DFTAVG User Manual
a FOURIER Transform Tool

Sven Peth
Dandy Eschricht

September 10, 2004

Contents

1 Getting started

When the program is executed with

```
>> dftavg -h
```

the information in Table 1 is plotted on the screen. It contains all the implemented options.

```
implemented options
[-a Navg] number of averagings
[-b beta] for filter function, between -0.5 & 0.5
[-c column] of data array in input file
[-d SPLoutput] SoundPressureLevel(dB)
[-f order] of filter (4, 6 or 8)
[-h] this help
[-i inputfile] name
[-j jump] ignores the first jump# of data elements
[-k length] number of elements of to use data
[-l setl] for FFT/DFT averaging
[-m] no substracting of mean of data before FFT/DFT
[-o outputfile] user defined output file name
[-p pname] Powerspectrum
[-q] quiet
[-r] RMS
[-s shift] shift for Navg in timesteps
[-v] variance (RMS) exact calculation
[-w window]
       window=1: RECTANGLE
       window=2: HANNING
       window=3: TRIANGLE
```

Table 1: Help information

A brief description of these options follows:

`-a Navg:` `Navg` (integer) is the number of averagings used to find the averaged spectrum in the frequency range. See section 4 for more information.

`-b beta:` `beta` (real) is a parameter used for filtering the raw data. If the option `-f` is set, but not `beta` than `beta` is set to 0. `beta` must lie in the range -0.5 and 0.5. The lower values have more impact, 0.5 has no effect. See section 5.

`-c column:` `column` (integer) denotes the column number of the dependent data in the input file to be used. It is set by default to 2. The array of the independent data (t or x) must be in the first column of the input file.

`-d SPLoutput:` `SPLoutput` (string) is the file name in which the Sound-Pressure-Level gets saved. The first column in the file is the array of frequencies in Hz, the second column is the magnitude in dB. The Sound-Pressure-Level (spl) is just a conversion of the

amplitudes ($\hat{p}$) of the FOURIER Spectrum.

$$p_{\text{rms}} = \frac{1}{2}\sqrt{2}\hat{p} \tag{1}$$

$$spl = 20 \cdot \log_{10}\left(\frac{p_{\text{rms}}}{2 \cdot 10^{-5}}\right) \tag{2}$$

`-f order:` This option activates the filtering of high frequencies. Whereby they get damped. `order` (integer) means the order of the filter used. Three orders 4,6 and 8 are implemented. See section 5 for details.

`-h:` shows the help in Table 1 and does nothing.

`-i inputfile:` `inputfile` (string) names the file with the input data. The array of independent variable (t) must be in the first column.

`-j jump:` `jump` (integer) denotes the number of data elements that are to be omitted at the beginning of the inputfile

`-k length:` `length` (integer) means the number of elements that is read from the input file. If omitted the data up to the end of the file is used for transformation

`-l setl:` `setl` (integer) is the number of data elements that is used for one sub-run of the averaging process. If `setl` is not set than it gets computed by `Navg` (option `-a`) and `shift` (option `-s`). If none of the three is set than there is no averaging and there is only one FOURIER Transform with all data elements. See section 4 for more details.

`-m:` If this option is set than there is no subtraction of the mean value of the raw/filtered data from the raw/filtered data, which will give rise to a peak at the first mode of the spectrum. Choosing this option adversely affects the RMS value (option `-r`) of the Power-Spectrum (option `-p`).

`-o output:` `output` (string) names the file name in which the result of the FOURIER Transform is saved. The first column consists of the frequency array, the second column of the amplitudes and the third of the phases. The phases may be incorrect due to the averaging. If this option is not specified than `output` is generated automatically from `inputfile` (option `-i`).

`-p pname:` Enables the calculation of the Power-Spectrum-Density (psd). It is simply a conversion of the corrected FOURIER Spectrum. `pname` (string) is the file name. The first column of the file consists of the frequencies, the second of the magnitudes.

$$psd = \frac{F_k^2}{\Delta f} \tag{3}$$

$$\text{with } \Delta f = \frac{1}{\Delta t \cdot \texttt{setl}} \tag{4}$$

`-q:` Suppresses all screen output.

`-r:` Computes the RMS value of the raw data. If the flag for filtering (option `-f`) and/or Power-Spectrum (option `-p`) is set then their RMS values will also be display on screen.

`-s shift:` `shift` (integer) is the number of elements to shift for a sub-run in the averaging process. See section 4 for more information.

`-v`: Enables variance exact spectrum (FOURIER, Power). Default is amplitude exactness. See section 3.

`-w window`: There are three window functions implemented (Rectangle, Hanning, Triangle). If the data does not start and end with the same value then all frequencies become spurious excited in the FOURIER spectrum. Weight functions (Hanning, Triangle) prevent this noisy behaviour by setting the beginning and end smoothly to zero. See section 3.

2 The Discrete-Fourier-Transform

The DISCRETE-FOURIER-TRANSFORM is based on the technique of interpolating a set of N data points $\hat{f}_i = \hat{f}(t_i) = \hat{f}(\triangle t \cdot i)$ with a set of N base harmonic functions $\Psi_{i,k} = e^{2\pi j \cdot ik/N}$:

$$\hat{f}_i \stackrel{!}{=} f_i = \sum_{k=0}^{N-1} F_k \cdot \Psi_{i,k} = \sum_{k=0}^{N-1} F_k \cdot e^{2\pi j \cdot ik/N} \quad \text{, with } j^2 = -1, \quad \forall\, i = 0 \ldots N-1 \tag{5}$$

where the unknown values to be solved for are the complex magnitudes F_k. This problem is easily solved, because the choosen base functions are orthogonal. Thus, the complex magnitudes F_k belonging to a two-sided spectrum can be calculated independently of one another:

$$F_k = \frac{1}{N} \sum_{i=0}^{N-1} f_i \cdot e^{-2\pi j \cdot ik/N} \triangleq \begin{cases} F\left(\omega_0\, k\right) & : \quad 0 \le k \le N/2 \\ F\left(\omega_0\, (k-N)\right) & : \quad N/2 < k < N \end{cases} \quad \text{with } \omega_0 = \frac{2\pi}{\Delta t\, N} \tag{6}$$

where $N/2$ is defined $N/2 \stackrel{def}{=} (N-1)/2$ if N is an odd number.

Hence, the calculation of all F_k, with $k = 0 \ldots N-1$, requires a number of $\mathcal{O}_{DFT}(N) \sim N^2$ operations. It can easily be shown that if the input data set is not complex, $\Im\{\hat{f}_i\} = 0$, the comlex magnitudes F_k with $N/2 < k < N$ are the complex conjugate of the F_k with $1 \le k < N/2$. In this case the number of required operations reduces to $\mathcal{O}_{DFT}(N) = N^2/2$ and equation 5 simplifies to:

$$f_i = \begin{cases} F_0 + \sum\limits_{k=1}^{N/2} 2|F_k| \cdot \cos(2\pi \cdot ik/N - \varphi_k) & : \quad N \text{ odd} \\ F_0 + |F_{N/2}| \cdot \cos(\pi i - \varphi_{N/2}) + \sum\limits_{k=1}^{N/2-1} 2|F_k| \cdot \cos(2\pi \cdot ik/N - \varphi_k) & : \quad N \text{ even} \end{cases} \tag{7}$$

Thus, only $1 + N/2$ complex magnitudes F_k and phase angles φ_k have to be calculated. This is called the one-sided spectrum and only this will be written to the `output` file.

The FAST-FOURIER-TRANSFORM follows the same method but a special property of equation 6 is exploited to reduce the number of required operations. For the calculation of any of the complex magnitudes F_k, the summation can be split into a sum over the odd and a sum over the even data points. With N as an even number and $M = N/2$ one obtains:

$$\begin{aligned} F_k &= \frac{1}{N} \sum_{i=0}^{N-1} f_i \cdot e^{2\pi j \cdot ik/N} \\ &= \frac{1}{N} \left[\sum_{i=0}^{M-1} f_{2i} \cdot e^{2\pi j \cdot 2ik/N} + \sum_{i=0}^{M-1} f_{2i+1} \cdot e^{2\pi j \cdot (2i+1)k/N} \right] \\ &= \frac{1}{2} \Bigg[\underbrace{\frac{1}{M} \sum_{i=0}^{M-1} f_{2i} \cdot e^{2\pi j \cdot ik/M}}_{F_k^e} + e^{2\pi j \cdot k/N} \underbrace{\frac{1}{M} \sum_{i=0}^{M-1} f_{2i+1} \cdot e^{2\pi j \cdot ik/M}}_{F_k^o} \Bigg] \end{aligned} \tag{8}$$

The complex magnitudes F_k can thus be calculated from the transform F_k^e of the even numbered points and the transform F_k^o of the odd numbered points. The number of required operations is reduced to $\mathcal{O}_{DFT,2} \sim \mathcal{O}_{DFT}(M) + \mathcal{O}_{DFT}(M) = 2M^2 = 2(N/2)^2 = N^2/2$.

If one considers $N = 2^p$, thus an integer power of 2, it is obvious that the splitting can be done p times. Each time, the number of operations required reduces by a faktor of 1/2. This leads to a first estimate of the operations required of $\mathcal{O}_{FFT}(N) \sim N^2/2^p = 2^{2p}/2^p = 2^p = N$. As some more operations are needed when this scheme is applied, the number of required operations is $\mathcal{O}_{FFT}(N) \sim N \cdot \log_2(N)$.

Hence, the application of the FFT results in an essential reduction of computation time. Consider $N = 1024$ as an example. The DFT algorithm would require $\mathcal{O}_{DFT}(1024) \sim 10^6$ operation, while only $\mathcal{O}_{FFT}(1024) \sim 10^4$ operations would be required by the FFT algorithm. The FFT is thus about a 100 times faster than the DFT algorithm.

3 Data windowing

The choosen base functions $\Psi_{i,k}$ of the interpolation in equation 6 can easily be shown to be of periodicity N in i. Applying $e^{j\alpha} = e^{j\alpha+2\pi jk}$, with k an integer number, leads to:

$$\begin{aligned}\Psi_{i,k} &= e^{-2\pi j \cdot ik/N} = e^{-2\pi j \cdot ik/N \mp 2\pi jk} = e^{-2\pi j \cdot (ik/N \pm k)} = e^{-2\pi j \cdot (i \pm N)k/N} \\ &= \Psi_{i\pm N,k}\end{aligned} \tag{9}$$

The interpolating set f_i is thus always periodical $f_i = f_{i\pm N}$, because it is the sum of periodic

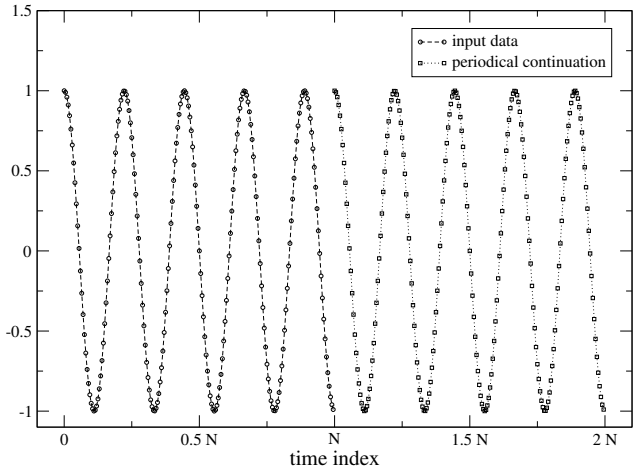

Figure 1: Periodic continuation of non-periodic data set

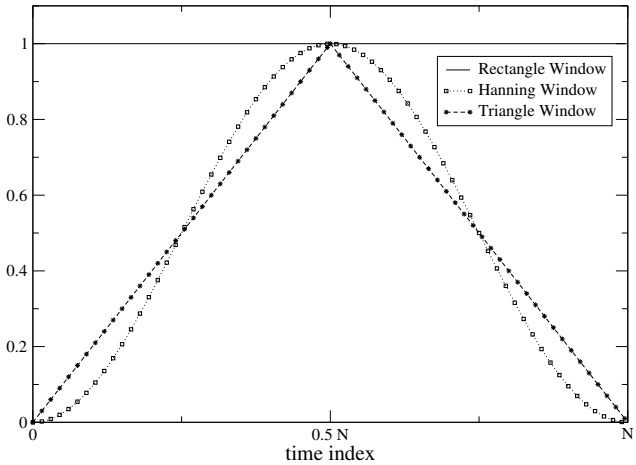

Figure 2: Available window functions

functions. If the input data set $\hat{f}_i$ is non-periodic, the implicit periodic continuation leads to a discontinuity in the set (see Figure 1). Because this discontinuity is also transformed and has a broadband spectrum, the calculated spectrum contains spurious high magnitudes that are not present in the time series. (see Figures 3 & 4)

Figure 1 shows the time series of the function $f(t) = \cos(2\pi\ t \cdot 4.5\text{Hz})$ which was sampled with a timestep size $\triangle t = 5 \cdot 10^{-3}$s for $N = 200$ steps. Therefore, the frequency resolution is $\triangle f = 1/(\triangle t \cdot N) = 1$Hz, and the magnitudes F_k of the angular frequencies $\omega_k = 2\pi\, k \cdot \triangle f$ with $k = 1 \ldots N/2$ are obtained. As can be seen in Figure 1 this function is not periodic in the sampled time interval of $T = N \cdot \Delta t = 1$s.

This problem of the spurious high magnitudes can be overcome by the appliance of window functions. The data set $\hat{f}_i$ is multiplied with a window function $w(i, N)$ that is zero at both

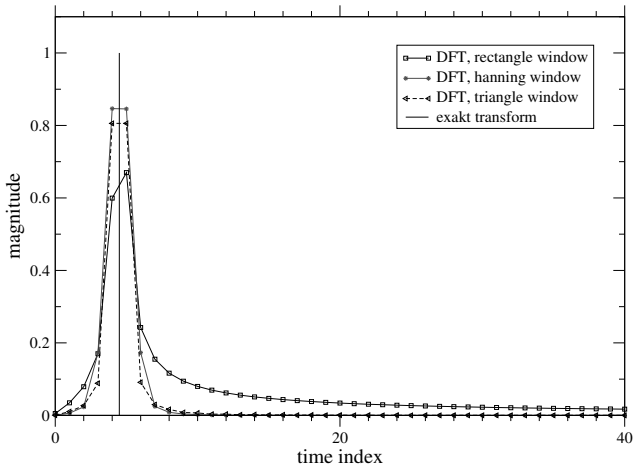

Figure 3: Transform of non-periodic set

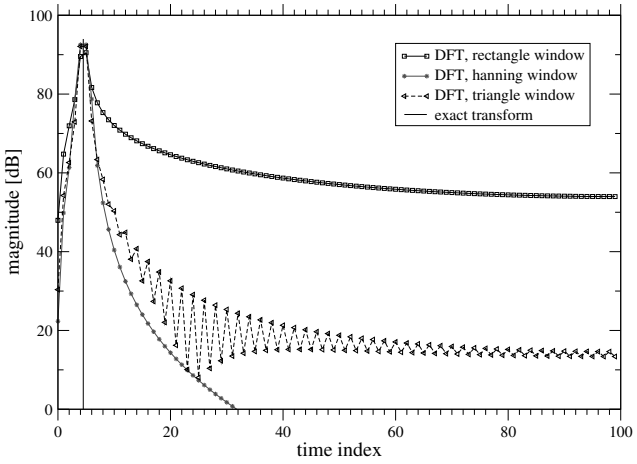

Figure 4: Transform of non-periodic set

ends of the interval $[0, T]$ to form a new set $\tilde{f}_i = w(i, N) \cdot \hat{f}_i$. Hence, this new set is zero at both ends and thus truly periodic. Figures 3 and 4 depict the FOURIER-transform obtained with and without appliance of a window function as well as the analytical solution for an infinite sample rate. The aforementioned spurious high magnitudes are clearly visible but noticeably reduced when a window is used. Furthermore, the spectrum becomes broadened around the analytical frequency, which means that neighbor frequencies have magnitudes not equal to zero. This can be illustrated analytically for the Hanning window $w_{han}(t)$ and a time signal $s(t)$ that is harmonic in the interval $\mathcal{I} = [0, T]$.

$$w_{han}(t) = \frac{1 - \cos(2\pi t/T)}{2}, \quad s(t) = \cos(2\pi\, k \cdot t/T)$$

If this signal is transformed without a window, the transform will show the correct result, that is only one peak magnitude, because the signal is periodic in the transformed time interval $\mathcal{I} = [0, T]$. However, if the Hanning window is applied, the function that will actually be transformed is

$$\begin{aligned} p(t) = w_{han}(t) \cdot s(t) &= \frac{1 - \cos(2\pi t/T)}{2} \cdot \cos(2\pi\, k \cdot t/T) \\ &= \frac{1}{2}\big[\cos(2\pi\, k \cdot t/T) - \cos(2\pi t/T)\cos(2\pi\, k \cdot t/T)\big] \end{aligned} \tag{10}$$

with $\cos(\alpha) + \cos(\beta) = 2\cos(\frac{\alpha+\beta}{2})\cos(\frac{\alpha-\beta}{2})$ the product of cosine-functions can be changed into a sum leading to:

$$p(t) = \frac{1}{2}\cos(2\pi\, k \cdot t/T) - \frac{1}{4}\cos(2\pi\,(k-1) \cdot t/T) - \frac{1}{4}\cos(2\pi\,(k+1) \cdot t/T) \tag{11}$$

which is a sum of harmonic functions in $\mathcal{I}$. When this sum is transformed it results in three magnitudes that are different from zero, these are $G_k = 2|F_k| = 1/2$ and $G_{k-1} = 2|F_{k-1}| = G_{k+1} = 2|F_{k+1}| = 1/4$. This shows clearly the broadening of the spectrum, because the input of a single harmonic results in a spectrum with three harmonic parts. Furthermore, the peak magnitude is clearly underpredicted. There are two possible ways to correct the magnitude.

The first method is the peak magnitude correction, whereby the obtained spectrum is simply multiplied with the correction factor $c_m = 2$. The result in the above example then is $G_k = 1$ and $G_{k-1} = G_{k+1} = 1/2$ which delivers the true expected magnitude for the peak frequency, but has the disadvantage that the variance of the time series is not conserved.

The variance of the FOURIER-transform for the one-side spectrum is:

$$\sigma^2 = \frac{1}{2}\sum_{k=1}^{N/2} G_k^2 \tag{12}$$

Thus, one obtains a variance $\sigma^2 = 3/4$ for the magnitude corrected spectrum, while the original time series has a $\sigma^2 = 1/2$. The broadening of the spectrum in combination with magnitude correction thus results in an overestimation of the variance.

The second method conserves the variance of the time series. The uncorrected result is $G_k = 1/2$ and $G_{k-1} = G_{k+1} = 1/4$ with a variance $\sigma^2 = 3/16$. Comparision with the expected value $\sigma^2 = 1/2$ leads to the correcting factor that must be $c_v = \sqrt{8/3}$.

Note that, the correcting factors can also be obtained by evaluation of certain integrals. The magnitude correcting factor c_m is obtained from the area under the window function itself:

$$\frac{1}{c_m} = \frac{1}{T}\int_0^T w(t)dt \tag{13}$$

what results in $c_m = 2$ for both the Hanning and the Triangle window. The variance preserving factor c_v can be obtained from the area under the squared window function:

$$\frac{1}{c_v} = \sqrt{\frac{1}{T}\int_0^T w^2(t)dt} \tag{14}$$

and one obtains $c_{v,han} = \sqrt{8/3}$ for the Hanning window and $c_{v,tri} = \sqrt{3}$ for the Triangle window.

The option `-v` switches between these two methods. The default is the peak magnitude correction (c_m). If `-v` is set then the second method (c_v) will be applied.

4 Averaged Spectrum

Averaging spectra has three important advantages:

- first of all it contains less values, therefore its calculation and evaluation requires a smaller computational effort,
- spurious fluctuations are eliminated during the process of averaging, hence the spectra becomes smoothed,
- with choosing the segment length, the frequency resolution can be chosen, averaging allows the full information in the data set to enter the obtained spectra anyhow.

For example, if one has a number of data elements that does not fit to the 2^p stencil, say 7000 elements then one had to decide between a DFT with 7000 values and a FFT with 4096 values. Both cases have disadvantages, one in computational time (DFT) and one in losing information by ignoring 2904 values (FFT). Averaging can make use of all data and still be faster than the DFT.

When averaging is activated (`DFTavg→T`), the FOURIER-series will be calculated from the individual FFTs or DFTs of multiple, overlapping segments (default is 50%) of the time series. Either the length of these segments is controlled with `setl` (option `-l`), or the number of averagings is controlled by `Navg` (option `-a`), or the shift of the overlapping part by `shift` (option `-s`). Two options can be set at one time. In the case where three values are set, `Navg` is set according to:

$$\texttt{Navg} = \frac{N_{time} - \texttt{setl}}{\texttt{shift}}, \tag{15}$$

where N_{time} is the length of the whole time series. `Navg` denotes the number of averagings, i.e., $\texttt{Navg} = 3$ means 4 sub-runs. Equation 15 is also applied if only `setl` is set. Then `shift` is set to `setl`/2. If neither `setl` nor `Navg` nor `shift` is set than averaging is deactivated and a single DFT or FFT is performed with $\texttt{setl} = N_{time}$. Given that only `Navg` and `shift` has been set, `setl` is calculated from:

$$\texttt{setl} = N_{time} - \texttt{Navg} \cdot \texttt{shift} \tag{16}$$

Or `shift` is set by:

$$\texttt{shift} = \frac{N_{time} - \texttt{setl}}{\texttt{Navg}} \tag{17}$$

One can deactivate averaging only by not using one of the options `-a`, `-l`, `-s`. To use just a part of the data in the input file use option `-k` to specify the length and/or `-j` to ignore data points.

Suppose that, the parameters have been set in such a way that there are $M = \texttt{Navg}$ segments, each with $N = \texttt{setl}$ time samples. From this setup, $M \times N$ complex magnitudes F_n^m can be calculated. Each complex magnitude $F_n^m = |F|_n^m \cdot e^{j\varphi_n^m}$ belongs to the m^{th} segment and the n^{th} frequency. There are two possible ways of calculating the resulting segment-averaged, complex magnitude $F_n = |F|_n \cdot e^{j\varphi_n}$. This can either be a linear averaging, where the magnitude $|F|_n$ is calculated from:

$$|F|_n = \frac{1}{M} \sum_{m=1}^{M} |F|_n^m, \tag{18}$$

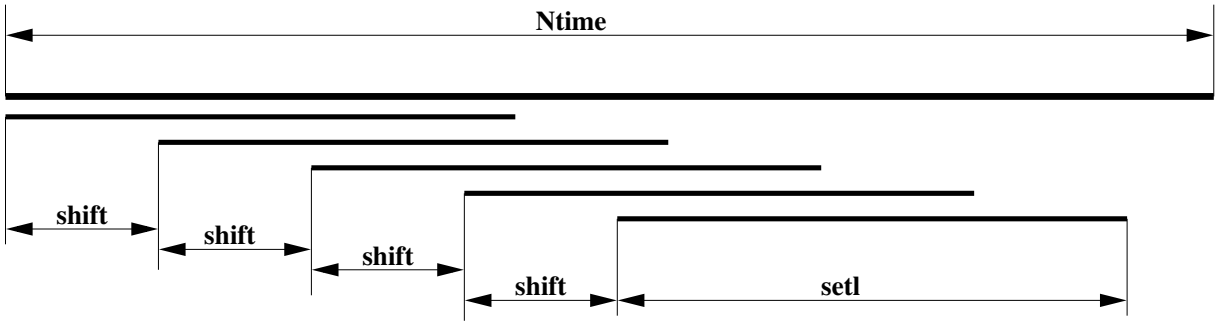

Figure 5: Overlapping segments for creation of averaged spectra, 5 segments thus `Navg` = 4

or a variance-conserving averaging, where $|F|_n$ is calculated according to:

$$|F|_n = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(|F|_n^m\right)^2}. \tag{19}$$

Only the second method ensures that the variance of the FOURIER-transform matches that of the time series. Default is linear averaging. Squared averaging is enabled by option `-v`.
The resulting phase angle φ_n is always calculated from:

$$\varphi_n = \arg\Big(\sum_{m=1}^{M} F_n^m\Big) = \arg\Big(\sum_{m=1}^{M} |F|_n^m \cdot e^{j\varphi_n^m}\Big) \tag{20}$$

which can be interpreted as the magnitude-weighted average of all appearing phase angles φ_n^m. Averaging has a small side effect in the first mode. Normally, the mean value of the entire

data is substracted from the data so that the first mode is always zero. However, this overall mean value is not necessarily the same value as the mean value of the piecewise data that is used for one sub-run of the averaging process. This may result in a first mode being not equal zero, sometimes even negative. As consequence the first mode is always set to 2.E-5 (0dB) after averaging, except if the option `-m` is set.

5 Filtering

A low-pass filter is ideal for the elimination of undesirable high-frequency FOURIER components and does not affect the remaining components of a numerical solution. In addition, the adopted spatial filter modifies only the amplitude and not the phase relationship among all FOURIER components of the solution. The spectral function of a symmetric numerical filter contains no imaginary part and has the ideal low-pass amplitude response. A tridiagonal filter can be given as

$$\beta U_{i-1} + U_i + \beta U_{i+1} = \sum \frac{a_n}{2}(u_{i+n} + u_{i-n}), \quad n = 0,\ 1,\ 2, \ldots,\ N, \tag{21}$$

where the variables U and u represent the filtered and raw values, respectively. The only free parameter of the spatial filter is bound by the values $-0.5 < \beta < 0.5$. It is interesting to note that when the parameter β is assigned a value of zero, the filter becomes an explicit operator. The filter degenerates to an identity, i.e., the computed results will not be modified, when the value of β approches 0.5. The coefficients for a fourth-, sixth, and eight-order filter are included in Table 2. The scheme in 21 is not modified for the boundaries. It is simply assumed that the

Coefficient	Fourth order	Sixth order	Eight order
a_0	$\frac{3(2\beta-1)}{8}+1$	$\frac{5(2\beta-1)}{16}+1$	$\frac{35(2\beta-1)}{128}+1$
a_1	$\frac{(2\beta-1)}{2}+1$	$\frac{17(2\beta-1)}{2}+1$	$\frac{9(2\beta-1)}{16}+1$
a_2	$\frac{2\beta-1}{8}$	$\frac{3(2\beta-1)}{16}$	$\frac{7(2\beta-1)}{32}$
a_3	0	$\frac{-(2\beta-1)}{32}$	$\frac{-2(2\beta-1)}{32}$
a_4	0	0	$\frac{2\beta-1}{128}$

Table 2: Coefficients of Spatial Filter Formula

data is periodic such that all terms of the sum are known. The tridiagonal system of equations is solved by the Thomas algorithm.

Filtering is only useful for data with known high-frequency numerical noise that should not appear in the spectrum. It reduces only high frequencies according to the chosen β and order. Low β and low orders damp more than higher ones.

6 Examples

The easiest way to show the usage of the tool is to perform some examples. In the three sections to follow, fundamental examples are in the first section. Examples concerning filtering and window functions are given in the second and third section, respectively.

6.1 Basics

Here the comparison of FFT, DFT & Averaging shall be performed. First, one needs raw data. This can be created with a simple program in `Matlab`, see Table 3. The advantage of `Matlab` is the straightforward programming and the ease with which noise can be generated by the `randn`-function.

```
clear
t = 0:0.001:7;
x = 0.5*sin(2*pi*30*t)+0.25*sin(2*pi*12*t)+0.25*sin(2*pi*100*t)+1.6666;
x2 = cos(20*pi*t);
y = x+2*randn(size(t));
y2 = x2+2*randn(size(t));
fid=fopen('testsignal','w');
for i=1:size(t,2)
   fprintf(fid,'%12.7g %12.7g %12.7g\n',t(i),y(i),y2(i));
end
fclose(fid);
```

Table 3: simple `Matlab` program

This program writes 2 time dependent signals into the file `testsignal`. The data file's first column consists of the increasing time array, and the second & third columns of the signals. In this example only the second column is used. It has a mean value of 1.666 and three ingrained frequencies (12, 30, 100Hz) with magnitudes of 0.25, 0.5 & 0.25, respectively.

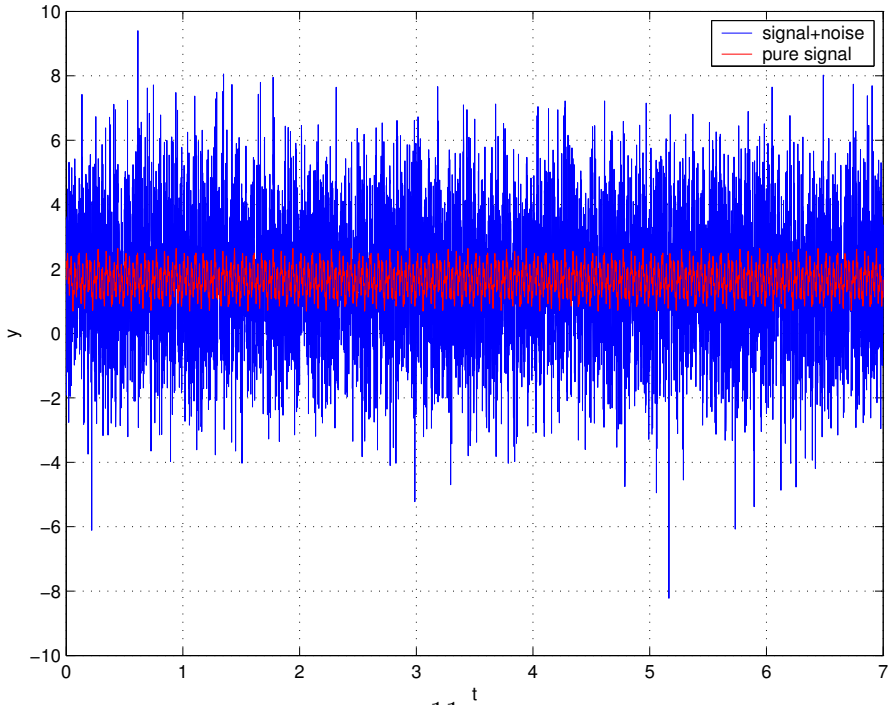

Figure 6: comparison of pure signal and noise overlaying the signal

In total there are 7001 time signal elements. In Figure 6 depicts the pure above descibed signal and the signal plus noise. As one can see the information of the pure signal is totally hidden in the noise.
Now, DFTAVG is used to transform the signal from the time domain to the frequency domain. This is done in three different ways: DFT, FFT & Averaging.

```
>> dftavg -i testsignal -o data_dft
```

This command executes the tool and specifies the input file as `testsignal` and the output file as `data_dft`. It writes some output to the screen:

```
 AVG-DFT FROM: testsignal
           TO: data_dft, WIN: HANNING
    DFT;  NO AVG,  DFTsetl: 7001,  df:  0.142836738
```

It shows that the Hanning window function was used. It performed a DFT with no averaging with a data length of 7001. The frequency difference between two modes is 0.142836738. If the time was given in seconds the unit of `df` is *Hz=1/s*

To perform a pure FFT, the data length must be set to a length within the 2^p-stencil. One enters:

```
>> dftavg -i testsignal -o data_fft -k 4096
```

Now only the first 4096 data elements of the input file are read. It appears on the screen:

```
AVG-DFT FROM: testsignal
           TO: data_fft, WIN: HANNING
    FFT;  NO AVG,  FFTsetl: 4096,  df:  0.244140625
```

FFT is used with no averaging and 4096 data elements. The smaller data array reflects in a coarser resolution of the frequencies. The difference between two modes is now 0.244140625.

The next case uses Averaging:

```
>> dftavg -i testsignal -o data_avg -l 4096 -s 250
```

The length of one Averaging element is 4096 with a shift of 250.

```
 AVG-DFT FROM: testsignal
           TO: data_avg, WIN: HANNING
    FFTAVG LINEAR  ;  FFTnAVG: 11,  FFTsetl: 4096,  df:  0.244140625
    FFTshift: 250;  DATAused: 6846,  DATAavail: 7001
```

The transformation is performed with FFT and Averaging with 11 (`FFTnAVG`) averagings and hence 12 FFT sub-runs. Due to the stencil only 6846 elements of the 7001 data elements were used. The frequency difference between two modes is the same as in the case before, because the data length (4096) is the same.
These runs are compared in Figures 7 & 8. Figure 7 shows the spectra of FFT and Averaging. One can see that the frequencies at 12 & 100 Hz of FFT are larger than by Averaging, but FFT includes several noise frequencies (200, 325, 480Hz). These frequencies are not visible when Averaging is used.

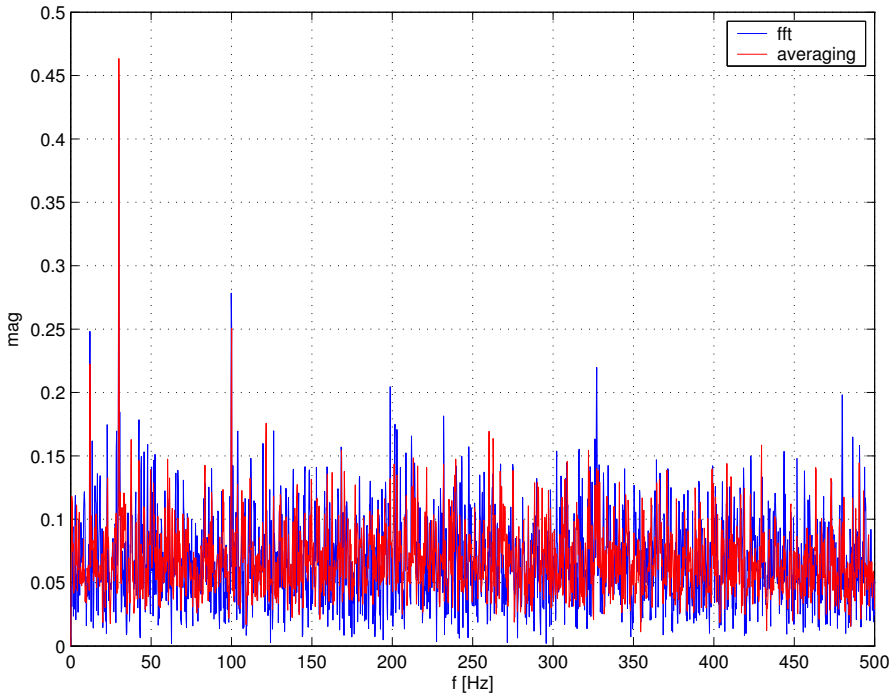

Figure 7: comparison of FFT & Averaging

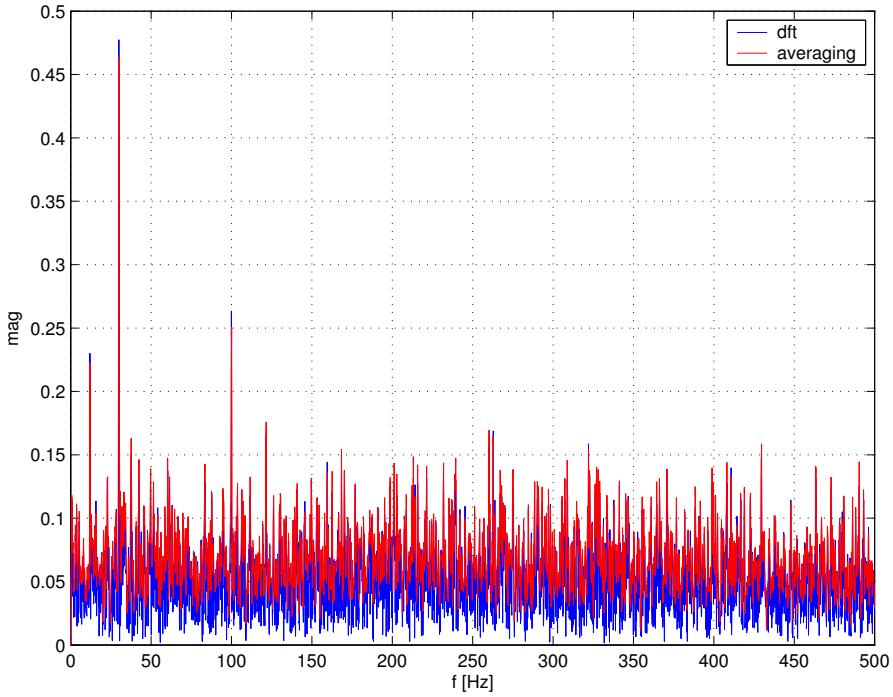

Figure 8: comparison of DFT & Averaging

Figure 8 shows the comparison between DFT & Averaging. One can see that the peaks of the spectrum have almost the same height. Only some noise frequencies are lower with DFT than with Averaging, which is caused by the resolution (data length).

6.2 Filtering

Now the filtering will be demonstrated. In all cases Averaging is used, but here the number of averagings is set to 10 (option `-a`). The RMS value (option `-r`) and the sound pressure level (option `-d`) options are enabled.
For a normal case without filtering one types:

```
>> dftavg -i testsignal -o exam2_f0 -l 4096 -a 10 -r -d exam2_f0_spl
```

The output to the shell reads:

```
 AVG-DFT FROM: testsignal
           TO: exam2_f0, WIN: HANNING
    FFTAVG LINEAR  ;  FFTnAVG: 10,  FFTsetl: 4096,  df:  0.244140625
    FFTshift: 290;  DATAused: 6996,  DATAavail: 7001
 SoundPressure: exam2_f0_spl
      raw DATA RMS:  2.06071363
```

The new features compared to the last example are in the last two lines. The sound pressure level is computed and written to the specified file. And the RMS value of the raw/input data is written to the the screen.

To enable the filter, the order of the filter must be set (option `-f`). Additionally the parameter β can be set (option `-b`). Default is 0.

```
>> dftavg -i testsignal -o exam2_f4 -l 4096 -a 10 -r -d exam2_f4_spl -f 4 -b 0.49
```

The onscreen message is::

```
 AVG-DFT FROM: testsignal
           TO: exam2_f4, WIN: HANNING
    FFTAVG LINEAR  ;  FFTnAVG: 10,  FFTsetl: 4096,  df:  0.244140625
    FFTshift: 290;  DATAused: 6996,  DATAavail: 7001
 SoundPressure: exam2_f4_spl
      raw DATA RMS:  2.06071363
 filtered DATA RMS:  1.94419823
```

The only difference in the output is in the last line. The is the RMS value of the filtered data is given, because the options `-r` (for RMS) and `-f` (for filtering) are set. There is no other way to see that filtering is activated.

The spectra for these two cases are shown in Figure 9. One can see that the graphs for low frequencies are the same. At around 400Hz they start to differ from each other significantly. In Figure 10 the sound pressure level of the two cases are shown.

The sound pressure level (spl, Fig. 10) is in dB which means that the ordinate scale is logarithmic. In the FOURIER spectrum the filtered frequencies tend to zero, hence they tend to minus infinity in the spl. The effect of filtering is thereby more clearly visible, the only effect of filtering is that the high frequencies are damped. The usefulness may vary.

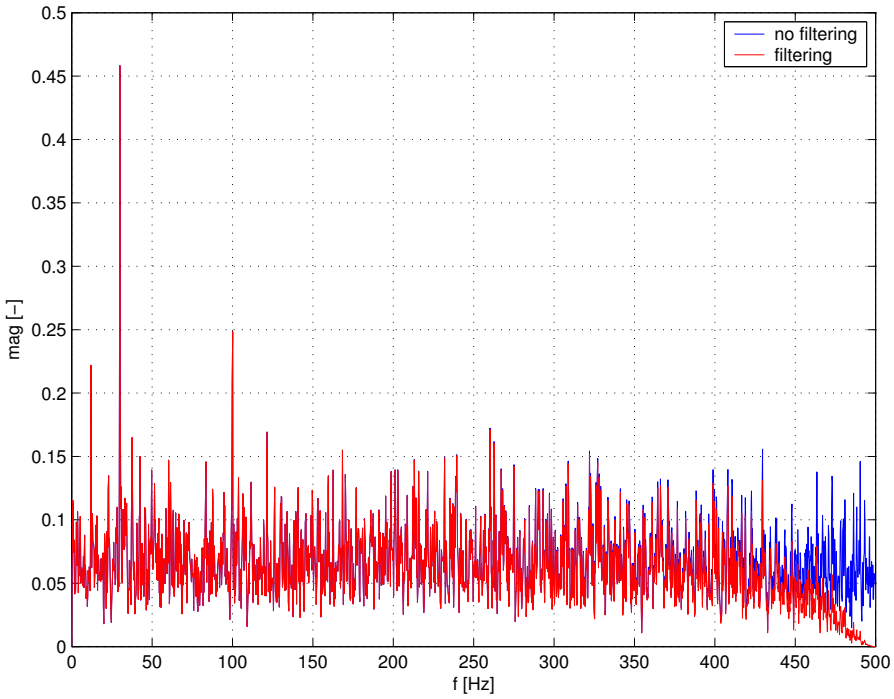

Figure 9: comparison of filtering & no filtering

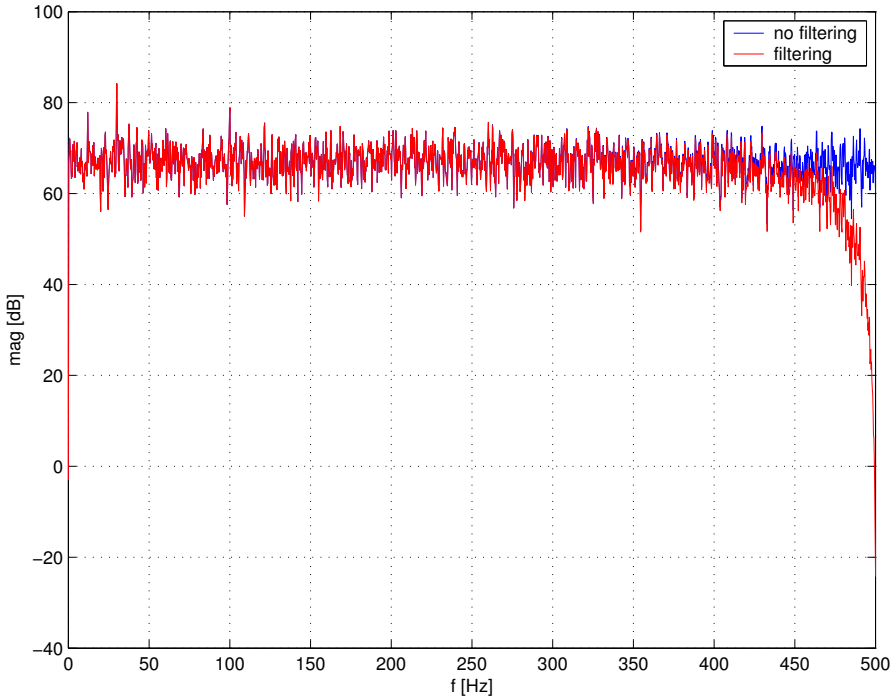

Figure 10: comparison of filtering & no filtering in the sound pressure plot

6.3 Window functions

There are 3 window functions in DFTAVG (option `-w`). The following example uses a curious but nonetheless a common case from electronics. Here there is no discrete frequency in the time signal, rather a continuous increase of the frequency, as it is coupled directly to the time. See Table 4. This effect is clearly visible in the plotted time signal in Figure 11.

```
clear
t = 0:0.001:45;
y = sin(2*pi*t.^2);
fid=fopen('testsignal2','w');
for i=1:size(t,2)
    fprintf(fid,'%12.7g %12.7g\n',t(i),y(i));
end
fclose(fid);
```

Table 4: `Matlab` program

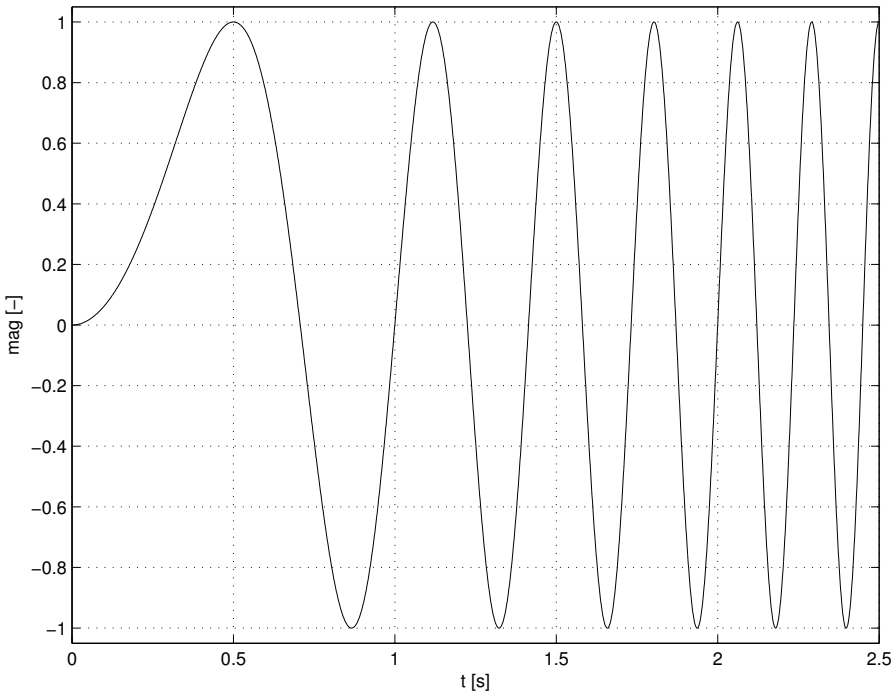

Figure 11: data with gliding frequency

First, the signal will be analyzed without window function (the same as Rectangle). Instead of using the total length of 45001 elements, only 32768 are used allowing an FFT. This case is named 'long FFT Rectangle', for reference:

```
>> dftavg -i testsignal2 -o exam3_w1j -w 1 -k 32768
```

The next step is to use averaging. with the same data length for comparability. The length of one averaging sub-run should be rather short and there should be a high degree of overlapping so that the results become smooth. This will be referred to as 'averaging Rectangle':

```
>> dftavg -i testsignal2 -o exam3_w1a -w 1 -l 2048 -s 100 -k 32768
```

The last two test use the same settings, just the window function is changed (option `-w`). The following four commands for the cases are used: 'long FFT Hanning', 'averaging Hanning', 'long FFT Triangle' & 'averaging Triangle':

```
>> dftavg -i testsignal2 -o exam3_w2j -w 2 -k 32768
>> dftavg -i testsignal2 -o exam3_w2a -w 2 -l 2048 -s 100 -k 32768
>> dftavg -i testsignal2 -o exam3_w3j -w 3 -k 32768
>> dftavg -i testsignal2 -o exam3_w3a -w 3 -l 2048 -s 100 -k 32768
```

The results are shown in Figure 12 & 13. Figure 12 shows the spectra with the Rectangle function. For the 'long FFT' case the result is not too bad. For low frequency the magnitude is constant and for high frequencies zero as it should be. Just around the jump one can see Gibb's phenomenon — strong oscillations. The averaged case is smooth, but the magnitudes are too high.

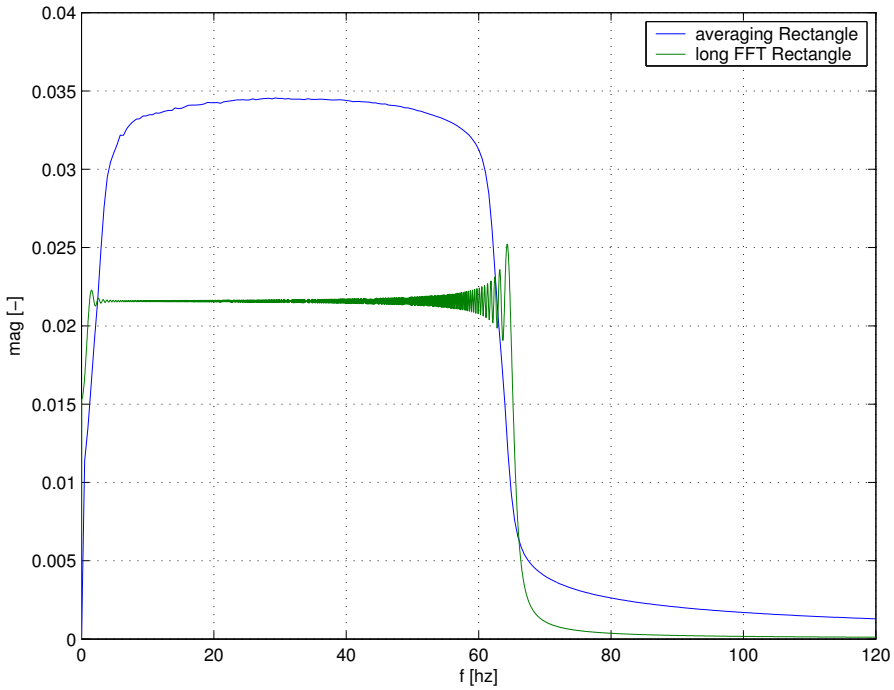

Figure 12: comparison of averaging & no averaging for the Rectangle window function

Figure 13 depicts the results for real window functions. For 'long FFT' the spectra have the same shape as the window functions. This effect comes about, because the included frequencies are not distributed over the whole data. The beginning of the data includes the low frequencies, whereas the higher frequencies are in the end of the data. If a window function is applied, then the amplitude of the data is reduced at the beginning and the end and the amplitudes of the low and high frequencies are damped. To prevent this effect, the length of one sub-run (`setl` option `-l`) must be rather short. And the level of overlapping must be proportionally high, the `shift` (option `-s`) must thus be small, so that there are many averagings. The good results obtained by this proceeding can be seen in Figure 13 in the cases 'averaging Hanning' & 'averaging Triangle'. The amplitude of the frequency band is constant, but there are still window effects for low and high frequencies, because the data that includes these frequencies can not be sufficiently averaged as they are at the beginning and end of the data array.

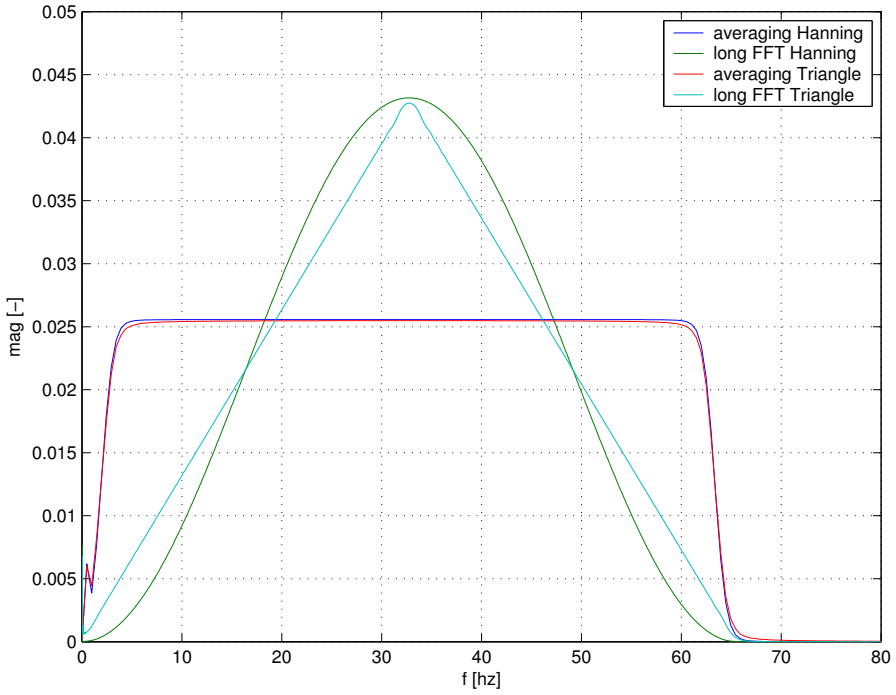

Figure 13: comparison of averaging & no averaging for the Hanning & Triangle window function

By comparing Figure 12 & 13 one can see that the Rectangle solution never really fits to the other window functions. This changes as one enables the variance exact computation (option `-v`). All other parameters are the same as previously. The results for all three window functions are in Figure 14. Now, they agree very well with each other.

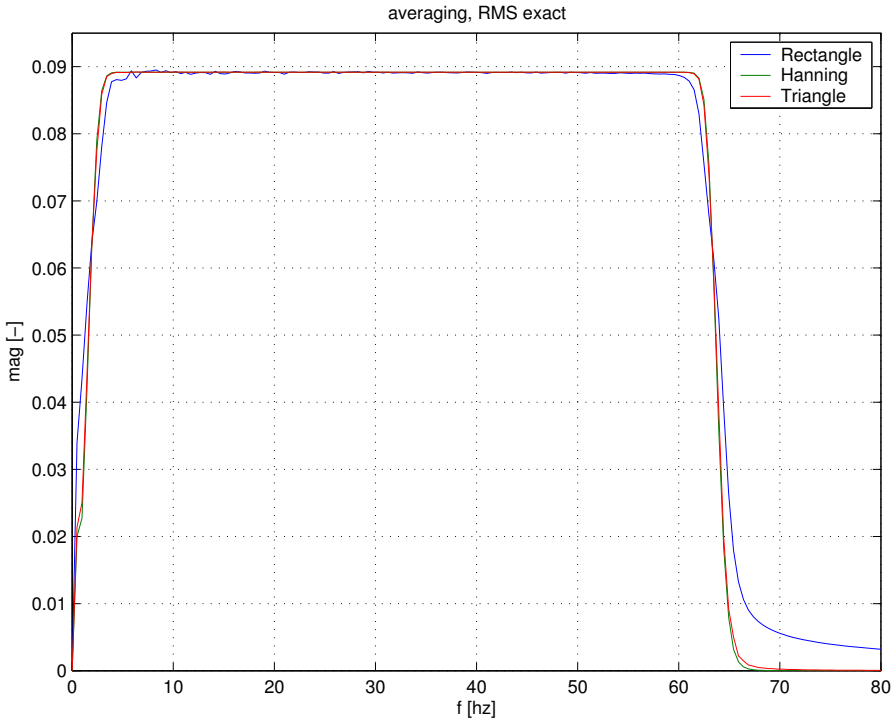

Figure 14: comparison of all three window function with variance (RMS)-exact calculation and averaging