lpsd revisited: ltf
S2-AEI-TN-3052
2008/02/07 1.1

G. Heinzel
AEI Hannover

February 7, 2008

Revision History

1.0 2008/01/29 initial version.

1.1 2008/02/07 first published version.

1 Introduction

Since the first implementation of the `lpsd` algorithm [1, 2], it has been widely used on many data files. In the context of implementing the algorithm in LTPDA (Lisa Technology Package Data Analysis) and applying it to multi-channel data for cross-spectra and transfer functions, a few modifications have been developed. These are described here, together with a implementation of the scheduler in C that can serve as reference for MATLAB or other ports, and in particular for its implementation in the LTPDA framework.

Since one motivation of revisiting `lpsd` was to apply it to transfer functions, the new algorithm will be called `ltf` (logarithmic transfer function) in order to distinguish it from the original algorithm described in [1].

2 Definition of the problem

By “algorithm” or “scheduler” we mean the short piece of code that determines the list of frequencies on the x-axis and all associated parameters for each frequency such as DFT length, bin number etc. that are necessary to compute the DFT at that frequency. In LTPDA, that algorithm will be MATLAB code while the core DFT may either be MATLAB or optimized C code for efficiency.

For the motivation and basic principles of the algorithm, the reader is referred to [1] and [3].

Compared to the original `lpsd` algorithm, the modifications described here concern

- The low frequency end: Citing [1]:

> *"It is well known that windowing of data segments is necessary in the WOSA method to reduce the bias of the spectral estimate. When calculating onesided spectral estimates containing only positive Fourier frequencies windowing causes a bias at low frequency bins — a fact that is also well known: one cannot trust the lowest frequency bins on the spectrum analyzer. The bias stems from aliasing of power from negative bins and bin zero to the lowest positive frequency bins. Aliasing from bin zero can be eliminated by subtracting the mean data value from the segment. Aliasing from negative bins however, cannot be reduced that way. Hence we propose not to use the first few frequency bins. The first frequency bin that yields unbiased spectral estimates depends on the window function used. The bin is given by the effective half-width of the window transfer function. Values for a variety of windows are tabulated in [3]."*

This feature had so far not been fully implemented in the `lpsd` algorithm. The new `ltf` algorithm described here has an optional input argument b_{min} (`binmin`), the value of which is typically taken from the table of window functions (where it is called `sbin`), and schedules the DFT operations such that no bin below b_{min} is used. Furthermore, the parameters of the low-frequency bins are now adjusted such that the full time series is used for each bin.

- The high-frequency end: Initial attempts to apply the `lpsd` scheduler to multi-channel data highlighted a problem that has since long ago occasionally been described in the literature (see, e.g. [4]):

  If there is a delay between two channels of which the cross-spectra are to be computed, the calculation is distorted when the length of each DFT segment becomes so short as to be comparable with the delay.

  The delay in question can either be a real delay in the physical process that was measured, or a group delay inherent in the transfer function between the channels.

  This phenomenon is not unique to `lpsd`, but also occurs in Welch's classical WOSA algorithm if the segments are short enough. It is, however, more likely to occur in `lpsd` because the high-frequency bins tend to use many averages of short segments.

  While a sophisticated solution to this problem is under investigation, a temporary remedy consists of simply requiring the permissible segment lengths to be above a certain minimum length. This has been implemented in `ltf` with optional parameter L_{min} (`mindftlen`). Its use implies that at the high-frequency end the bins are spaced more densely than a perfect logarithmic spacing would require.

2.1 Input data

The input to the scheduler consists of the following parameters:

N_{data} **(`ndata`):** an (unsigned) integer, the total length of the input time series to be processed.

f_{samp} **(`fsamp`):** a (double precision) floating point number, the sampling frequency of the time series in Hz.

ξ_{ovlap} **(`ovlap`):** a (double precision) floating point number, the desired overlap between adjacent segments in %. This is a property of the chosen window function [3].

b_{min} **(`binmin`):** an (double precision) floating point number, the minimum bin number to be used. This is a property of the chosen window function with typical values between 1 and 8.

L_{min} **(`mindftlen`):** an (unsigned) integer, the smallest allowable segment length to be processed by the DFT. This is a new feature introduced for the case of multi-channel applications which have a delay between their signal contents, as discussed above.

J_{des} **(`freqdes`):** an (unsigned) integer, the desired number of frequencies on the x axis. A typical value might be 1000. Within the algorithm J_{des} is used to determine the spacing between frequencies (such as if one would specify 'n points per octave') at middle and high frequencies. At the low frequency end, this spacing cannot be maintained such that the final number of frequencies will typically be smaller than J_{des}, but could also be higher if the optional specification of L_{min} causes frequency bins to be more crowded at high frequencies.

K_{des} **(`avgdes`):** an (unsigned) integer, the desired number of segments to be averaged. This value is almost nowhere met exactly; at low frequencies only less averages than K_{des} are possible while at high frequencies more are available. K_{des} is thus only used as control parameter in the algorithm to find a compromise between conflicting goals.

2.2 Scheduler output

The main output of the scheduler is a list of frequencies $f_j, j = 0 \dots J-1$ which are as much as possible equally spaced on a logarithmic axis between the extrema

$$f_{\text{min}} = b_{\text{min}} \cdot f_{\text{samp}}/N_{\text{data}} \tag{1}$$

$$f_{\text{max}} = f_{\text{samp}}/2 \tag{2}$$

and the following DFT parameters for each frequency:

r_j **(`fres`):** the frequency resolution (bin width) in Hz, a (double precision) floating point number.

b_j **(`bin`):** the bin number (i.e. number of cycles in one DFT), a (double precision) floating point number. Note that `lpsd/lpsd` require non-integer bin numbers.

L_j **(`dftlen`):** The length of each segment to be processed by the DFT, an (unsigned) integer.

K_j **(`nseg`):** The number of segments to be averaged, an (unsigned) integer.

$q_k^{(j)}$ **(`segstart[]`):** (optional) the set of starting indices of each segment $k = 0 \dots K_j - 1$ within the the complete time series.

2.3 Relations, constraints and targets for the scheduler

The operation of the scheduler is governed by the following set of relations between the parameters for each frequency. The input parameters listed in Section 2.1 are assumed constant. The relations are classified into

- equations between parameters that must always be fulfilled,
- constraints, i.e. limits to parameters expressed by unequalities that must always be fulfilled,
- targets, i.e. relationships that the scheduler strives to fulfill as good as possible.

Some of these are conflicting, otherwise the job would be trivial.

DFT equations

$$\blacktriangleright \quad f_j = r_j \cdot b_j, \tag{3}$$

$$\blacktriangleright \quad f_{\text{samp}} = r_j \cdot L_j. \tag{4}$$

These two 'hard' relations leave only two independent variables in the set of f_j, r_j, b_j and L_j. We nevertheless keep all four of them since the other relations are easier to formulate with all of them available. Only L_j is required to be an integer.

Segment equation

$$\blacktriangleright \quad K_j = \text{round}\left(\frac{N_{\text{data}} - L_j}{\eta \cdot L_j} + 1\right) \tag{5}$$

This equation describes how many segments of length L_j can be fitted into the total data length N_{data} when the non-overlapping fraction is as close as possible to

$$\eta = 1 - \frac{\xi_{\text{ovlap}}}{100\,\%}. \tag{6}$$

This equation is illustrated in Figure 1.

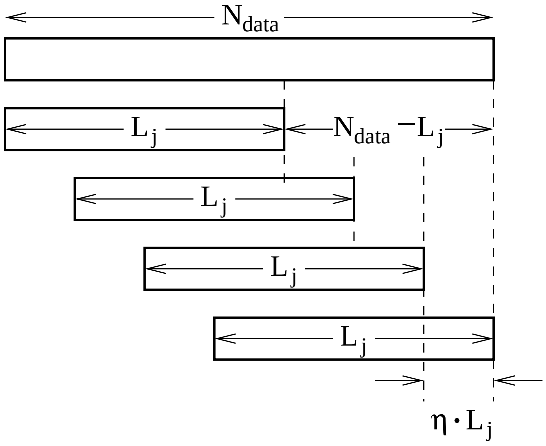

Figure 1: Illustration of Equation (5).

Constraints

- ▶ $L_j \leq N_{\text{data}}$ (7)

No segment can be longer than the total length of the time series.

- ▶ $L_j \geq L_{\text{min}}$ (8)
- ▶ $b_j \geq b_{\text{min}}$ (9)

These are the optional user constraints discussed above.

Targets

- ▶ $f_{j+1} - f_j = r_j$ (10)

This ensures that the local spacing between frequency bins has the same relationship with the bin width as in the original WOSA method. If $f_{j+1} - f_j$ was considerably larger than r_j, spectral features could be missed because they would not be captured by any frequency bin. If, on the other hand, $f_{j+1} - f_j$ was considerably smaller than r_j, the frequency bins would overlap so much that they would contain mostly redundant information.

An alternative would be to require $f_{j+1} - f_j = r_{j+1}$ or some intermediate solution in between these two. We have chosen Eu. (10) since it ensures that the frequency spacing to adjacent bins is either equal or slightly smaller, but never larger, than the resolution bandwidth r_j.

- ▶ $\frac{r_j}{f_j} \approx \text{const}$ (11)

This ensures the logarithmic spacing of frequencies on the x-axis. The constant has the value

$$\left(\frac{N_{\text{data}}}{2}\right)^{(1/J_{\text{des}})} - 1 \quad (12)$$

- ▶ $L_j = N_{\text{data}}, \quad \text{if} \quad K_j = 1$ (13)

This describes the requirement to use the complete time series. In the case of $K_j > 1$, the starting points of the individual segments can and will be adjusted such that the complete time series is used, at the expense of not precisely achieving the desired overlap (see Section 4 below).

- ▶ $K_j \geq K_{des}$ (14)

This describes the desire to have at least K_{des} (`avgdes`) segments for averaging at each frequency. As mentioned above, this cannot be met at low frequencies but is easy to over-achieve at high frequencies, such that this serves only as a guideline for finding compromises in the scheduler.

3 Algorithm

With the above preliminaries, the simplified C code of the algorithm should be understandable. Error checking, memory allocation etc. have been omitted for clarity. The full C code is available from the author.

```
void ltf_plan
  (unsigned ndata, double fsamp, double ovlap, double binmin,
   unsigned mindftlen, unsigned freqdes, unsigned avgdes)
{
  double xov, fmin, fmax, fres, f, bin,
         freslim, fresmin, logfact;
  unsigned j, k, dftlen, segstartpos, nseg;

  xov = (1. - ovlap / 100.);                                  Equation (6)
  fmin = fsamp / ndata * binmin;                              Equation (1)
  fmax = fsamp / 2;                                           Equation (2)
  fresmin = fsamp / (double) ndata;                           see Ref. [1]
  freslim = fresmin * (1 + xov * (avgdes - 1));               see Ref. [1]
  logfact = (pow (ndata / 2., 1. / freqdes) - 1.);            Equation (12)

  f = fmin;
  for (j = 0; f < fmax; j++)
    {
      fres = f * logfact;                                     Target (11)
      fres = (fres > freslim) ? fres : sqrt (fres * freslim); see Ref. [1]
      if (fres < fresmin)
        fres = fresmin;                                       Constraint (7)

      bin = (f / fres);                                       Equation (3)
      if (bin < binmin)
        {
          bin = binmin;                                       Constraint (9)
          fres = f / bin;                                     Equation (3)
        }

      dftlen = round (fsamp / fres);                          Equation (4)
      if (dftlen > ndata)
        dftlen = ndata;                                       Constraint (7)
      if (dftlen < mindftlen)
        dftlen = mindftlen;                                   Constraint (8)
      nseg = round (((double) (ndata - dftlen) /
                     (xov * (double) dftlen)) + 1);           Equation (5)
      if (nseg == 1)
        dftlen = ndata;                                       Target (13)

      fres = fsamp / dftlen;                                  Equation (4)
      bin = f / fres;                                         Equation (3)
/* at this point store the results:
   f, fres, bin, dftlen, nseg */

      f += fres;                                              Target (10)
    }
}
```

4 Distributing the segments

Once the number of segments K_j and the segment length L_j at one frequency bin j are known, the segments starting points must be distributed over the total data length N_{data}. Again, there are conflicting goals: either the prescribed overlap can be exactly met, but then some data at one end of the time series will have to be omitted, or the segments are equally distributed such as to cover the complete time series at the expense of only approximately achieving the desired overlap. We have chosen the second approach such as not to exclude possible features at either end of the time series.

We denote the starting index by $q_k^{(j)}$ with

$$q_k^{(j)} \in [0, N_{\text{data}} - L_j], \quad k = 0, \dots, K_j - 1, \tag{15}$$

and find according to Figure 2:

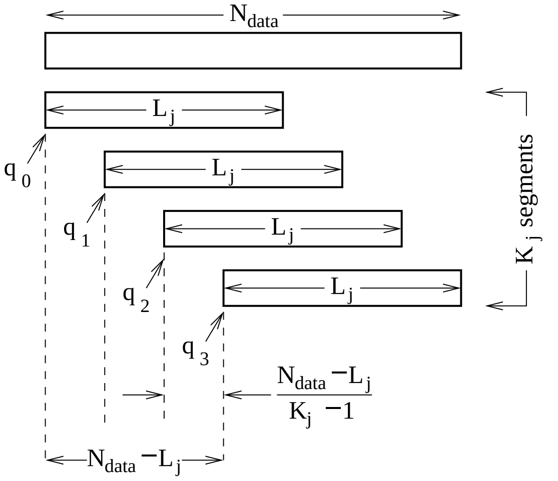

Figure 2: Illustration of segment distribution.

$$q_0^{(j)} = 0, \tag{16}$$

$$q_k^{(j)} = \text{round}\left(k \cdot \frac{N_\text{data} - L_j}{K_j - 1}\right), \quad \text{for} \quad k = 1, \ldots, K_j - 2 \tag{17}$$

$$q_{K_j-1}^{(j)} = N_\text{data} - L_j, \tag{18}$$

$$\tag{19}$$

The complete C code includes the option to compute all starting indices in the scheduler.

5 Sample output

As reference for other implementations, here are the results for the case
N_data (`ndata`) = 1000000
f_samp (`fsamp`) = 10 Hz
ξ_ovlap (`ovlap`) = 75 %
b_min (`binmin`) = 5.3
L_min (`mindftlen`) = 100
J_des (`freqdes`) = 200
K_des (`avgdes`) = 100

Usually, J_des (`freqdes`) will be chosen higher (1000 is a typical value); 200 has been chosen here to save space in the printed table.

Targets that are not met are printed ***like this***

j	f_j	r_j	b_j	K_j	L_j	$\frac{f_j - f_{j-1}}{r_j}$	$\frac{f_{j+1} - f_j}{r_j}$	$\frac{r_j}{c \cdot f_j}$	$q_k^{(j)}$
0	53.00μ	10.00μ	5.30	***1***	1000000	—	1.000	***2.782***	0
1	63.00μ	11.89μ	5.30	***2***	841270	0.841	1.000	***2.782***	0 158730
2	74.89μ	14.13μ	5.30	***3***	707735	0.841	1.000	***2.782***	0 146133 292265
3	89.02μ	16.80μ	5.30	***4***	595396	0.841	1.000	***2.782***	0 134868 269736 404604
4	105.8μ	19.96μ	5.30	***5***	500889	0.841	1.000	***2.782***	0 124778 249556 ... 499111
5	125.8μ	23.73μ	5.30	***6***	421383	0.841	1.000	***2.782***	0 115723 231447 ... 578617
6	149.5μ	28.21μ	5.30	***8***	354497	0.841	1.000	***2.782***	0 92215 184429 ... 645503
7	177.7μ	33.53μ	5.30	***10***	298227	0.841	1.000	***2.782***	0 77975 155950 ... 701773
8	211.2μ	39.86μ	5.30	***13***	250890	0.841	1.000	***2.782***	0 62426 124852 ... 749110
9	251.1μ	47.38μ	5.30	***16***	211066	0.841	1.000	***2.782***	0 52596 105191 ... 788934
10	298.5μ	56.32μ	5.30	***20***	177563	0.841	1.000	***2.782***	0 43286 86572 ... 822437
11	354.8μ	66.94μ	5.30	***24***	149379	0.841	1.000	***2.782***	0 36984 73967 ... 850621
12	421.7μ	79.57μ	5.30	***29***	125668	0.841	1.000	***2.782***	0 31226 62452 ... 874332
13	501.3μ	93.56μ	5.36	***34***	106881	0.851	1.000	***2.752***	0 27064 54128 ... 893119
14	594.9μ	101.9μ	5.84	***38***	98116	0.918	1.000	***2.527***	0 24375 48750 ... 901884
15	696.8μ	110.3μ	6.32	***41***	90657	0.924	1.000	***2.334***	0 22734 45467 ... 909343
16	807.1μ	118.7μ	6.80	***44***	84235	0.929	1.000	***2.169***	0 21297 42594 ... 915765
17	925.8μ	127.1μ	7.28	***48***	78649	0.934	1.000	***2.025***	0 19603 39206 ... 921351
18	1.053m	135.6μ	7.77	***51***	73748	0.938	1.000	***1.899***	0 18525 37050 ... 926252
19	1.189m	144.1μ	8.25	***55***	69414	0.941	1.000	***1.787***	0 17233 34466 ... 930586
20	1.333m	152.5μ	8.74	***58***	65555	0.944	1.000	***1.688***	0 16394 32788 ... 934445
21	1.485m	161.0μ	9.22	***61***	62097	0.947	1.000	***1.599***	0 15632 31263 ... 937903

j	f_j	r_j	b_j	K_j	L_j	$\frac{f_j - f_{j-1}}{r_j}$	$\frac{f_{j+1} - f_j}{r_j}$	$\frac{r_j}{c \cdot f_j}$	$q_k^{(j)}$
22	1.646m	169.5μ	9.71	***65***	58981	0.950	1.000	***1.519***	0 14703 29407 ... 941019
23	1.816m	178.1μ	10.20	***68***	56160	0.952	1.000	***1.446***	0 14087 28174 ... 943840
24	1.994m	186.6μ	10.69	***72***	53594	0.954	1.000	***1.380***	0 13330 26659 ... 946406
25	2.180m	195.1μ	11.17	***75***	51249	0.956	1.000	***1.320***	0 12821 25642 ... 948751
26	2.376m	203.7μ	11.66	***78***	49099	0.958	1.000	***1.264***	0 12349 24699 ... 950901
27	2.579m	212.2μ	12.15	***82***	47121	0.960	1.000	***1.213***	0 11764 23528 ... 952879
28	2.791m	220.8μ	12.64	***85***	45294	0.961	1.000	***1.166***	0 11366 22731 ... 954706
29	3.012m	229.3μ	13.13	***89***	43603	0.963	1.000	***1.123***	0 10868 21736 ... 956397
30	3.242m	237.9μ	13.62	***92***	42032	0.964	1.000	***1.082***	0 10527 21054 ... 957968
31	3.479m	246.5μ	14.12	***96***	40570	0.965	1.000	***1.045***	0 10099 20199 ... 959430
32	3.726m	255.1μ	14.61	***99***	39205	0.966	1.000	***1.010***	0 9804 19608 ... 960795
33	3.981m	270.0μ	14.75	105	37042	0.945	1.000	1.000	0 9259 18518 ... 962958
34	4.251m	288.3μ	14.75	112	34690	0.937	1.000	1.000	0 8696 17393 ... 965310
35	4.539m	307.8μ	14.75	120	32487	0.936	1.000	1.000	0 8130 16261 ... 967513
36	4.847m	328.7μ	14.75	128	30424	0.936	1.000	1.000	0 7634 15269 ... 969576
37	5.176m	351.0μ	14.75	137	28492	0.936	1.000	1.000	0 7143 14287 ... 971508
38	5.527m	374.8μ	14.75	147	26682	0.936	1.000	1.000	0 6667 13333 ... 973318
39	5.902m	400.2μ	14.75	157	24988	0.937	1.000	1.000	0 6250 12500 ... 975012
40	6.302m	427.3μ	14.75	168	23401	0.936	1.000	1.000	0 5848 11696 ... 976599
41	6.729m	456.3μ	14.75	180	21915	0.936	1.000	1.000	0 5464 10928 ... 978085
42	7.185m	487.3μ	14.75	192	20523	0.936	1.000	1.000	0 5128 10256 ... 979477
43	7.673m	520.3μ	14.75	205	19220	0.937	1.000	1.000	0 4808 9615 ... 980780
44	8.193m	555.6μ	14.75	219	17999	0.936	1.000	1.000	0 4505 9009 ... 982001
45	8.748m	593.3μ	14.75	234	16856	0.936	1.000	1.000	0 4220 8439 ... 983144
46	9.342m	633.5μ	14.75	250	15786	0.937	1.000	1.000	0 3953 7905 ... 984214
47	9.975m	676.5μ	14.75	268	14783	0.936	1.000	1.000	0 3690 7380 ... 985217
48	10.65m	722.3μ	14.75	286	13844	0.936	1.000	1.000	0 3460 6920 ... 986156
49	11.37m	771.3μ	14.75	306	12965	0.937	1.000	1.000	0 3236 6472 ... 987035
50	12.15m	823.6μ	14.75	326	12142	0.937	1.000	1.000	0 3040 6079 ... 987858
51	12.97m	879.4μ	14.75	349	11371	0.937	1.000	1.000	0 2841 5682 ... 988629
52	13.85m	939.1μ	14.75	373	10649	0.937	1.000	1.000	0 2660 5319 ... 989351
53	14.79m	1.003m	14.75	398	9972	0.936	1.000	1.000	0 2494 4988 ... 990028
54	15.79m	1.071m	14.75	425	9339	0.937	1.000	1.000	0 2336 4673 ... 990661
55	16.86m	1.143m	14.75	454	8746	0.937	1.000	1.000	0 2188 4376 ... 991254
56	18.00m	1.221m	14.75	485	8191	0.937	1.000	1.000	0 2049 4098 ... 991809
57	19.23m	1.304m	14.75	519	7670	0.936	1.000	1.000	0 1916 3831 ... 992330
58	20.53m	1.392m	14.75	554	7183	0.937	1.000	1.000	0 1795 3591 ... 992817
59	21.92m	1.487m	14.75	592	6727	0.937	1.000	1.000	0 1681 3361 ... 993273
60	23.41m	1.587m	14.75	632	6300	0.937	1.000	1.000	0 1575 3150 ... 993700
61	25.00m	1.695m	14.75	675	5900	0.937	1.000	1.000	0 1475 2950 ... 994100
62	26.69m	1.810m	14.75	721	5525	0.936	1.000	1.000	0 1381 2762 ... 994475
63	28.50m	1.933m	14.75	770	5174	0.936	1.000	1.000	0 1294 2587 ... 994826
64	30.43m	2.064m	14.75	822	4846	0.937	1.000	1.000	0 1212 2424 ... 995154
65	32.50m	2.204m	14.75	878	4538	0.936	1.000	1.000	0 1135 2270 ... 995462
66	34.70m	2.353m	14.75	938	4250	0.937	1.000	1.000	0 1063 2125 ... 995750
67	37.05m	2.513m	14.75	1002	3980	0.936	1.000	1.000	0 995 1990 ... 996020
68	39.57m	2.683m	14.75	1070	3727	0.936	1.000	1.000	0 932 1864 ... 996273
69	42.25m	2.865m	14.74	1143	3490	0.936	1.000	1.000	0 873 1745 ... 996510
70	45.11m	3.059m	14.75	1221	3269	0.937	1.000	1.000	0 817 1634 ... 996731

j	f_j	r_j	b_j	K_j	L_j	$\frac{f_j - f_{j-1}}{r_j}$	$\frac{f_{j+1} - f_j}{r_j}$	$\frac{r_j}{c \cdot f_j}$	$q_k^{(j)}$
71	48.17m	3.267m	14.75	1304	3061	0.936	1.000	1.000	0 765 1530 ... 996939
72	51.44m	3.488m	14.75	1392	2867	0.937	1.000	1.000	0 717 1434 ... 997133
73	54.93m	3.724m	14.75	1487	2685	0.937	1.000	1.000	0 671 1342 ... 997315
74	58.65m	3.978m	14.75	1588	2514	0.936	1.000	1.000	0 629 1257 ... 997486
75	62.63m	4.246m	14.75	1696	2355	0.937	1.000	1.000	0 589 1177 ... 997645
76	66.88m	4.535m	14.75	1811	2205	0.936	1.000	1.000	0 551 1103 ... 997795
77	71.41m	4.843m	14.75	1934	2065	0.937	1.000	1.000	0 516 1033 ... 997935
78	76.25m	5.171m	14.75	2065	1934	0.937	1.000	1.000	0 484 967 ... 998066
79	81.42m	5.522m	14.75	2206	1811	0.936	1.000	1.000	0 453 905 ... 998189
80	86.95m	5.896m	14.75	2355	1696	0.936	1.000	1.000	0 424 848 ... 998304
81	92.84m	6.297m	14.74	2516	1588	0.936	1.000	1.000	0 397 794 ... 998412
82	99.14m	6.725m	14.74	2687	1487	0.936	1.000	1.000	0 372 743 ... 998513
83	105.9m	7.179m	14.75	2869	1393	0.937	1.000	1.000	0 348 696 ... 998607
84	113.0m	7.663m	14.75	3062	1305	0.937	1.000	1.000	0 326 653 ... 998695
85	120.7m	8.183m	14.75	3270	1222	0.936	1.000	1.000	0 306 611 ... 998778
86	128.9m	8.741m	14.74	3494	1144	0.936	1.000	1.000	0 286 572 ... 998856
87	137.6m	9.337m	14.74	3732	1071	0.936	1.000	1.000	0 268 535 ... 998929
88	147.0m	9.970m	14.74	3985	1003	0.937	1.000	1.000	0 251 502 ... 998997
89	156.9m	10.64m	14.75	4252	940	0.937	1.000	1.000	0 235 470 ... 999060
90	167.6m	11.36m	14.75	4542	880	0.936	1.000	1.000	0 220 440 ... 999120
91	178.9m	12.14m	14.74	4851	824	0.936	1.000	1.000	0 206 412 ... 999176
92	191.1m	12.95m	14.75	5178	772	0.937	1.000	1.000	0 193 386 ... 999228
93	204.0m	13.83m	14.75	5530	723	0.937	1.000	1.000	0 181 361 ... 999277
94	217.9m	14.77m	14.75	5905	677	0.936	1.000	1.000	0 169 339 ... 999323
95	232.6m	15.77m	14.75	6306	634	0.936	1.000	1.000	0 159 317 ... 999366
96	248.4m	16.84m	14.76	6731	594	0.937	1.000	0.999	0 149 297 ... 999406
97	265.2m	17.99m	14.75	7191	556	0.936	1.000	1.000	0 139 278 ... 999444
98	283.2m	19.19m	14.76	7675	521	0.937	1.000	0.999	0 130 260 ... 999479
99	302.4m	20.49m	14.76	8194	488	0.937	1.000	0.999	0 122 244 ... 999512
100	322.9m	21.88m	14.76	8750	457	0.936	1.000	0.999	0 114 228 ... 999543
101	344.8m	23.36m	14.76	9343	428	0.937	1.000	0.999	0 107 214 ... 999572
102	368.2m	24.94m	14.76	9972	401	0.937	1.000	0.999	0 100 201 ... 999599
103	393.1m	26.67m	14.74	10664	375	0.935	1.000	1.000	0 94 187 ... 999625
104	419.8m	28.49m	14.73	11393	351	0.936	1.000	1.001	0 88 176 ... 999649
105	448.3m	30.40m	14.75	12155	329	0.937	1.000	1.000	0 82 165 ... 999671
106	478.6m	32.47m	14.74	12984	308	0.936	1.000	1.000	0 77 154 ... 999692
107	511.1m	34.60m	14.77	13838	289	0.938	1.000	0.998	0 72 144 ... 999711
108	545.7m	37.04m	14.73	14812	270	0.934	1.000	1.001	0 67 135 ... 999730
109	582.8m	39.53m	14.74	15807	253	0.937	1.000	1.000	0 63 127 ... 999747
110	622.3m	42.19m	14.75	16875	237	0.937	1.000	1.000	0 59 118 ... 999763
111	664.5m	45.05m	14.75	18015	222	0.937	1.000	1.000	0 56 111 ... 999778
112	709.5m	48.08m	14.76	19228	208	0.937	1.000	0.999	0 52 104 ... 999792
113	757.6m	51.28m	14.77	20510	195	0.938	1.000	0.998	0 49 97 ... 999805
114	808.9m	54.95m	14.72	21975	182	0.933	1.000	1.002	0 46 91 ... 999818
115	863.8m	58.48m	14.77	23389	171	0.940	1.000	0.998	0 43 85 ... 999829
116	922.3m	62.50m	14.76	24997	160	0.936	1.000	0.999	0 40 80 ... 999840
117	984.8m	66.67m	14.77	26664	150	0.938	1.000	0.998	0 37 75 ... 999850
118	1.051	71.43m	14.72	28568	140	0.933	1.000	1.002	0 35 70 ... 999860
119	1.123	76.34m	14.71	30531	131	0.936	1.000	1.002	0 33 66 ... 999869

j	f_j	r_j	b_j	K_j	L_j	$\frac{f_j-f_{j-1}}{r_j}$	$\frac{f_{j+1}-f_j}{r_j}$	$\frac{r_j}{c\cdot f_j}$	$q_k^{(j)}$
120	1.199	81.30m	14.75	32517	123	0.939	1.000	1.000	0 31 62 ... 999877
121	1.281	86.96m	14.73	34780	115	0.935	1.000	1.001	0 29 57 ... 999885
122	1.367	92.59m	14.77	37034	108	0.939	1.000	0.998	0 27 54 ... 999892
123	1.460	99.01m	14.75	39601	101	0.935	1.000	1.000	0 25 50 ... 999899
124	1.559	100.0m	15.59	39997	100	0.990	1.000	***0.946***	0 25 50 ... 999900
125	1.659	100.0m	16.59	39997	100	1.000	1.000	***0.889***	0 25 50 ... 999900
126	1.759	100.0m	17.59	39997	100	1.000	1.000	***0.838***	0 25 50 ... 999900
127	1.859	100.0m	18.59	39997	100	1.000	1.000	***0.793***	0 25 50 ... 999900
128	1.959	100.0m	19.59	39997	100	1.000	1.000	***0.753***	0 25 50 ... 999900
129	2.059	100.0m	20.59	39997	100	1.000	1.000	***0.716***	0 25 50 ... 999900
130	2.159	100.0m	21.59	39997	100	1.000	1.000	***0.683***	0 25 50 ... 999900
131	2.259	100.0m	22.59	39997	100	1.000	1.000	***0.653***	0 25 50 ... 999900
132	2.359	100.0m	23.59	39997	100	1.000	1.000	***0.625***	0 25 50 ... 999900
133	2.459	100.0m	24.59	39997	100	1.000	1.000	***0.600***	0 25 50 ... 999900
134	2.559	100.0m	25.59	39997	100	1.000	1.000	***0.576***	0 25 50 ... 999900
135	2.659	100.0m	26.59	39997	100	1.000	1.000	***0.555***	0 25 50 ... 999900
136	2.759	100.0m	27.59	39997	100	1.000	1.000	***0.534***	0 25 50 ... 999900
137	2.859	100.0m	28.59	39997	100	1.000	1.000	***0.516***	0 25 50 ... 999900
138	2.959	100.0m	29.59	39997	100	1.000	1.000	***0.498***	0 25 50 ... 999900
139	3.059	100.0m	30.59	39997	100	1.000	1.000	***0.482***	0 25 50 ... 999900
140	3.159	100.0m	31.59	39997	100	1.000	1.000	***0.467***	0 25 50 ... 999900
141	3.259	100.0m	32.59	39997	100	1.000	1.000	***0.452***	0 25 50 ... 999900
142	3.359	100.0m	33.59	39997	100	1.000	1.000	***0.439***	0 25 50 ... 999900
143	3.459	100.0m	34.59	39997	100	1.000	1.000	***0.426***	0 25 50 ... 999900
144	3.559	100.0m	35.59	39997	100	1.000	1.000	***0.414***	0 25 50 ... 999900
145	3.659	100.0m	36.59	39997	100	1.000	1.000	***0.403***	0 25 50 ... 999900
146	3.759	100.0m	37.59	39997	100	1.000	1.000	***0.392***	0 25 50 ... 999900
147	3.859	100.0m	38.59	39997	100	1.000	1.000	***0.382***	0 25 50 ... 999900
148	3.959	100.0m	39.59	39997	100	1.000	1.000	***0.372***	0 25 50 ... 999900
149	4.059	100.0m	40.59	39997	100	1.000	1.000	***0.363***	0 25 50 ... 999900
150	4.159	100.0m	41.59	39997	100	1.000	1.000	***0.355***	0 25 50 ... 999900
151	4.259	100.0m	42.59	39997	100	1.000	1.000	***0.346***	0 25 50 ... 999900
152	4.359	100.0m	43.59	39997	100	1.000	1.000	***0.338***	0 25 50 ... 999900
153	4.459	100.0m	44.59	39997	100	1.000	1.000	***0.331***	0 25 50 ... 999900
154	4.559	100.0m	45.59	39997	100	1.000	1.000	***0.323***	0 25 50 ... 999900
155	4.659	100.0m	46.59	39997	100	1.000	1.000	***0.317***	0 25 50 ... 999900
156	4.759	100.0m	47.59	39997	100	1.000	1.000	***0.310***	0 25 50 ... 999900
157	4.859	100.0m	48.59	39997	100	1.000	1.000	***0.303***	0 25 50 ... 999900
158	4.959	100.0m	49.59	39997	100	1.000	—	***0.297***	0 25 50 ... 999900

6 Possible extension

If it turns out that use of the optional $b_{\min}$ (`binmin`) parameter causes many equidistant bins to be scheduled at the upper frequency end, the overall efficiency could be slightly improved by computing these bins all at once with an FFT, at the expense of increased complexity in the program that uses the schedule to perform the DFT. The bin numbers of all these bins would need to be truncated to integers.

References

[1] Tröbs, M., Heinzel, G.: "Improved spectrum estimation from digitized time series on a logarithmic frequency axis", Measurement 39 (2006) 120–129.

[2] Tröbs, M., Heinzel, G.: Corrigendum to “Improved spectrum estimation from digitized time series on a logarithmic frequency axis”, Measurement, to be published.

[3] Heinzel, G.: “Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new at-top windows.”, AEI report 2002.

[4] Seybert A.F., Hamilton J.F.: “Time-delay bias errors in estimating frequency-response and coherence functions”, Journal of Sound and Vibration 60 (1): 1-9, 1978.