setName method can modify the property 'name' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setName method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 326: Unit tests for rational/setName.

rational/setOunits			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setOunits method works with a vector of RATIONAL objects as input.	Test that the setOunits method works for a vector of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'rav' 2) Check that each output contains the correct data.	pass
03	Tests that the setOunits method works with a matrix of RATIONAL objects as input.	Test that the setOunits method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'ram' 2) Check that each output contains the correct data.	pass
04	Tests that the setOunits method works with a list of RATIONAL objects as input.	Test that the setOunits method works for a list of RATIONAL objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setOunits method works with a mix of different shaped RATIONAL objects as input.	Test that the setOunits method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setOunits method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setOunits method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setOunits'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setOunits'. 3) Check that the method rebuild produces the same object as 'out'.	pass

rational/setOunits			
07	Tests that the setOunits method can modify the input RATIONAL object.	Test that the setOunits method can modify the input RATIONAL object by calling with no output.	pass
		1) Check that 'ra3' and 'ain' are now different. 2) Check that 'ain' has the correct ounits field	pass
08	Tests that the setOunits method can set the property with a plist.	Test that the setOunits method can modify the property 'ounits' with a value in a plist.	pass
		1) Check that 'ain' has the correct ounits field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setOunits method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 327: Unit tests for rational/setOunits.

rational/string			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the string method works with a vector of RATIONAL objects as input.	Test that the string method works for a vector of RATIONAL objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
03	Tests that the string method works with a matrix of RATIONAL objects as input.	Test that the string method works for a matrix of RATIONAL objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
04	Tests that the string method works with a list of RATIONAL objects as input.	Test that the string method works for a list of RATIONAL objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
05	Tests that the string method works with a mix of different shaped RATIONAL objects as input.	Test that the string method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
06	Tests that the string method properly applies history.	The method string doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Tests that the string method doesn't work if the RATIONAL object have more than one history step.	The method string throws an error because the input object have more than one history step.	pass
			pass

Table 328: Unit tests for rational/string.

rational/type			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the type method works with a vector of RATIONAL objects as input.	Test that the type method works for a vector of RATIONAL objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the type method works with a matrix of RATIONAL objects as input.	Test that the type method works for a matrix of RATIONAL objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the type method works with a list of RATIONAL objects as input.	Test that the type method works for a list of RATIONAL objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the type method works with a mix of different shaped RATIONAL objects as input.	Test that the type method works with an input of matrices and vectors and single RATIONAL objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the type method properly applies history.	The method type doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 329: Unit tests for rational/type.

smodel/addAliases			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/addAliases] method works for a list of objects as input.	Tests that the [smodel/addAliases] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/addAliases] method properly applies history.	Test that the result of applying the [smodel/addAliases] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/addAliases]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/addAliases] method can modify the input AO.	Test that the [smodel/addAliases] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass

smodel/addAliases			
genericOutput	Check that the [smodel/addAliases] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 330: Unit tests for smodel/addAliases.

smodel/addParameters			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape	Tests that the [smodel/addParameters] method works with any shape of objects as input.	Test that the [smodel/addParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/addParameters] method works with any shape of objects as input.	Test that the [smodel/addParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/addParameters] method works with any shape of objects as input.	Test that the [smodel/addParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/addParameters] method works with any shape of objects as input.	Test that the [smodel/addParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/addParameters] method works with any shape of objects as input.	Test that the [smodel/addParameters] method works for any shape of objects as input.	pass

smodel/addParameters			
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/addParameters] method works with any shape of objects as input.	Test that the [smodel/addParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/addParameters] method works with any shape of objects as input.	Test that the [smodel/addParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/addParameters] method works with any shape of objects as input.	Test that the [smodel/addParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericModify	Tests that the [smodel/addParameters] method can modify the input AO.	Test that the [smodel/addParameters] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [smodel/addParameters] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass

smodel/addParameters			
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/addParameters] method works for a list of objects as input.	Tests that the [smodel/addParameters] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/addParameters] method properly applies history.	Test that the result of applying the [smodel/addParameters] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/addParameters]'. 2) Check that the re-built object is the same object as the input.	pass

Table 331: Unit tests for smodel/addParameters.

smodel/clearAliases			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape	Tests that the [smodel/clearAliases] method works with any shape of objects as input.	Test that the [smodel/clearAliases] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/clearAliases] method works for a list of objects as input.	Tests that the [smodel/clearAliases] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/clearAliases] method properly applies history.	Test that the result of applying the [smodel/clearAliases] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/clearAliases]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/clearAliases] method can modify the input AO.	Test that the [smodel/clearAliases] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass

smodel/clearAliases			
genericOutput	Check that the [smodel/clearAliases] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 332: Unit tests for smodel/clearAliases.

smodel/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 333: Unit tests for smodel/copy.

smodel/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 334: Unit tests for smodel/loadobj.

smodel/setAliases			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setAliases] method works for a list of objects as input.	Tests that the [smodel/setAliases] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setAliases] method properly applies history.	Test that the result of applying the [smodel/setAliases] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setAliases]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/setAliases] method can modify the input AO.	Test that the [smodel/setAliases] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass

smodel/setAliases			
genericOutput	Check that the [smodel/setAliases] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 335: Unit tests for smodel/setAliases.

smodel/setParameters			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape	Tests that the [smodel/setParameters] method works with any shape of objects as input.	Test that the [smodel/setParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParameters] method works with any shape of objects as input.	Test that the [smodel/setParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParameters] method works with any shape of objects as input.	Test that the [smodel/setParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParameters] method works with any shape of objects as input.	Test that the [smodel/setParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParameters] method works with any shape of objects as input.	Test that the [smodel/setParameters] method works for any shape of objects as input.	pass

smodel/setParameters			
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParameters] method works with any shape of objects as input.	Test that the [smodel/setParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParameters] method works with any shape of objects as input.	Test that the [smodel/setParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParameters] method works with any shape of objects as input.	Test that the [smodel/setParameters] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericModify	Tests that the [smodel/setParameters] method can modify the input AO.	Test that the [smodel/setParameters] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [smodel/setParameters] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass

smodel/setParameters			
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setParameters] method works for a list of objects as input.	Tests that the [smodel/setParameters] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setParameters] method properly applies history.	Test that the result of applying the [smodel/setParameters] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setParameters]'. 2) Check that the re-built object is the same object as the input.	pass

Table 336: Unit tests for smodel/setParameters.

smodel/setParams			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass

smodel/setParams			
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape	Tests that the [smodel/setParams] method works with any shape of objects as input.	Test that the [smodel/setParams] method works for any shape of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data. 3) Rebuild the object	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setParams] method works for a list of objects as input.	Tests that the [smodel/setParams] method works for a list of objects as input.	pass

smodel/setParams			
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setParams] method properly applies history.	Test that the result of applying the [smodel/setParams] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setParams]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/setParams] method can modify the input AO.	Test that the [smodel/setParams] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [smodel/setParams] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 337: Unit tests for smodel/setParams.

smodel/setTrans			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setTrans] method works for a list of objects as input.	Tests that the [smodel/setTrans] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [smodel/setTrans] method works for a list of objects as input.	Tests that the [smodel/setTrans] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setTrans] method properly applies history.	Test that the result of applying the [smodel/setTrans] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setTrans]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/setTrans] method can modify the input AO.	Test that the [smodel/setTrans] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

smodel/setTrans			
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [smodel/setTrans] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 338: Unit tests for smodel/setTrans.

smodel/setValues			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setValues] method works for a list of objects as input.	Tests that the [smodel/setValues] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [smodel/setValues] method works for a list of objects as input.	Tests that the [smodel/setValues] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setValues] method properly applies history.	Test that the result of applying the [smodel/setValues] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setValues]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/setValues] method can modify the input AO.	Test that the [smodel/setValues] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass

smodel/setValues			
genericOutput	Check that the [smodel/setValues] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 339: Unit tests for smodel/setValues.

smodel/setXunits			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setXunits] method works for a list of objects as input.	Tests that the [smodel/setXunits] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [smodel/setXunits] method works for a list of objects as input.	Tests that the [smodel/setXunits] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setXunits] method properly applies history.	Test that the result of applying the [smodel/setXunits] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setXunits]'. 2) Check that the re-built object is the same object as the input.	pass

smodel/setXunits			
genericModify	Tests that the [smodel/setXunits] method can modify the input AO.	Test that the [smodel/setXunits] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [smodel/setXunits] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 340: Unit tests for smodel/setXunits.

smodel/setXvals			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setXvals] method works for a list of objects as input.	Tests that the [smodel/setXvals] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [smodel/setXvals] method works for a list of objects as input.	Tests that the [smodel/setXvals] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setXvals] method properly applies history.	Test that the result of applying the [smodel/setXvals] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setXvals]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/setXvals] method can modify the input AO.	Test that the [smodel/setXvals] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

smodel/setXvals			
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [smodel/setXvals] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 341: Unit tests for smodel/setXvals.

smodel/setXvar			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setXvar] method works for a list of objects as input.	Tests that the [smodel/setXvar] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [smodel/setXvar] method works for a list of objects as input.	Tests that the [smodel/setXvar] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setXvar] method properly applies history.	Test that the result of applying the [smodel/setXvar] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setXvar]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/setXvar] method can modify the input AO.	Test that the [smodel/setXvar] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

smodel/setXvar			
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [smodel/setXvar] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 342: Unit tests for smodel/setXvar.

smodel/setYunits			
minfo	Tests that the getInfo call works for this a general setter method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericAnyShape			pass
			pass
genericList	Tests that the [smodel/setYunits] method works for a list of objects as input.	Tests that the [smodel/setYunits] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericList	Tests that the [smodel/setYunits] method works for a list of objects as input.	Tests that the [smodel/setYunits] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'objs' 2) Check that each output object contains the correct data.	pass
genericHistory	Tests that the [smodel/setYunits] method properly applies history.	Test that the result of applying the [smodel/setYunits] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/setYunits]'. 2) Check that the re-built object is the same object as the input.	pass
genericModify	Tests that the [smodel/setYunits] method can modify the input AO.	Test that the [smodel/setYunits] method can modify the input object by calling with no output and that the method doesn't change the input of the function notation (with a equal sign).	pass

smodel/setYunits			
		1) Check that the modified input is changed by the method 2) Check that 'out' and 'obj_eq' are now different. 3) Check that 'obj_eq' is not changed 4) Check that out and amodi are the same	pass
genericOutput	Check that the [smodel/setYunits] method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 343: Unit tests for smodel/setYunits.

smodel/smodel			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [smodel/smodel] method works with a vector of objects as input.	Test that the [smodel/smodel] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [smodel/smodel] method works with a matrix of objects as input.	Test that the [smodel/smodel] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [smodel/smodel] method works with a list of objects as input.	Test that the [smodel/smodel] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [smodel/smodel] method works with a mix of different arrays of objects as input.	Tests that the [smodel/smodel] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [smodel/smodel] method properly applies history.	Test that the result of applying the [smodel/smodel] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[smodel/smodel]'. 2) Check that the re-built object is the same object as the input.	pass

smodel/smodel			
07	Tests that the smodel can read older MAT files which have different values in 'xvals'	Tests that the smodel can read older MAT files which have different values in 'xvals' The smodel object was created with the commands: s = smodel('A.*(t >= toff)'); s.setXvar('t'); s.setParams('A','toff', 5,300); s.setName('Step at toff'); s.setDescription('Step function of amplitude A at time toff'); s.setXunits('Hz1̂/2 km2̂'); s.setYunits('s2̂ m-̂1'); s.setXvals(1:1e3); s.setAliasNames('a'); s.setAliasValues(8);	pass
		1) Check that the objects are quite the same 2) Check that 'xvals' is a cell-array with the numbers from 1 to 1000	pass
60	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
61	Tests that the constructor properly applies history to the read XML-file constructor.	Tests that the constructor properly applies history to the read XML-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
62	Tests that the constructor properly applies history in the struct constructor.	Tests that the constructor properly applies history in the struct constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
64	Tests that the contructor properly applies history to the plist(filename) constructor.	Tests that the contructor properly applies history to the plist(filename) constructor.	pass

smodel/smodel			
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
65	Tests that the contructed object can be submitted and retrieved.	Tests that the contructed object can be submitted and retrieved.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
68	Tests that the constructor properly applies history to the conn+Id constructor.	Tests that the constructor properly applies history to the conn+Id constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 344: Unit tests for smodel/smodel.

ssm/addParameters			
02	Tests that the [ssm/addParameters] method works with a vector of objects as input.	Test that the [ssm/addParameters] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ssm/addParameters] method works with a matrix of objects as input.	Test that the [ssm/addParameters] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ssm/addParameters] method works with a list of objects as input.	Test that the [ssm/addParameters] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ssm/addParameters] method works with a list of objects as input.	Test that the [ssm/addParameters] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ssm/addParameters] method works with a mix of different arrays of objects as input.	Tests that the [ssm/addParameters] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ssm/addParameters] method properly applies history.	Test that the result of applying the [ssm/addParameters] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ssm/addParameters]'. 2) Check that the re-built object is the same object as the input.	pass

Table 345: Unit tests for ssm/addParameters.

ssm/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 346: Unit tests for ssm/copy.

ssm/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 347: Unit tests for ssm/loadobj.

model/DFACS			
901	Tests that the model <MODEL> has a meaningful name.	Test that the model <MODEL> has a meaningful name.	pass
		1) Check that the name of the object is not empty and not equal to 'none'	pass
902	Tests that the model <MODEL> has a meaningful description.	Test that the model <MODEL> has a meaningful description.	pass
		1) Check that the description of the object is not empty and not equal to 'none'	pass
903	Tests that the model <MODEL> responds to 'DIM' configuration key.	Test that the model <MODEL> responds to 'DIM' configuration key.	pass
		1) Check that the model builds with DIM==1 2) Check that the model builds with DIM==2 3) Check that the model builds with DIM==3 4) Check that the 3 models are different	pass

Table 348: Unit tests for model/DFACS.

model/IFO			
901	Tests that the model <MODEL> has a meaningful name.	Test that the model <MODEL> has a meaningful name.	pass
		1) Check that the name of the object is not empty and not equal to 'none'	pass
902	Tests that the model <MODEL> has a meaningful description.	Test that the model <MODEL> has a meaningful description.	pass
		1) Check that the description of the object is not empty and not equal to 'none'	pass
903	Tests that the model <MODEL> responds to 'DIM' configuration key.	Test that the model <MODEL> responds to 'DIM' configuration key.	pass
		1) Check that the model builds with DIM==1 2) Check that the model builds with DIM==2 3) Check that the model builds with DIM==3 4) Check that the 3 models are different	pass

Table 349: Unit tests for model/IFO.

ssm/ssm			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the [ssm/ssm] method works with a vector of objects as input.	Test that the [ssm/ssm] method works for a vector of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'vec' 2) Check that each output object contains the correct data.	pass
03	Tests that the [ssm/ssm] method works with a matrix of objects as input.	Test that the [ssm/ssm] method works for a matrix of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
04	Tests that the [ssm/ssm] method works with a list of objects as input.	Test that the [ssm/ssm] method works for a list of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
05	Tests that the [ssm/ssm] method works with a mix of different arrays of objects as input.	Tests that the [ssm/ssm] method works with a mix of different arrays of objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'mat' 2) Check that each output object contains the correct data.	pass
06	Tests that the [ssm/ssm] method properly applies history.	Test that the result of applying the [ssm/ssm] method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to '[ssm/ssm]'. 2) Check that the re-built object is the same object as the input.	pass
60	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	Tests that the constructor method doesn't apply history to the read MAT-file constructor.	pass

ssm/ssm			
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
61	Tests that the constructor properly applies history to the read XML-file constructor.	Tests that the constructor properly applies history to the read XML-file constructor.	pass
		1) Check that the history is the same as the history of the saved object. Because save and load shouldn't add a history step. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
62	Tests that the constructor properly applies history in the struct constructor.	Tests that the constructor properly applies history in the struct constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
63	Tests that the constructor properly applies history to the pzmodel constructor.	Tests that the constructor properly applies history to the pzmodel constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
64	Tests that the contructor properly applies history to the plist(filename) constructor.	Tests that the contructor properly applies history to the plist(filename) constructor.	pass
		1) Check that the save method doesn't change the input object 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
65	Tests that the contructed object can be submitted and retrieved.	Tests that the contructed object can be submitted and retrieved.	pass
		1) Check that the last entry in the history of 'out' corresponds to the class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
66	Tests that the construuctor properly works with the plist(pzmodel) constructor.	Tests that the constructor properly works with the plist(pzmodel) constructor.	pass

ssm/ssm			
		1) Check that the last entry in the history of 'out' corresponds to 'ssm'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
67	Tests that the constructor properly applies history to the pole/zero model + plist object constructor.	Tests that the constructor properly applies history to the pole/zero model + plist object constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
68	Tests that the constructor properly applies history to the conn+Id constructor.	Tests that the constructor properly applies history to the conn+Id constructor.	pass
		1) Check that the last entry in the history of 'out' corresponds to class name. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 350: Unit tests for ssm/ssm.

time/datenum			
901	Tests the 'datenum' method.	Create a time object and use datenum to produce a serial date number.	pass
		Check that the returned value is the same as the one you get when using time/format with the same format string when converted using MATLAB's datestr() function.	pass
901	Tests the 'datenum' method.	Create a time object and use datenum to produce a serial date number.	pass
		Check that the returned value is the same as the one you get when using time/format with the same format string when converted using MATLAB's datestr() function.	pass
903	Tests 'double' method.	Set the time-zone to PST and create a time object and use datenum to produce a serial date number.	pass
		Check that the returned value is the same as the one you get when using time/format with the same format string when converted using MATLAB's datestr() function.	pass

Table 351: Unit tests for time/datenum.

time/double			
901	Tests 'double' method.	Use the double() method on single time objects and vectors of time objects.	pass
		Check that double returns the expected numerical values for each case.	pass

Table 352: Unit tests for time/double.

time/format			
901	Tests 'format' static method.	Use the static format method to produce a number of time strings in different formats and for different time-zones.	pass
		Check that the returned strings match the expected strings.	pass

Table 353: Unit tests for time/format.

time/minus			
01	Tests time object minus operator.	Compute the difference between time objects and doubles.	pass
		Check the resulting time objects have the correct values.	pass

Table 354: Unit tests for time/minus.

time/parse			
901	Tests 'parse' static method with time string only.	Use parse() to parse different time strings with different time-zones set.	pass
		Check the resulting millsecond values are the expected ones.	pass
902	Tests 'parse' static method with time and format strings.	Use parse() to parse different time strings with different time-zones set. The time-string format is specified as a second input argument.	pass
		Check the resulting millsecond values are the expected ones.	pass
903	Tests 'parse' static method with time string and numeric format.	Use parse() to parse different time strings using the supported MATLAB numeric time formats.	pass
		Check the resulting millsecond values against the result of parsing a time string. (See time/parse utp_902.)	pass
904	Tests 'parse' static method with time string containing timezone information.	Use parse() to parse different time strings which contain the time-zone.	pass
		Check the resulting millisecond value is the expected one.	pass
905	Tests 'parse' static method with time string and timezone specification.	Use parse() to parse different time strings when passing the time-zone as the third argument. The second argument (the format) is left empty.	pass
		Check the resulting millisecond value is the expected one.	pass

Table 355: Unit tests for time/parse.

time/plus			
101	Tests time object plus operator.	Compute the sum of time objects and doubles.	pass
		Check the resulting time objects have the correct values.	pass

Table 356: Unit tests for time/plus.

time/string			
901	Tests that the output of the 'string' method can be used to recreate the time object.	Use string to convert a time object to a string. Use eval on the result to recreate a time object.	pass
		Check the recreated time object matches the original.	pass

Table 357: Unit tests for time/string.

time/time			
901	Tests time object constructor without arguments. Should return the current time.	Call the time() constructor with no inputs.	pass
		No test can be done here since we don't know the time when the constructor is called. We could check that the resulting time is > 0, but that's not so useful.	pass
902	Tests time object constructor from numeric value.	Call the time() constructor with a numeric input (number of seconds since epoch).	pass
		Check the time object has the expected millisecond value.	pass
903	Tests time object constructor from string.	Call the time() constructor with different time-string inputs.	pass
		Check the time objects have the expected millisecond values.	pass
904	Tests time object constructor from a cell array of strings.	Call the time() constructor with cell-array of time-strings.	pass
		Check the time objects have the expected millisecond values.	pass
905	Tests time object constructor from plist with 'time' parameter.	Call the time() constructor with plist input. Check a plist with a numeric value and a plist with a time-string.	pass
		Check the time objects have the expected millisecond values.	pass
906	Tests time object constructor from plist with 'milliseconds' parameter	Call the time() constructor with plist input using the 'millisecond' key.	pass
		Check the time object has the expected millisecond value.	pass
907	Tests time object constructor from plist with more parameters.	Call the time() constructor with plist input containing the time string and the time-zone. key.	pass
		Check the time objects have the expected millisecond values.	pass
908	Tests time object constructor from structure.	Call the time() constructor with an input structure obtained from calling struct() on a time object. key.	pass
		Check the original and reconstructed time objects have the same millisecond value.	pass
909	Tests time object constructor from string and time format.	Call the time() constructor with an input time string and time format string.	pass
		Check the resulting time object has the expected millisecond value.	pass

time/time			

Table 358: Unit tests for time/time.

time/timeformat			
901	Tests that the method really returns what is in the user preferences.	Check that the timeformat method runs without error.	pass
		Check that the returned value is the same as the one set in the user preferences.	pass
902	Change user preferences and do it again.	Set the time string format in the preferences and check that the timeformat method runs without error.	pass
		Check that the returned value is the same as the one set in the user preferences.	pass
903	Change user preferences and do it again.	Set the time string format in the preferences and check that the timeformat method runs without error.	pass
		Check that the returned value is the same as the one set in the user preferences.	pass

Table 359: Unit tests for time/timeformat.

time/timezone			
901	Tests that the method really returns what is in the user preferences.	Call the static timezone method.	pass
		Check the resulting timezone object against the one in the user preferences.	pass
902	Change user preferences and do it again.	Set the timezone in the preferences then call the static timezone method.	pass
		Check the resulting timezone object against the one in the user preferences.	pass
903	Change user preferences and do it again.	Set the timezone in the preferences then call the static timezone method.	pass
		Check the resulting timezone object against the one in the user preferences.	pass

Table 360: Unit tests for time/timezone.

timespan/char			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the char method works with a vector of TIMESPAN objects as input.	Test that the char method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the output contain at least each start time	pass
03	Tests that the char method works with a matrix of TIMESPAN objects as input.	Test that the char method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the output contain at least each start time	pass
04	Tests that the char method works with a list of TIMESPAN objects as input.	Test that the char method works for a list of TIMESPAN objects as input.	pass
		1) Check that the output contain at least each start time	pass
05	Tests that the char method works with a mix of different shaped TIMESPAN objects as input.	Test that the char method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check that the output contain at least each start time	pass
06	Tests that the char method properly applies history.	The method char doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 361: Unit tests for timespan/char.

timespan/copy			
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass
69	Tests that the copy method works with a single object as an input.	Test the positive (copy-case) and the negative (non copy-case) case.	pass
		1) Check that the output is a 'real' copy or only a copy of the handle	pass

Table 362: Unit tests for timespan/copy.

timespan/created			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the created method works with a vector of TIMESPAN objects as input.	Test that the created method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsv' 2) Check that each output contains the correct data.	pass
03	Tests that the created method works with a matrix of TIMESPAN objects as input.	Test that the created method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsm' 2) Check that each output contains the correct data.	pass
04	Tests that the created method works with a list of TIMESPAN objects as input.	Test that the created method works for a list of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the created method works with a mix of different shaped TIMESPAN objects as input.	Test that the created method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the created method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the created method can be used with the modify command.	Tests that the created method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the created method retruns always a well defined time object even for an empty input object.	Test that the created method with an empty 'TIMESPAN object	pass

timespan/created			
		1) Check that the output is a time object with a ell defined time.	pass

Table 363: Unit tests for timespan/created.

timespan/creator			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the creator method works with a vector of TIMESPAN objects as input.	Test that the creator method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsv' 2) Check that each output contains the correct data.	pass
03	Tests that the creator method works with a matrix of TIMESPAN objects as input.	Test that the creator method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsm' 2) Check that each output contains the correct data.	pass
04	Tests that the creator method works with a list of TIMESPAN objects as input.	The creator method doesn't work for a list of TIMESPAN objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the creator method works with a mix of different shaped TIMESPAN objects as input.	The creator method doesn't work with different shaped input objects.	pass
		1) Nothing to test	pass
06	Tests that the creator method properly applies history	This method doesn't change the input object, thus no history is added to the object.	pass
		1) Nothing to check.	pass
07	Tests that the creator method can be used with the modify command.	Tests that the creator method can be used with the modify command.	pass
		1) Check the single object 2) Check the matrix object	pass
08	Tests that the creator method retruns all creator(s)/modifier(s) which are in the history.	Test that the creator method uses the option 'all' direct or in a plist. The test file must have the modifier 'first', 'second' and 'third'	pass
		1) Check that out1 contains only one creator 2) Check that out2 contain more creator/modifier	pass

timespan/creator			
09	Tests the negative case for the option 'all'.	Test that the creator method throws an error if the option 'all' is used in connection with a matrix/vector of TIMESPAN objects.	pass
		1) Nothing to test.	pass

Table 364: Unit tests for timespan/creator.

timespan/display			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the display method works with a vector of TIMESPAN objects as input.	Test that the display method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the output contain at least each object name	pass
03	Tests that the display method works with a matrix of TIMESPAN objects as input.	Test that the display method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the output contain at least each object name	pass
04	Tests that the display method works with a list of TIMESPAN objects as input.	Test that the display method works for a list of TIMESPAN objects as input.	pass
		1) Check that the output contain at least each object name	pass
05	Tests that the display method works with a mix of different shaped TIMESPAN objects as input.	Test that the display method works with an input of matrices and vectors and single TIMESPAN objects as.	pass
		1) Check that the output contain at least each object name	pass
06	Tests that the display method properly applies history.	The method display doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 365: Unit tests for timespan/display.

timespan/eq			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the eq method works with a vector of TIMESPAN objects as input.	Test that the eq method works for a vector of TIMESPAN objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
03	Tests that the eq method works with a matrix of TIMESPAN objects as input.	Test that the eq method works for a matrix of TIMESPAN objects as input. Test the positive and the negative case.	pass
		1) Check the output of the eq function.	pass
04	Tests that the eq method works with a list of TIMESPAN objects as input.	The eq method doesn't works for a list of TIMESPAN objects as input. Nothing to do.	pass
			pass
05	Tests that the eq method works with a mix of different shaped TIMESPAN objects as input.	The eq method doesn't works for a list of TIMESPAN objects as input. Nothing to do.	pass
			pass
06	Tests that the eq method properly applies history.	The eq method doesn't change the TIMESPAN object, thus will no history added. Nothing to do	pass
			pass
07	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because ts is created at an other time.	pass
		1) Check the output.	pass
08	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'endt'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because ts is created at an other time.	pass
		1) Check the output.	pass

timespan/eq			
09	Test the eq method with an exception list. With the LTPDA toolbox 2.0 it is only possible to test the exception list with properties where a public set method exist.	Test the eq method with the exception 'startT'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because ts is created at an other time.	pass
		1) Check the output.	pass
12	Test the eq method with an exception list which is in a plist.	Test that the eq method uses the exception list in a plist.	pass
		1) Check the output.	pass

Table 366: Unit tests for timespan/eq.

timespan/get			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests the get method of the timespan class.	Test that the get returns returns the value of the specified property. Do this for all properties of the TIMESPAN object.	pass
		1) Check the correct value of the output	pass
03	Tests that the get method works with a plist.	Test that the get returns returns the value of the specified property which is defined in a plist.	pass
		1) Check the correct value of the output	pass
04	Tests the get method of the timespan class.	Test that the get throws an error if the input are more than one TIMESPAN object.	pass
		1) Nothing to test	pass

Table 367: Unit tests for timespan/get.

timespan/index			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the index method works with a vector of TIMESPAN objects as input.	Test that the index method works for a vector of TIMESPAN objects as input. The following indexing should work: I = [1 2 3] or (I/J) = [(1,1), (1,2), (1,3)]	pass
		1) Check that the index method selects the correct object.	pass
03	Tests that the index method works with a matrix of TIMESPAN objects as input.	Test that the index method works for a matrix of TIMESPAN objects as input. The following indexing should work: I = [1 3 5] or (I/J) = [(1,1), (1,2), (1,3)] [2 4 6] [(2,1), (2,2), (2,3)]	pass
		1) Check that the index method selects the correct object.	pass
04	Tests that the index method works with a list of TIMESPAN objects as input.	The index method doesn't work for a list of TIMESPAN objects as input.	pass
		1) Nothing to test.	pass
05	Tests that the index method properly applies history.	Test that the result of index have an additional history step.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'index'.	pass
06	Tests that the index method works for the modifier command.	Tests that the index method works for the modifier command.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
07	Control the method with a plist.	Test that the index method can be controled with a plist.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass
08	Test that the index method selects more objects if I have more indices.	Test that the index method selects more objects if I have more indices.	pass
		1) Check that the history-plist contains the used indices. 2) Check that the index method selects the correct object	pass

timespan/index			

Table 368: Unit tests for timespan/index.

timespan/isprop			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the isprop method works with a vector of TIMESPAN objects as input.	Test that the isprop method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsv' 2) Check that each output contains the correct data.	pass
03	Tests that the isprop method works with a matrix of TIMESPAN objects as input.	Test that the isprop method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsm' 2) Check that each output contains the correct data.	pass
04	Tests that the isprop method works with a list of TIMESPAN objects as input.	Test that the isprop method works for a list of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the isprop method works with a mix of different shaped TIMESPAN objects as input.	Test that the isprop method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the isprop method properly applies history.	The method isprop doesn't change the object, thus it is not necessary to apply history.	pass
			pass
07	Tests that the isprop method works for each property.	Test that the isprop method works for the properties: startT', 'endT', 'timeformat', 'timezone', 'interval' 'hist' and 'name'	pass
		1) Check that each output contains the correct data.	pass
08	Test the negative case and the not function command.	Test that the isprop method retrun false for a unknown property and for methods of the object.	pass

timespan/isprop			
		1) Check that each output contains the correct data.	pass

Table 369: Unit tests for timespan/isprop.

timespan/loadobj			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check the shape of the loaded objects.	pass

Table 370: Unit tests for timespan/loadobj.

timespan/ne			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the ne method works with a vector of TIMESPAN objects as input.	Test that the ne method works for a vector of TIMESPAN objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
03	Tests that the ne method works with a matrix of TIMESPAN objects as input.	Test that the ne method works for a matrix of TIMESPAN objects as input. Test the positive and the negative case.	pass
		1) Check the output of the ne function.	pass
04	Tests that the ne method works with a list of TIMESPAN objects as input.	The ne method doesn't works for a list of TIMESPAN objects as input. Nothing to do.	pass
			pass
05	Tests that the ne method works with a mix of different shaped TIMESPAN objects as input.	The ne method doesn't works for a list of TIMESPAN objects as input. Nothing to do.	pass
			pass
06	Tests that the ne method properly applies history.	The ne method doesn't change the TIMESPAN object, thus will no history added. Nothing to do	pass
			pass
07	Test the ne method with an exception list. The function timespan/ne use the function timespan/eq so it is not necessary to check all possibilities of the exception list.	Test the ne method with the exception 'name'. Use the option 'internal' to suppress the history. It is necessary to add 'created' to the exception list because ts is created at an other time.	pass
		1) Check that each output contains the correct data.	pass
08	Test the ne method with an exception list which is in a plist.	Test that the ne method uses the exception list in a plist.	pass
		1) Check that each output contains the correct data.	pass

Table 371: Unit tests for timespan/ne.

timespan/rebuild			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the rebuild method works with a vector of TIMESPAN objects as input.	Test that the rebuild method works for a vector of TIMESPAN objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the rebuild method works with a matrix of TIMESPAN objects as input.	Test that the rebuild method works for a matrix of TIMESPAN objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the rebuild method works with a list of TIMESPAN objects as input.	Test that the rebuild method works for a list of TIMESPAN objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the rebuild method works with a mix of different shaped TIMESPAN objects as input.	Test that the rebuild method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the rebuild method properly applies history.	The method rebuild doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Check that the rebuild method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 372: Unit tests for timespan/rebuild.

timespan/save			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the save method works with a vector of TIMESPAN objects as input.	Test that the save method works for a vector of TIMESPAN objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'tsv' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
03	Tests that the save method works with a matrix of TIMESPAN objects as input.	Test that the save method works for a matrix of TIMESPAN objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in 'tsm' 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same.	pass
04	Tests that the save method works with a list of TIMESPAN objects as input.	Test that the save method works for a list of TIMESPAN objects as input. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out1' and 'out2' are the same as in the list 2) Check that the loaded objects are the same as the saved objects. 3) The outputs 'out1' and 'out2' must be the same except.	pass
05	Tests that the save method works with a mix of different shaped TIMESPAN objects as input.	Test that the save method works with an input of matrices and vectors and single TIMESPAN objects. Test both formats 'xml' and 'mat'.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output TIMESPAN object contains the correct data.	pass

timespan/save			
06	Tests that the save method properly applies history.	Test that the result of applying the save method can be processed back to an m-file. Do this for both extensions 'mat' and 'xml'	pass
		1) Check that the history applies to the output object. Check that save doesn't add a history step to the input object. 2) Check that the read object doesn't contain the save + load history steps. 3) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the save method works with the modify command.	Use the save method with the modifier command.	pass
		1) Check that the save method doesn't apply the history. 2) Check the output against the input. 3) Check the history of the output against the input.	pass
08	Control the method with a plist.	Test that the save method uses the filename which is stored in a plist.	pass
		1) Check the output	pass

Table 373: Unit tests for timespan/save.

timespan/setEndT			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setEndT method works with a vector of TIMESPAN objects as input.	Test that the setEndT method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsv' 2) Check that each output contains the correct data.	pass
03	Tests that the setEndT method works with a matrix of TIMESPAN objects as input.	Test that the setEndT method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsm' 2) Check that each output contains the correct data.	pass
04	Tests that the setEndT method works with a list of TIMESPAN objects as input.	Test that the setEndT method works for a list of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setEndT method works with a mix of different shaped TIMESPAN objects as input.	Test that the setEndT method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setEndT method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setEndT method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setEndT'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setEndT'. 3) Check that the method rebuild produces the same object as 'out'.	pass

timespan/setEndT			
07	Tests that the setEndT method can modify the input TIMESPAN object.	Test that the setEndT method can modify the input TIMESPAN object by calling with no output.	pass
		1) Check that 'ts5' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
08	Tests that the setEndT method can set the property with a plist.	Test that the setEndT method can modify the property 'endT' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setEndT method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the setEndT method accept different inputs.	Test that the setEndT method accept input of double, char and time-objects.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 374: Unit tests for timespan/setEndT.

timespan/setName			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setName method works with a vector of TIMESPAN objects as input.	Test that the setName method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsv' 2) Check that each output contains the correct data.	pass
03	Tests that the setName method works with a matrix of TIMESPAN objects as input.	Test that the setName method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsm' 2) Check that each output contains the correct data.	pass
04	Tests that the setName method works with a list of TIMESPAN objects as input.	Test that the setName method works for a list of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setName method works with a mix of different shaped TIMESPAN objects as input.	Test that the setName method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setName method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setName method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setName'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setName'. 3) Check that the method rebuild produces the same object as 'out'.	pass

timespan/setName			
07	Tests that the setName method can modify the input TIMESPAN object.	Test that the setName method can modify the input TIMESPAN object by calling with no output.	pass
		1) Check that 'ts5' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
08	Tests that the setName method can set the property with a plist.	Test that the setName method can modify the property 'name' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setName method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 375: Unit tests for timespan/setName.

timespan/setStartT			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the setStartT method works with a vector of TIMESPAN objects as input.	Test that the setStartT method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsv' 2) Check that each output contains the correct data.	pass
03	Tests that the setStartT method works with a matrix of TIMESPAN objects as input.	Test that the setStartT method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in 'tsm' 2) Check that each output contains the correct data.	pass
04	Tests that the setStartT method works with a list of TIMESPAN objects as input.	Test that the setStartT method works for a list of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
05	Tests that the setStartT method works with a mix of different shaped TIMESPAN objects as input.	Test that the setStartT method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check that the number of elements in 'out' is the same as in input. 2) Check that each output contains the correct data.	pass
06	Tests that the setStartT method properly applies history and that the option 'internal' suppresses the history.	Test that the result of applying the setStartT method can be processed back to an m-file.	pass
		1) Check that the last entry in the history of 'out1' corresponds to 'setStartT'. 2) Check that the last entry in the history of 'out2' NOT corresponds to 'setStartT'. 3) Check that the method rebuild produces the same object as 'out'.	pass

timespan/setStartT			
07	Tests that the setStartT method can modify the input TIMESPAN object.	Test that the setStartT method can modify the input TIMESPAN object by calling with no output.	pass
		1) Check that 'ts5' and 'ain' are now different. 2) Check that 'ain' has the correct name field	pass
08	Tests that the setStartT method can set the property with a plist.	Test that the setStartT method can modify the property 'startT' with a value in a plist.	pass
		1) Check that 'ain' has the correct name field 2) Check that the method rebuild produces the same object as 'out'.	pass
09	Check that the setStartT method pass back the output objects to a list of output variables or to a single variable.	Call the method with a list of output variables and with a single output variable. Additionaly check that the rebuild method works on the output.	pass
		1) Check that the output contains the right number of objects 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Check that the setStartT method accept different inputs.	Test that the setStartT method accept input of double, char and time-objects.	pass
		1) Check the output 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 376: Unit tests for timespan/setStartT.

timespan/string			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the string method works with a vector of TIMESPAN objects as input.	Test that the string method works for a vector of TIMESPAN objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
03	Tests that the string method works with a matrix of TIMESPAN objects as input.	Test that the string method works for a matrix of TIMESPAN objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
04	Tests that the string method works with a list of TIMESPAN objects as input.	Test that the string method works for a list of TIMESPAN objects as input.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
05	Tests that the string method works with a mix of different shaped TIMESPAN objects as input.	Test that the string method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check that the output is a executable string. 2) Check the correct number of rout 3) Check the rebuild objects.	pass
06	Tests that the string method properly applies history.	The method string doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass
07	Tests that the string method doesn't work if the TIMESPAN object have more than one history step.	The method string throws an error because the input object have more than one history step.	pass
			pass

Table 377: Unit tests for timespan/string.

timespan/timespan			
00	Tests that the timespan constructor does what is supposed to do.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the timespan method works with a vector of TIMESPAN objects as input.	Test that the timespan method works with a vector of TIMESPAN objects as input.	pass
		1) Check that the shape of the output TIMESPANs is the same as the input shape. 2) Check that each output TIMESPAN is a copy of the input TIMESPAN. 3) Check that the copy have an additional history step.	pass
03	Tests that the timespan method works with a matrix of TIMESPAN objects as input.	Test that the timespan method works with a matrix of TIMESPAN objects as input.	pass
		1) Check that the shape of the output TIMESPANs is the same as the input shape. 2) Check that each output TIMESPAN is a copy of the input TIMESPAN. 3) Check that the copy have an additional history step.	pass
04	Tests that the timespan method works with a list of TIMESPAN objects as input.	Test that the timespan method works with a list of TIMESPAN objects as input.	pass
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output TIMESPAN is a copy of the input TIMESPAN. 3) Check that the copy have an additional history step.	pass
05	Tests that the timespan method works with a mix of different shaped TIMESPANs as input.	Test that the timespan method works with a mix of different shaped TIMESPANs as input.	pass

timespan/timespan			
		1) Check that the number of elements in 'out' is the same of the number in the input. 2) Check that each output TIMESPAN is a copy of the input TIMESPAN. 3) Check that the copy have an additional history step.	pass
06	Tests that the timespan method properly applies history.	Test that the result of applying the timespan method can be processed back.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timespan'. 2) Check that the method rebuild produces the same object as 'out'.	pass
07	Tests that the timespan method properly applies history to the copy constructor.	Test that the output can be processed back with the 'rebuild' method. Test the constructor with a different number of inputs.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timespan'. 2) Check that the original objects are not changed by the setter function 3) Check that the method rebuild produces the same object as 'out'.	pass
08	Tests that the timespan method properly applies history to the read MAT-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the 'rebuild' method produces the same object as 'out'.	pass
09	Tests that the timespan method properly applies history to the read XML-file constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the 'rebuild' method produces the same object as 'out'.	pass
10	Tests that the timespan method properly applies history to the struct constructor.	Test that the output can be processed back with the 'rebuild' method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timespan'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
11	Tests that the timespan method properly applies history to the plist(filename) constructor.	Test that the output can be processed back to an m-file.	pass

timespan/timespan			
		1) Check that the save method doesn't change the input object 2) Check that the last two entries in the history of 'out' corresponds to 'timespan' and 'save' 3) Check that the 'rebuild' method produces the same object as 'out'.	pass
12	Tests that the TIMESPAN method properly applies history to the plist(conn) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timespan'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
13	Tests that the TIMESPAN method properly applies history to the plist(type) constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timespan'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
15	Tests that the TIMESPAN method properly applies history to the start + end time constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timespan'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
16	Tests that the TIMESPAN method properly applies history to the conn+Id constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timespan'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass
17	Tests that the TIMESPAN method properly applies history to the start + end time + format constructor.	Test that the output can be processed back with the rebuild method.	pass
		1) Check that the last entry in the history of 'out' corresponds to 'timespan'. 2) Check that the 'rebuild' method produces the same object as 'out'.	pass

Table 378: Unit tests for timespan/timespan.

timespan/type			
01	Tests that the getInfo call works for this method.	Test that the getInfo call works for no sets, all sets, and each set individually.	pass
		1) Check that getInfo call returned an minfo object in all cases. 2) Check that all plists have the correct parameters.	pass
02	Tests that the type method works with a vector of TIMESPAN objects as input.	Test that the type method works for a vector of TIMESPAN objects as input.	pass
		1) Check the rebuilt output.	pass
03	Tests that the type method works with a matrix of TIMESPAN objects as input.	Test that the type method works for a matrix of TIMESPAN objects as input.	pass
		1) Check the rebuilt output.	pass
04	Tests that the type method works with a list of TIMESPAN objects as input.	Test that the type method works for a list of TIMESPAN objects as input.	pass
		1) Check the rebuilt output.	pass
05	Tests that the type method works with a mix of different shaped TIMESPAN objects as input.	Test that the type method works with an input of matrices and vectors and single TIMESPAN objects.	pass
		1) Check the rebuilt output.	pass
06	Tests that the type method properly applies history.	The method type doesn't change the data, thus it is not possible to check the history. Nothing to do.	pass
			pass

Table 379: Unit tests for timespan/type.