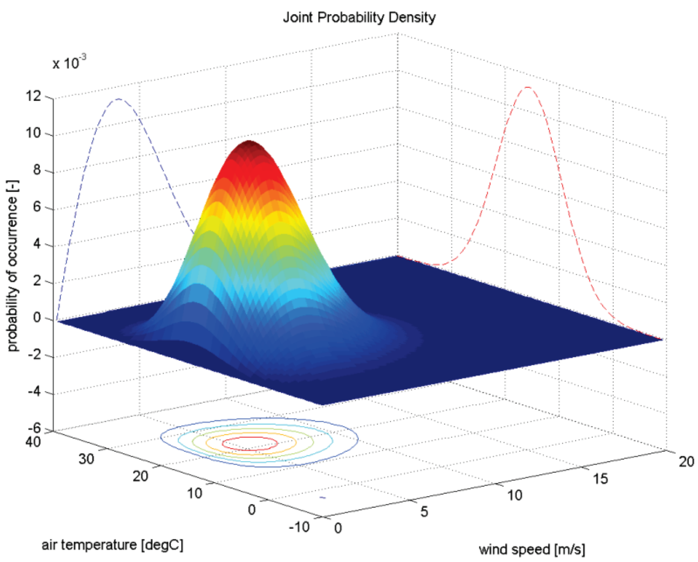
Data & Probability Analysis Tools



H. Holger Hundborg Koss

User's Manual

For Matlab scripts to analyse measured time series and to perform some probability analysis. Background material on probability analysis.

Department for Civil Engineering

2014 (Ver.2.1)

DTU Civil Engineering
Department of Civil Engineering

This page intentionally left blank.

DTU–Byg Education

Data & Probability Analysis Tools

User's Manual for Matlab scripts to analyse measured time series and to perform some probability analysis. Background material on probability analysis.

.

First Steps in Application

H. Holger Hundborg Koss

2014

Technical University of Denmark – DTU

Department of Civil Engineering

BYG – Structural Engineering

Editorial Note:

This document has been prepared as accompanying material to different courses on Master and PhD-level using or addressing data analysis and probabilistic

Technical University of Denmark – DTU
Department of Civil Engineering (BYG•DTU)
Section of Structural Engineering

LAST SAVED ON
23 October 2014

Table of Content

page intentionally left blank

1. Introduction

1.1 Background

The Matlab routines presented in this document have been developed for data analysis in connection with wind tunnel testing but can easily be applied on measurements of stochastic processes in general. As a consequence, we consider our data as time-dependent measurement signal or time histories in general. The scripts focus on different aspects in data analysis but also on probabilistic application of the results. Following scripts are described in the following chapters:

THA **Time History Analysis**
This script was developed to perform a first analysis on measured data from a wind tunnel test. A first check of the measurement quality is done by visual inspection, meaning that we just take a look at the raw data of the measurements. Furthermore, the probability density function of all data points is shown and compared to a Normal distribution density to indicate possible skewness of the ensemble of all data points. Additional functions like the calculation of the power spectral density, digital filtering and detrending, and application of sub-series allow for basic signal processing.

TSCorr **Time Series Correlation**
In case we would like to compare two processes, $X_A(t)$ and $X_B(t)$, occurring at the same time we often are interested in how parallel the fluctuations in the two processes are to each other. A common graphical method is to create a correlation plot where for each single time instant, t_i, the values $X_A(t_i)$ and XB(ti) are the coordinates in a Cartesian system. The resulting graph gives an image of the correlation between both time processes.

JPDF **Joint Probability Density Function**
Once the distribution density of a stochastic process is known we can use it to calculate probabilities for occurrence, exceedance or non-exceedance of a particular value. In this case we reduce the underlying stochastic time process to a stochastic variable characterised by it probability density. For events consisting of at least two variables we can calculate a joint probability density function (JPDF). This scripts illustrates the JPDF of two variables and allows calculating the probability of a specific case consisting of certain combinations of the two variables.

The THA-script uses a number of Matlab routines for signal processing: `detrend` for removing a linear trend in the recoded time history of the signal, `butter` and `filtfilt` for digital filtering of the signal (high-pass, low-pass and band-pass filters) and `pwelch` to calculate the power spectral density of the signal. These routines are described (Matlab documentation files) in chapter 4.

More detailed information on the different scripts is given in the following chapters.

1.2 On Measuring and Sampling of Time Series

In wind engineering and in structural dynamics the analysis of the time-dependent load and the resulting response is a prerequisite to understand the nature of the processes behind. For this purpose load and response are measured as time series. The variation of the signal in time is usually of random nature and can be described by statistical parameters and properties such as mean value, standard deviation or probability distribution. An example of a measured random time process X(t) is shown in Figure 1.1 below:

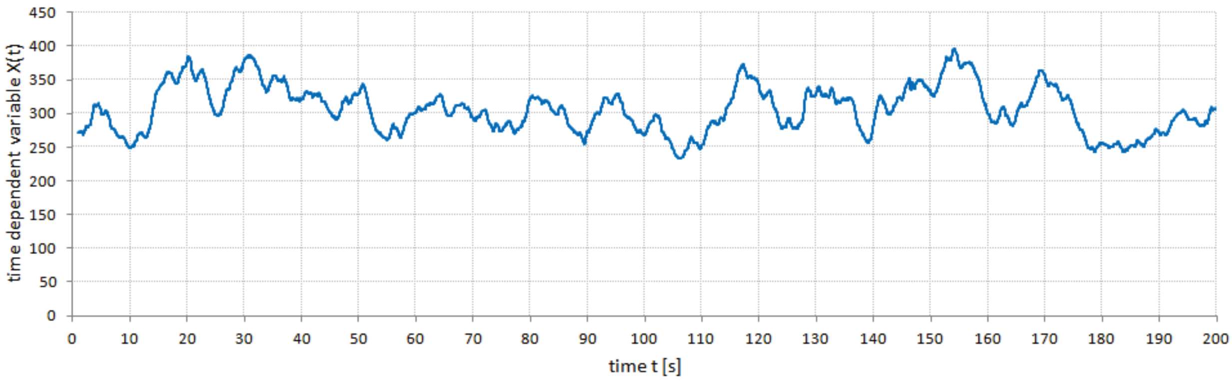
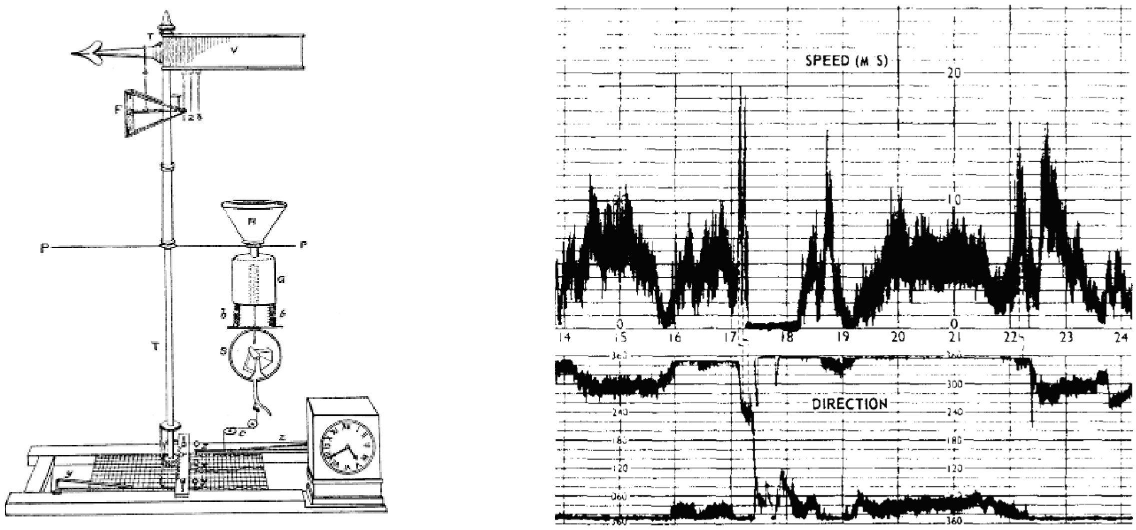


Figure 1.1 Example of a measured time series. The variation of the signal in time is usually of random nature and can be described through statistical parameters and probability distribution.

Before we start analysing measured data we need to discuss the implications related to measurement and data sampling. In a first step let's assume that our measured time series in Figure 1.1 is the result from an analogue measurement. This means that the graph is continuous and contains at any instant of time the information of the measured phenomenon. An example of such analogue measurement is illustrated in Figure 1.2.



Principle of Osler's self-registering pressure plate anemometer, 1837. The instrument is shown with a tipping-bucket rain gauge. (From Abbe, 1888, reported in Multhauf, 1961).

Anemograph trace from East Sale for 26 November 1978, showing the variable width of the direction trace (Moriarty, 1985). The graphs is written as continuous lines on paper, displaying hence the measurement results with infinite density.

Figure 1.2 Example of an analogue measurement of wind speed and direction.

The continuous writing of the measurement result on a paper roll gives an absolute analogue image of the measured phenomenon. At no point in time the registration is missing a part of the signal, which means that the information of signal is available with infinite density.

This relation changes when using electronic systems to measure a continuous phenomenon in nature. The measurement system consists in general of several components, which in their combination can be considered as the *acquisition chain* as shown in Figure 1.3. While passing through the different components or stages of the acquisition chain the signal gets modified and finally converted into discrete numerical values that later can be used for computerised data analysis.

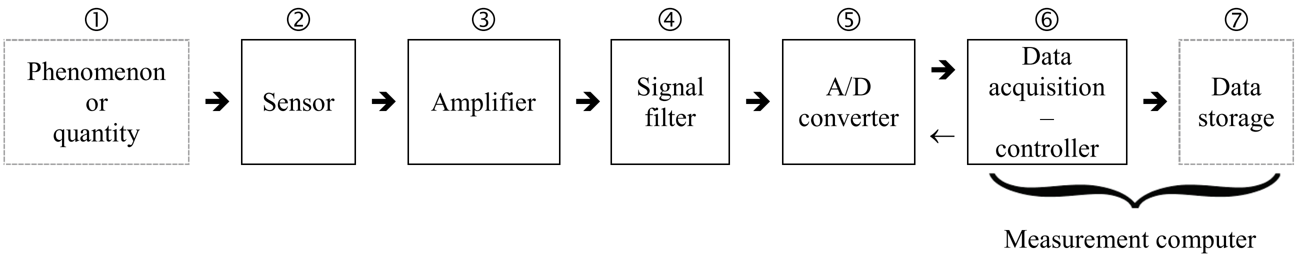


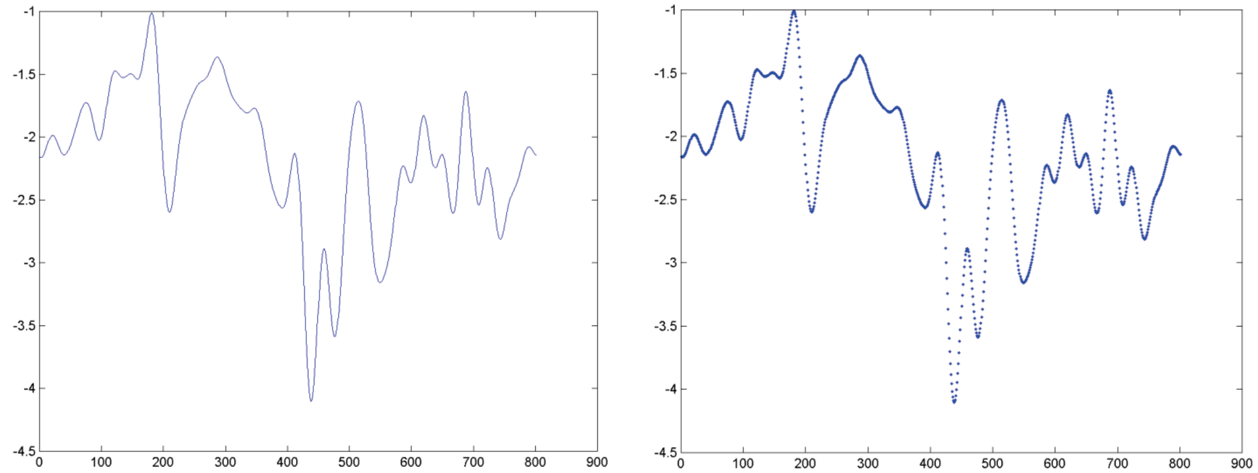
Figure 1.3 Simplified structure of a measurement or data acquisition chain with main elements.

Data Acquisition Chain

It starts with the phenomenon ① we would like to investigate and for which we need a sufficient amount of data. For the investigation we need to consider which quantity we need to measure. For example for studying the wind we would usually be interested in the airspeed, but also flow direction and air temperature could be of interest. Each of which are different physical quantities requiring different types of sensors. In our acquisition chain the sensor ② is now registering the quantity and provides the reading as an electric signal, for example as a varying output voltage. Depending on the sensor the voltage signal can be quite small and is often amplified ③ to a magnitude the following components of the acquisition chain can better work with.

In the next step the signal gets filtered ④ for several reasons: one reason is the removal of effects from the signal that are not part of initial sensor reading such as electronic noise (which is unavoidable when using electronic equipment) or other disturbing influences. Secondly, the filtering shall remove high-frequent components in the signal that would bias the analysis in frequency domain (*aliasing* effect). Filtering of the (still!) analogue signal at this stage of the acquisition chain is in particular important since filtering of a digitalised signal requires a much higher time resolution of the signal. This can in some cases exceed the capacity of the acquisition equipment. The filtering can also be considered as one form of *signal conditioning*.

After the filter we should have an electric signal that to the best extend reflects the magnitude and variation of our physical quantity at the beginning ① of the chain. This continuous or analogue signal will now be transferred into numerical values. The accuracy with which the signal can be converted depends on with how many different but discrete values the analogue signal can be described. This step ⑤ transforms or converts the analogue signal into digital numeral system (A/D conversion). This conversion affects not only the resolution of the signal magnitude but also its resolution in time: the density of data points over time is depending on the sample frequency (*signal sampling*) and the resolution of the signal depends on the quantisation and coding (*signal digitalisation*). To preserve the information regarding the investigated phenomenon all components of the acquisition chain including sampling frequency have to be chosen carefully.



Continuous analogue signal. Sampled and digitalised discrete signal.

Figure 1.4 Illustration of the difference between an analogue and a discrete signal.

In the following some of the aforementioned terms in connection with data acquisition are discussed in more detail:

- **Sampling:** the continuous process in nature is registered at discrete moments in time, which leads to an image of the process consisting of individual points instead of a continuous curve. This step is usually referred to as conversion from analogue to digital information (A/D conversion). The density of the data points in time, the time step Δt, depends on the sample frequency $f_{samp} = 1/\Delta t$. It goes without saying that the smaller Δt the better the image of the process in nature (*resolution in time*).

- **Signal resolution:** the values of our time process vary within a certain range. When sampling the process at discrete moments in time the measurement instrument "reads" the values with a certain "sharpness". The sharpness of the reading results from the combination of two factors: the range in which the instrument operates reliable (instrument measurement range) and the quality of the digitalisation process. The latter defines the number of steps the measurement range can be described with. Using a binary numeral system the number of steps is calculated with the number of bits available for the A/D conversion. The bit-number stands for the word size that can be formed based on the elementary information of I and 0. For example, an anemometer can read velocities between 0 and 50m/s (= measurement range). Using a 16-bit conversion we have 2^{16} = 65,536 numerical words available to resolve the measurement range. Hence, the resulting resolution of the signal reading (sharpness) is 50/65,536 = 0.00076m/s.

- **Record length:** in case of a time series the record length is usually equal to the time duration. It can as well refer to the number of data points in the recorded time series. The latter becomes relevant for the algorithm of the Fast Fourier Transformation in the calculation of the power spectral density (here: `pwelch`).

- **Number of records:** usually each recording is of finite length. To achieve a better statistical stability in the analysis the observed phenomenon may be recorded several times. This can be difficult for full-scale observations but relatively easy to obtain from laboratory experiments such as wind tunnel tests.

1.3 Getting started

To use the here described Matlab scripts you need a licensed version of Matlab5.1 or newer including the signal processing toolbox. Copy the scripts from chapter 5 into a text editor program like "notepad" or "WordPad" from the Microsoft Office Accessories and save them unformatted with the corresponding name. Don't forget the ".m" extension. Alternatively you start the Matlab editor and copy the scrip into a new document.

Remember to define the location of your working directory in the Matlab command window. For example with the 'change directory' command, `cd`, as sown in the example below:

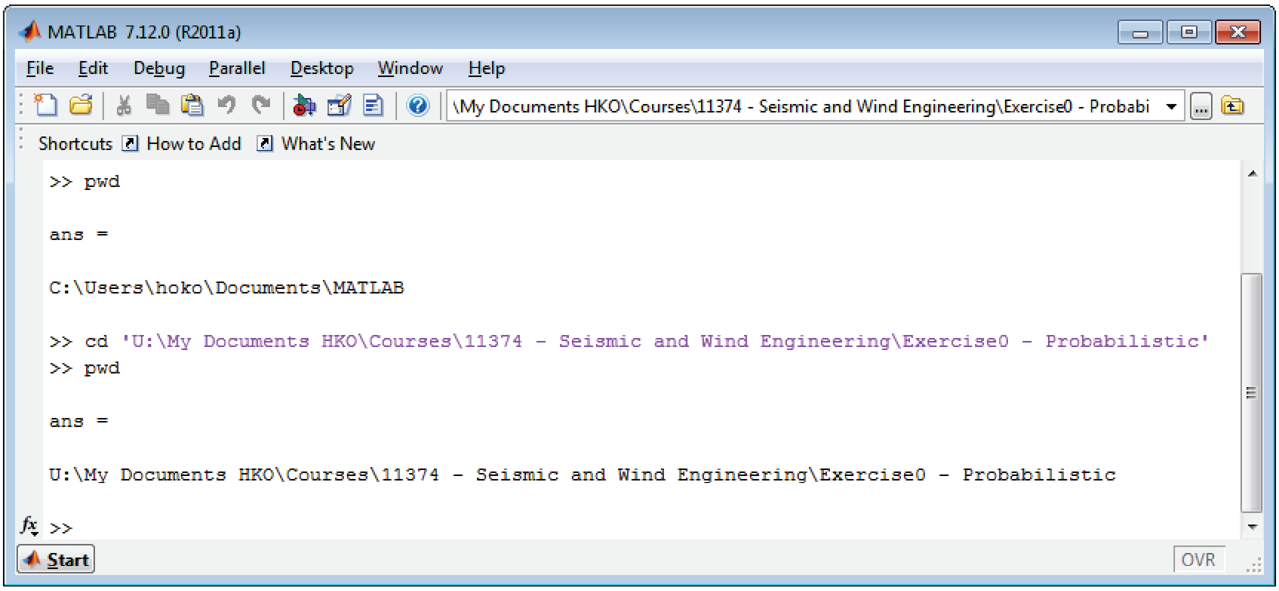


Figure 1.5 Matlab command window and procedure to change working directory.

The software package is introduced in the course 11374 "Seismic and Wind Engineering" and "Introduction to Wind Tunnel Testing in Civil Engineering" and is designed to read certain data files (see chapter 2.2.2) for exercise purpose. The scripts can of course be adopted to read any kind of data format.

2. Matlab Scripts

2.1 Definitions

Nomenclature

ΔT	Time step width in time axis of signal
f_{rel}	Relative frequency
f_{samp}	Sample frequency (= $1/\Delta T$)
n_i	Count of data points per bin in histogram. The bin width, Δx, is calculated by dividing the maximum data range (minimum to maximum) by the intended number of bins N_{bin}.
N	Number of data points in the measured signal time history.
N_{bin}	Number of bins equally distributed of the data range, hence defining the bin width Δx.
nfft	Non-uniform Fast Fourier Transform
PDF	Probability Density Function

Relative Frequency

$$f_{rel} = \frac{n_i}{\Delta x \cdot N}$$

Population
A statistical population is a set of entities (data points) concerning which statistical inferences are to be drawn, often based on a random sample taken from the population. Population is also used to refer to a set of potential measurements or values, including not only cases actually observed but those that are potentially observable.

Sample and Parent Population
A parent population is usually understood as a sample (measurement) of a phenomenon where the number of data points or observation goes to ∞. This is important because the parent population tells us the exact distribution of the data points. Any sample of limited length can only reflect the nature of the phenomenon with some uncertainty or error. This, in turn, gives us the chance to examine the error associated with making measurements. If the number of samples is high enough the mean value and standard deviation of the parent population is well reflected by the measurement. Estimation of other parameters such as extreme values is still affected by the limitation of a sample population.
Since in practice a population of infinite length will be difficult to obtain the term *parent population* is often also applied on very large sample populations. This becomes important when working with sub-series to emphasize the relation between short and long sample spaces.

2.2 Time History Analysis - THA

2.2.1 General Information

The script has been developed to get first information on the characteristic of a measure signal time history and to perform some basic signal conditioning. Figure 1.4 shows an example of the screen surface with different graph windows created when running `THA.m`.

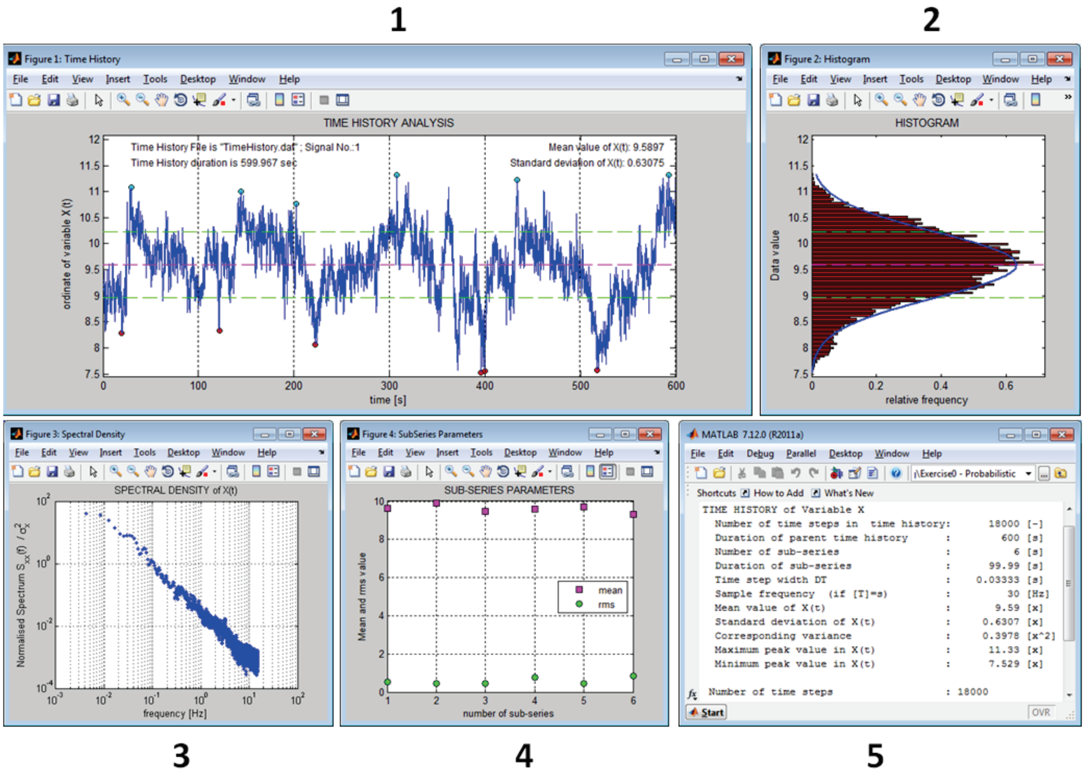


Figure 2.1 Display with different windows created when running "`THA5.m`".

Below, purpose and content of the different windows are briefly described:

1. **Figure 1: Time History**
   Plot of the time history of the measured signal, usually starting with the untreated raw data to get a first visual assessment of the data set quality. The electronic acquisition chain can add noise, spikes or trends to the actual signal, which before further analysis needs to be removed. The mean value and standard deviation of all data points are plotted on the graph. In case sub-series are defined and the maximum and minimum values shall be identified the corresponding information is shown on graph as well.
   In case of digital filtering the original and modified time series can be plotted in the same graph to control the effect of the filtering.

2. **Figure 2: Histogram**
   A further assessment of the variation characteristic of the data points is provided by the histogram of all data points. The individual bin count n_i is converted to relative frequency, whereby the area of the histogram becomes unity and is hence interpretable as the probability density function (or *discrete* PDF since defined in bins) of all data points. The curve of a normal distribution is plotted over the histogram to visualize possible skewness and kurtosis of the PDF. Mean and standard deviation are indicated. Not applicable on sub-series. In case of digital filtering the histogram is calculated from the modified time series.

3. **Figure 3: Spectral Density**
   The power spectral density of the entire measured signal time history is calculated using the `pwelch` Matlab routine and plotted – usually – in a double-logarithmic graph. The area underneath the spectrum is normalized with the variance and is hence unity.
   In case of digital filtering the spectrum of both original and modified time series can be plotted on the same graph to control the effect of the filtering.

4. **Figure 4:SubSeries Parameter**
   In case sub-series have been defined the mean value and standard deviation of each sub-series is shown on the graph. For better comparison the level of mean and standard deviation of the entire time series are plotted on the graph (not shown in Figure 1.4).
   In case of digital filtering the evaluation of the sub-series is applied on the modified time series.

5. **Matlab Command Window**
   Echo print of main information on time series and analysis. An example of the echo print is given below:

```
Data read from file:TimeHistory.dat
TIME HISTORY of Variable X
  Number of time steps in time history :      18000 [-]
  Duration of parent time history      :     599.97 [s]
  Number of sub-series                 :         10 [s]
  Duration of sub-series               :      60.00 [s]
  Time step width DT                   :    0.03333 [s]
  Sample frequency (if [T]=s)          :         30 [Hz]
  Mean value of X(t)                   :      9.589 [x]
  Standard deviation of X(t)           :     0.6094 [x]
  Corresponding variance               :     0.3714 [x^2]
  Maximum peak value in X(t)           :      11.01 [x]
  Minimum peak value in X(t)           :      7.803 [x]

SPECTRAL DENSITY parameters for Sxx
 Number of overlapping sub-windows     :          8 [-]
 Sub-window length                     :       4000 [-]
 Filter depth of fft-routine           :       3072 [-]
```

2.2.2 Input Data Format

In principle any type of data set can be used. Only requirement is that the time series is properly loaded and all required information is defined (see note in script line 349 to 368). Following information is required:

```
SignalNo = Number of signal that has been chosen to be analysed - saved as X0(i)
X0(i)    = Vector with data of stochastic process (length = m1)
TAx(i)   = Vector with values for time axis (length = m1)
m1       = number of data points (time steps)
NFFTcase = FFT filter length, determine by trial, shall not exceed m1
Nwindow  = Number of windows (default = 8) for pwelch SFD calculation
Fsamp    = sample frequency in [Hz]
DT       = time step between data points [s] = 1/Fsamp
Nbin     = Number of bins to generate histogram
Nsub     = Number of sub-series in which the signal can be divided to calculate sub-mean
           and rms-values
```

The example data sets are of different format and structure and are in following briefly described.

TimeHistory.dat
`Iread = 1`

Ascii file containing in the first column the time axis and in the second column the time history of the investigated quantity (not further specified).

Time t [s]	X(t) [-]
0	9.17406
0.033333	9.12009
0.066667	9.17764
0.1	9.20783
0.133333	9.09194
0.166667	9.16704
0.2	9.1348
0.233333	9.1484
0.266667	9.28838
0.3	9.27324
…	…
…	

The Matlab syntax for reading the data from the input file is shown below (only command lines for reading and storing data):

```
FileName = 'TimeHistory.dat';          % Name of input file
fid      = fopen(FileName,'r');
Series   = fscanf(fid,'%e',[2 inf]);   % 2-column matrix with time axis (col.1) and time series (col.2)
Series   = Series';                    % Transposed matrix
[m1 n1]  = size(Series);               % Number of rows (m1) and columns (n1)
status   = fclose(fid);

TAx      = Series(:,1);                % Time series
X0       = Series(:,2);                % Saving selected data to variable vector
DT       = (TAx(10)-TAx(1))/9;         % Calculation of time step
```

The data file "BendTS.txt" is of similar structure. Time series contains the base bending moment [N] of a high-rise building.

18Signals.dat

`Iread = 2`

File with 18 time series of pressure coefficients measured in a wind tunnel test on a model low-rise building (based on cpcent.00). At the top of the data set of 18 individual signals (pressure coefficients along the centre bay of a low-rise building) the mean velocity at building's eaves height (model scale) is given in [kPa]. The data set has <u>no</u> time axis! To plot the signal correctly and to calculate the spectral density the sample frequency has to be defined separately.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0.035583																	
0.574160	0.544350	0.530770	0.116290	-1.415100	-1.068400	-1.200500	-0.641010	-0.414110	-0.459180	-0.451410	-0.404180	-0.351740	-0.189850	-0.196810	-0.234350	-0.234600	-0.238150
0.616210	0.594510	0.564020	0.126780	-1.428300	-0.917270	-1.105800	-0.711080	-0.413680	-0.405490	-0.416620	-0.366270	-0.334470	-0.157090	-0.163440	-0.216980	-0.217250	-0.216550
0.652350	0.638380	0.598000	0.157840	-1.437900	-0.881480	-0.970450	-0.803260	-0.428030	-0.361010	-0.375270	-0.334100	-0.301430	-0.160100	-0.145670	-0.204300	-0.205730	-0.198890
0.672520	0.659770	0.626500	0.193560	-1.445200	-0.977970	-0.837730	-0.892350	-0.469890	-0.343980	-0.334800	-0.318770	-0.269280	-0.197260	-0.150890	-0.206270	-0.208330	-0.194820
0.674170	0.654250	0.646720	0.219210	-1.452600	-1.174600	-0.758210	-0.946190	-0.542190	-0.366670	-0.302900	-0.319910	-0.254730	-0.251160	-0.176840	-0.225100	-0.226330	-0.207160
0.662180	0.630750	0.657710	0.228310	-1.460300	-1.403900	-0.767810	-0.939800	-0.633170	-0.430850	-0.287240	-0.326780	-0.265150	-0.297320	-0.212490	-0.254150	-0.253110	-0.230780
0.646090	0.606190	0.661100	0.225310	-1.469300	-1.592300	-0.871410	-0.866380	-0.718970	-0.526040	-0.293220	-0.324780	-0.293470	-0.316060	-0.243080	-0.281080	-0.277590	-0.255300
0.635310	0.596550	0.660620	0.221930	-1.478100	-1.688700	-1.038100	-0.743420	-0.770970	-0.629920	-0.323180	-0.303240	-0.319580	-0.300820	-0.256530	-0.293700	-0.289320	-0.269820
...	...	...	...														
...	...	...															
...	...																

Since the data structure is slightly irregular with a single value of the velocity pressure leading the actual data set with 18 columns, the reading syntax is modified accordingly:

```
FileName = '18Signals.dat';            % Name of input file
fid      = fopen(FileName,'r');        % Echo print on screen
qhmwk    = fscanf(fid,'%e',[1 1]);     % velocity pressure [kN/m^2]
cp       = fscanf(fid,'%e',[18 inf]);  % Matrix with pressure coefficient time series
Series   = cp';                        % Transposed matrix where each column is a signal
[m1 n1]  = size(Series);               % Number of rows (m1) and columns (n1)
status   = fclose(fid);

SignalNo = 12;                         % Number of signal to be analysed (1-18)
X0       = Series(:,SignalNo);         % Saving selected data to input vector

Fsamp    = 1600;                       % Sample frequency in
DT       = 1/Fsamp;                    % Calculation of time step

for i=1:m1                             % Generation of time axis with m1 steps
    TAx(i) = (i-1)*DT;
end
```

Here, the signal number, `SignalNo`, marks the column of the data file, which for the analysis in this script is copied to the time series vector `X0`. Since the data file does not contain an explicit time axis, the corresponding time-step values are generated based on the time step length `DT`.

`Iread = 5`

For the calculation of the dynamic non-linear response of the steel frame supporting structure the data have been organized alternatively in 12 blocks each lead by the mean velocity pressure applicable on the subsequent data set. This fragmented format of the measured wind load process is contained in file "**cpcent.00**" (family of 100 data sets *cpcent.00* to *cpcent.99*). This data format has been chosen within the "BEATRICE Joint Project: Wind Action on low-rise buildings" and is hence included in the THA5.m script for research purpose. The Matlab syntax for reading the data from the input file is shown overleaf (only command lines for reading and storing data):

```
FileName = 'cpcent.00';              % Name of input file
Nstorm   = 12;                       % number of sub-series
Ntap     = 18;                       % No of taps
length   = 4096;                     % number of time steps per storm

Series   = zeros(Nstorm*length,Ntap); % pre-allocation of space for fast data handling
fid      = fopen(FileName,'r');       % Echo print on screen (data file reading number)

index=0;
for istorm = 1:12                                 % Loop over all 12 data bocks
    f1              = (istorm-1)*length+1;         % block start address in data combination on Series
    f2              = istorm*length;               % block end address in data combination on Series
    qhmwk(istorm)   = fscanf(fid,'%e',[1 1]);      % velocity pressure [kN/m^2]
    cp              = fscanf(fid,'%e',[18 length]); % data block with 18 columns (= data time series)
    Series(f1:f2,:) = cp';                         % saving the 12 data sets as continuous times series
    fprintf(1,'Storm Number considered: %g %g\n',istorm,qhmwk(istorm))
end

status   = fclose(fid);
[m1 n1]  = size(Series);             % Number of rows (m1) and columns (n1)

SignalNo = 3;                        % Number of signal to be analysed (1-18)
X0       = Series(:,SignalNo);       % Saving selected data to input vector
Fsamp    = 1600;                     % Sample frequency in
DT       = 1/Fsamp;                  % Calculation of time step

for i=1:m1                           % Generation of time axis with m1 steps
    TAx(i) = (i-1)*DT;
end
```

For proper reading the length of each block (storm event), `length`, and the number of all storm events, `Nstorm`, contained in the data file needs to be pre-defined. Hereafter, the procedure of reading one value for velocity pressure followed by a specific data set with 18 time series of wind load processes given as pressure coefficients (18 columns with 4096 data points each) is repeated for each block or storm – 12 times in total.

The time series from the 12 blocks are saved as continuous time series, similar to the format **18Signals.dat** is already given in. There is no "jump" where the series or blocks meet since the wind loads were actually measured as continuous time series but for the application in the dynamic analysis artificially split up into 12 sub-events. The re-connection to a continuous series allows for subdivisions other than 12 blocks to sample maximum and minimum values for extreme value statistic.

Similar to reading *18Signals.dat*, the signal number, `SignalNo`, marks the column in the data file, which for the analysis in this script is copied to the time series vector `X0`. The data file does not contain an explicit time axis, hence the corresponding time-step values are generated based on the time step length `DT`.

Reykjavik2002.txt
`Iread = 3`

The data are provided by a private weather station located in the harbour of Reykjavik. The file contains amongst other 10 minutes mean and gust wind speeds continuously recorded throughout the year 2002. The file format is given in the table below. To plot the values of column 6 to 10 as a time history the time information (columns 1 to 5) need to be converted into a more convenient format.

1	2	3	4	5	6	7	8	9	10
Hours	Minutes	Day	Month	Year	mean wind speed	gust wind speed	wind dir.	RMS of wind dir.	atm. press.
[h]	[min]	[d]	[month]	[a]	[m/s]	[m/s]	[dir]	[deg]	[mbar]
0	0	1	1	2002	4.9	8.1	214	13.4	1008.5
0	10	1	1	2002	4.7	7.8	211	14.4	1008.6
0	20	1	1	2002	4	8.3	208	15.3	1008.6
0	30	1	1	2002	2.9	6.9	205	20.3	1008.6
0	40	1	1	2002	3.1	5.6	206	17.8	1008.7
0	50	1	1	2002	2.5	5.5	203	19.1	1008.7
1	0	1	1	2002	2.9	5.6	204	19.8	1008.7
1	10	1	1	2002	2.9	5.4	205	16.7	1008.8
1	20	1	1	2002	2.5	4.3	207	18.2	1008.8
1	30	1	1	2002	2.3	4.2	201	19.8	1008.6
…	…	…	…	…	…	…	…	…	…
…	…	…							
…									

Below, the reading sequence of that data file is given. Since the measured wind speed values are defined as 10-minutes mean values the sample frequency is calculated for data points 10 minutes or 600 seconds apart: f_{samp} = 1/600 = 0.001667Hz. At the end, the time information is converted into a continuous time axis given in minutes.

```
FileName = 'Reykjavik2002.txt';         % Name of the 1st time history file
fid1     = fopen(FileName,'r');
Series   = fscanf(fid1,'%g',[11 inf]);
Series   = Series';
[m1 n1]  = size(Series);
status   = fclose(fid1);

SignalNo = 6;                           % Number of signal to be analysed (1-18)
X0       = Series(:,SignalNo);          % Saving selected data to input vector

NFFTcase = 10*1024;                     % Filter depth of the fft-routine N*1024
Nwindow  = 8;                           % Number of windows (default = 8) for pwelch SFD calculation
Fsamp    = 0.001667;                    % Sample frequency in [Hz]
DT       = 600;                         % 10min mean data
Nbin     = 50;                          % Number of bins to generate histogram
Nsub     = 12;                          % Number of sub-series

% Calculating a continous time index:
% ------------------------------------
% (time index is generated in 10-minutes steps as the smallest time unit available)

DayMon    = [31 28 31 30 31 30 31 31 30 31 30 31]; % days per month for time axis
for i=1:m1
    if Series(i,4)==1;  Month=0     ; end
    if Series(i,4)==2;  Month=44640 ; end
    if Series(i,4)==3;  Month=84960 ; end
    if Series(i,4)==4;  Month=129600; end
    if Series(i,4)==5;  Month=172800; end
    if Series(i,4)==6;  Month=217440; end
    if Series(i,4)==7;  Month=260640; end
    if Series(i,4)==8;  Month=305280; end
    if Series(i,4)==9;  Month=349920; end
    if Series(i,4)==10; Month=393120; end
    if Series(i,4)==11; Month=437760; end
    if Series(i,4)==12; Month=480960; end
    TAx(i) = Series(i,1)*60+Series(i,2)+(Series(i,3)-1)*24*60+Month;  % time axis in minutes
end
```

Other Data File Formats

In many cases data from measurements are stored in files with text headers documenting the test configuration and signal parameters. This information is of importance when discussing and comparing the obtained results to other studies or for reconstructing the test situation in case of additional studies or variations of a particular case. For numerical analysis of the data time series the headers need to be considered for the reading process. Below, an example is given on how to read text lines in a quite simple way. Assumed we have an ascii-formatted data file of following structure:

```
Block size: 32
Sample rate: 2048

Time      Fx1        Fy1        Fz1       Mx1       My1       Mz1        ExcA
0.0000    -0.0360    0.0313     2.4076    6.1196    4.6910    -1.8326    9.9875
0.0005    -0.0412    0.0303     2.4050    6.1193    4.6854    -1.8339    9.9875
0.0010    -0.0399    0.0320     2.4050    6.1183    4.6887    -1.8286    9.9878
0.0015    -0.0386    0.0313     2.4040    6.1156    4.6818    -1.8316    9.9878
…         …          …
…         …
…
```

The above given data file can be read with following script:

```
fid     = fopen(input1,'r');
Block   = fscanf(fid, '%*s %*s %d\n', 1);
Fsamp   = fscanf(fid, '%*s %*s %d\n\n', 1);
Dummy   = fscanf(fid, '%*s %*s %*s %*s %*s %*s %*s %*s\n', 8);
Serie1  = fscanf(fid,'%g',[8 inf]);
status  = fclose(fid);
Serie   = Serie1';
```

Here, "`input1`" is the name of the data file. "`Block`" is the numerical variable to which the block size 32 will be assigned and "`Fsamp`" is the variable for the sample frequency of 2048Hz. The actual words such as "`Block`" and "`size:`" are read as individual strings "`%*s`" of unknown length but separated by spaces. Line break is marked with "`\n`"

2.2.3 Analysis Control Settings

The usage of the THA-script is in essence controlled through parameter settings. We distinguish between two types of parameters: *values parameters* like for example `Nbin` the number of bins applied for the generation of the histogram and *control parameters* activating or switching actions on or off like detrending or digital filtering. The parameters of the THA-script are:

Value Parameters

In sequence of appearance. Parameters 2 to 13 are defined specifically for each data files since the format changes and for some the time axis needs to be generated. Parameters 14 to 21 are general parameters concerning histogram, spectral density and digital filtering.

#	parameter	script line	description
1	`Iread`	66	Integer value indicating which data file (format) should be read
2	`FileName`	*	Name of data file
3	`NFFTcase`	*	Filter length of FFT routine for specific data set
4	`Series`	*	Internal array onto which the values from data file are saved
5	`m1`	*	Lines of Series = number of time steps
6	`n1`	*	Columns of Series (number of signals)
7	`TAx`	*	Values of time axis (length = m1)
8	`X0`	*	Vector containing the unmodified data of the signal to be analysed
9	`DT`	*	Time step (=1/fsamp)
10	`Fsamp`	*	Sample frequency (=1/DT)
11	`SignalNo`	*	Number of the signal (column in Series)
12	`Nbin`	*	Number of bins for histogram
13	`Nsub`	*	Number of sub-series the time series in X0 shall be divided in
14	`Nw`	431	Number of 50% overlapping windows for `pwelch` routine
15	`window`	432	FFT window length for `pwelch` (calculated)
16	`nfft`	433	FFT length parameter in `pwelch` copied from NFFTcase
17	`fs`	434	Sample frequency in `pwelch` copied from Fsamp
18	`Fn`	445	Order of digital filter (using standard 6^{th}- order Butterworth filter)
19	`Ftype`	446	Filter type (high- or low-pass filter)
20	`CutOff`	447	Cut-off frequency for digital filter
21	`X1`	463	Vector with modified data after detrending (even though if not applied)
21	`X2`	484	Vector with modified data after filtering (even though if not applied)

* parameter defined in each input sequence individually

Control Parameters

Parameters 1 to 4 are switches turning a certain action on <1> or off <0>

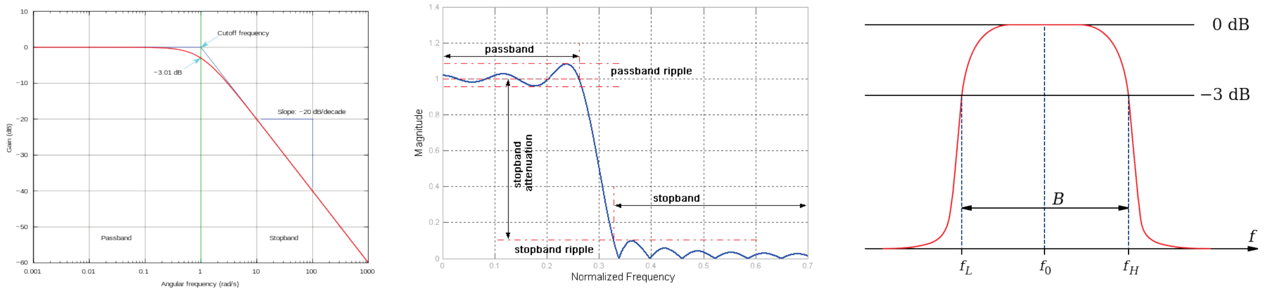
#	parameter	script line	description
1	`DoAct1`	387	Linear detrending the time series (no break points)
2	`DoAct2`	388	Identify, display and safe sub-series maxima and minima in vector
3	`DoAct3`	389	Digital filtering
4	`DoAct4`	390	Saving modified data in external file "Data2.dat"
5	`Displ`	396	Display parameter: 1 = resulting data, 2 = both initial and modified data

Digital Filtering

With digital filtering we can reduce or even eliminate the contribution of a certain frequency range to the characteristic of the investigated signal. A filter is in principle a transfer function defined in frequency domain assuming values of either “1” or “0”. Where the transfer function is “1” the corresponding frequencies remain unchanged in the signal and where the function is “0” the corresponding frequencies will be removed from the signal. The point where the two states of the transfer or filter function meet is called “cutoff” frequency. We usually distinct between three different types of filters:

- Low-pass filter: all frequencies below cut-off frequency pass through unchanged
- High-pass filter: all frequencies above cut-off frequency pass through unchanged
- Band-pass filter: all frequencies between two cut-off frequencies pass unchanged

In reality the filter function is not sharp and rectangular. The transition between “1” and “0” at cut-off frequency is inclined to avoid numerical instabilities in the filter algorithm. If the inclination becomes quite steep the filter function begins to oscillate near the cut-off frequency affecting the amplitudes of the corresponding frequencies in the filtered signal (Figure 2.2c).



a) Transfer function of a low-pass filter.

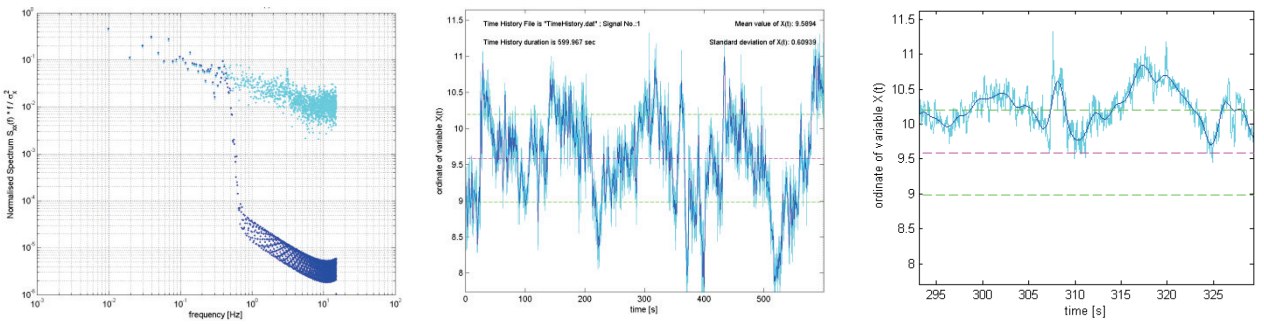
b) Low-pass filter magnitude response.

c) Principle transfer function of band-pass filter.

Figure 2.2 Illustration of filter function in digital filtering.

Below, a low-pass filter has been applied on the “TimeHistory.dat” with following filter parameter settings:

```
Fn     = 6;
Ftype  ='low';
CutOff = 0.5;
```



Spectral density of filtered and unfiltered signal.

Time history of filtered and unfiltered signal.

Zoom of time history showing the filtered signal over the unfiltered.

Figure 2.3 Illustration of filter function in digital filtering.

Use parameter `displ` to switch the view of filtered over unfiltered signal on or off.

Sub-Series

One way to collect data on the occurring extreme values is to divide the measured time series into sub-series of equal length. To estimate how statistically similar the different sub-series are to each other (theorem of ergodicity) the mean value and standard deviation of each sub-set is calculated and displayed on the graph in Figure 2.4. The dashed lines represent the mean and standard deviation of the whole time series.

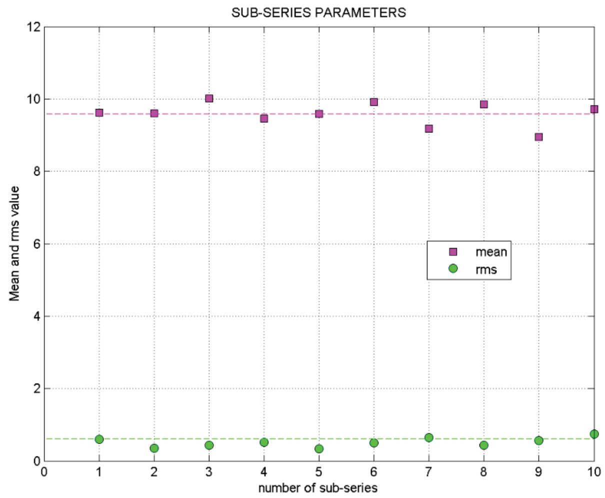


Figure 2.4 Comparison of mean values and standard deviation from each sub-series to estimate their statistical similarity to each other (ergodicity).

In order to proof or assess ergodicity of the sub-series more information than just the similarity of mean value and standard deviation is required. In principle similarity shall be proven on the probability density functions of all sub-series to the parent series including the four moments of the pdf: mean value, variance, skewness and excess kurtosis.

Histogram and Probability Density Function (PDF)

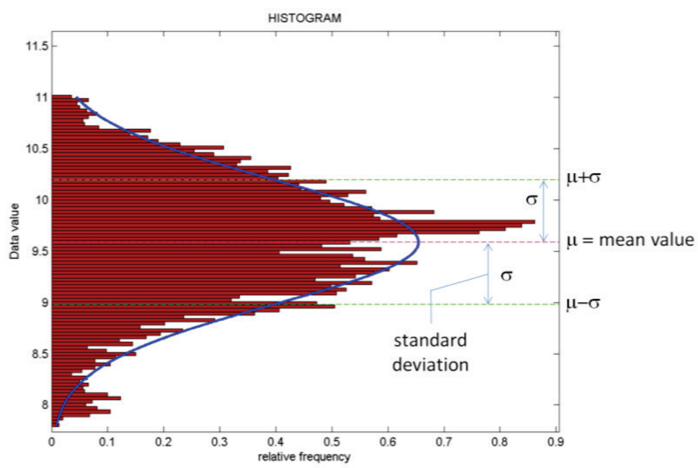
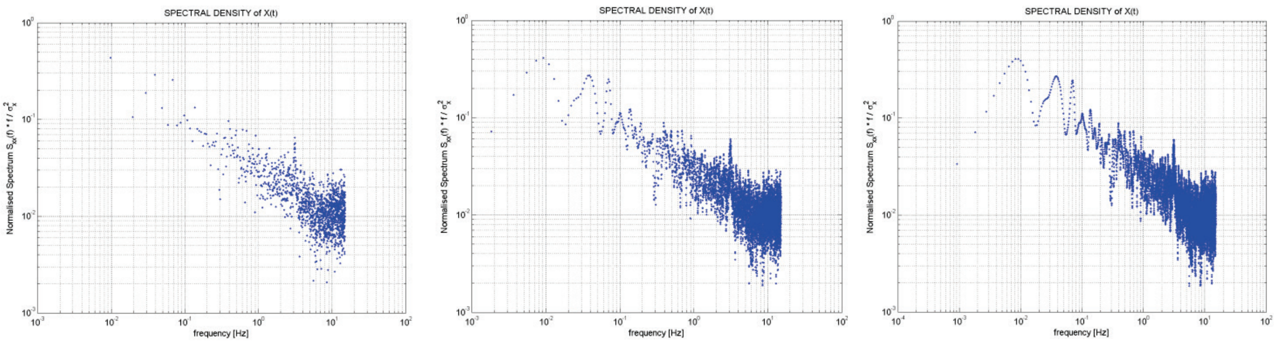


Figure 2.5 Normalised histogram compared to normal distribution density.

The histogram is calculated on the modified data (detrended / filtered) if some signal processing has been applied and converted into values of relative frequency. Hence, the histogram can directly be compared to probability density function (PDF). In the graph the curve of a normal distribution density is plotted over the histogram for better comparison.

Power Spectral Density

For the calculation of the power spectral density with `pwelch` a couple of parameters need to be defined such as the sample frequency, `fsamp`, number of overlapping windows, `Nw`, and the filter length, `nfft`. The routine `pwelch` operates with eight to 50% overlapping windows as a standard. To get a better feeling what happens you can change `Nw` and observe the effect on the resulting spectrum.



`nfft = 3072`   `nfft = 16384`   `nfft = 32768`

Figure 2.6 Effect of filter length `nfft` on resulting power spectrum.

The filter length `nfft` determines how many data points are used in the calculation of the PSD. Hence large values of `nfft` allow to "see" long waves in the signal (= low frequencies) and small values will consequently reduce of the "visible" wavelength. If `nfft` is short compared to the available number of data points in the time history the analysis will be subsequently be repeated on the remaining data points. The resulting spectra will be averaged to the final result. This principle has the effect that short values of `nfft` focus the analysis on the high frequency range (short wavelength) but reduces the scatter in the final PSD graph. The higher the value for `nfft` the more lower frequencies are visible in the spectrum but with increased scatter in the high frequency range.

The decision which value for `nfft` shall be used depends on which part of the spectrum you are more interested in. The best way to create accurate spectral densities is to analyze several time sufficiently series and calculate an average PSD from all individual spectra (ensemble averaging)

Recommendations from literature recommend that `nfft` should include all data points of the time series. The value for `nfft` should be the power of 2 (2n, where n is an integer) that is just next above the size of the data record length. For example the time history in Figure 2.6 has a record length of 18000 data points. The figure shows the difference in the spectral density when using different values of `nfft`. Here, `nfft` = 16384 = 2^{14} is just under the record length and `nfft` = 32768 = 2^{15} is the next above. It should be noted that if `nfft` is larger than the time record, the function will just append 0's to the end of the record correcting its size – called "zero padding". It has been observed that considerable zero padding adds some strange behavior to the spectral density.

The "power of 2"-rule makes the algorithm fast because of the way the FFT algorithm splits data records but is <u>not compulsory</u>! Another way for choosing a value for `nfft` is a multiple integer of the basis length 1024:

$$\texttt{nfft} = n \cdot 1024$$

Spectral Density Formats

The power spectral density (PSD) of a stochastic process X(t) is by definition the distribution of variance in the process, σ_x^2. Hence, the ordinate of the PSD has usually the same unit as the variance divided by frequency unit. This way, an integration of the PSD over frequency results again into the variance. Depending on the algorithm to calculate the PSD the area underneath the spectral ordinates might vary from the variance directly calculated from the time series. In an example of pressure fluctuation on a low-rise building (stagnation point) the two different variances are:

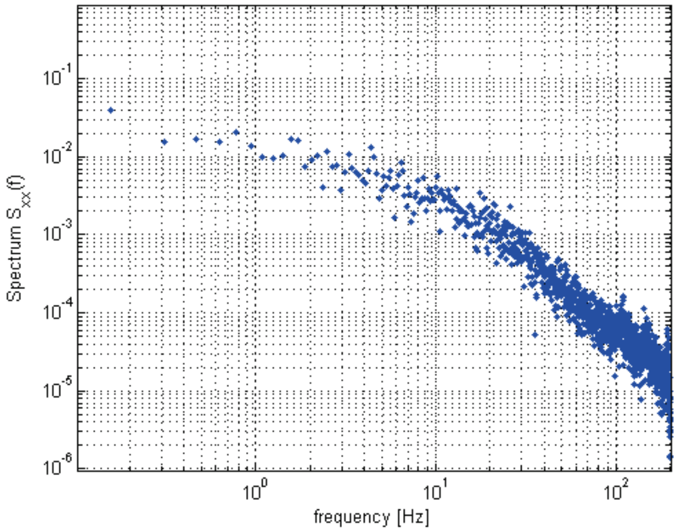


Figure 2.7 Power spectral density calculated with `pwelch`. If the time series of the wind pressure time series would be given in [Pa] the unit of the PSD is $[Pa]^2/[Hz]$

$$\sigma^2_{x,statistical} = 0.1222 \quad \text{calculated directly from time history}$$

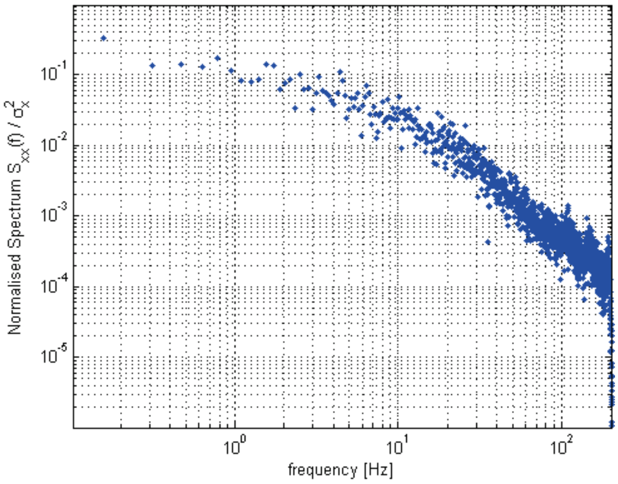
$$\sigma^2_{x,geometrical} = 0.1193 \quad \text{calculated through integration of PSD (area)}$$

The difference when using `pwelch` for calculating the spectrum in this case is about 2% and hence the spectrum is fairly accurate. Other algorithms might differ more significantly and the graph needs to be re-scaled to the actual variance:

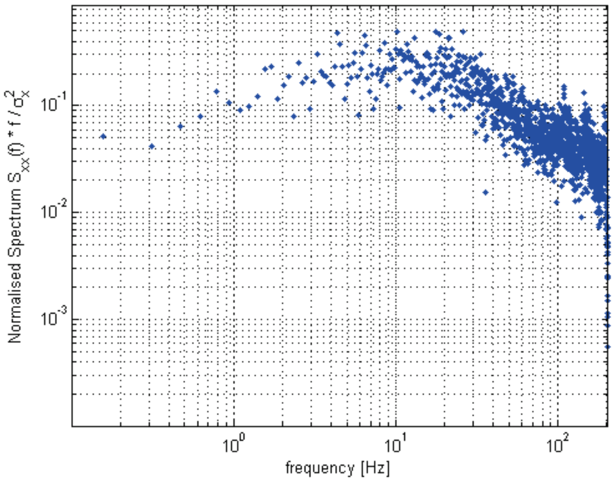
$$S_{xx}(f) = S_{xx,calc}(f) \cdot \frac{\sigma^2_{x,statistical}}{\sigma^2_{x,geometrical}}$$

Apart from the natural format of the PSD applications in other context prefer a different format. For example when comparing the characteristic of the PSD of different processes, the actual magnitude of the values, i.e. magnitude of the variance and hence the area underneath, may handicap the comparison. In this case the PSDs can be normalized to unity area dividing the ordinates with the variance (Figure 2.8, left). Integrating the spectrum would then give “1”.

Another way of presenting the PSD is to normalize the ordinate through division with variance and multiplication with frequency. This format is widely used for energy spectra of the turbulent wind (Figure 2.8, right)



Normalized PSD with unity area.



PSD with normalized ordinate.

Figure 2.8 Different normalisations of the Power Spectral Density.

The script provides all of the aforementioned formats of the PSD and plots them on top of each other at window position 3 in Figure 2.1. In the script the data of the different spectra are saved in separate fields. The spectra of the initial (unfiltered) time series are in (line 478-481):

`SNx1(:,1)` = Ordinate normalised Spectral Curves (Figure 2.8, right)
`SNx1(:,2)` = Normalizing Spectral Curves to Unit Area (Figure 2.8, left)
`SNx1(:,3)` = Spectrum with statistical variance as area (similar to Figure 2.7)
`SNx1(:,4)` = Spectrum as calculated with `pwelch` (Figure 2.7)

In case digital filtering has been applied the spectra of the modified time series are in (line 614-617):

`SNx2(:,1)` = Ordinate normalised Spectral Curves (Figure 2.8, right)
`SNx2(:,2)` = Normalizing Spectral Curves to Unit Area (Figure 2.8, left)
`SNx2(:,3)` = Spectrum with statistical variance as area (similar to Figure 2.7)
`SNx2(:,4)` = Spectrum as calculated with `pwelch` (Figure 2.7)

If no filtering has been applied `SNx1` and `SNx2` are idential.

2.2.4 Result Parameters and Vectors

At the end of the calculation the result is contained in different parameters and vectors. An overview on the available information is given below:

General Information (also displayed on screen).

`m1`	Number of time steps in time history [-]
`Tend`	Duration of parent time history [s]
`Nsub`	Number of sub-series [s]
`Tend/Nsub`	Duration of sub-series [s]
`DT`	Time step width DT [s]
`Fsamp`	Sample frequency (if [T]=s) [Hz]
`Xmean`	Mean value of X(t) [x]
`Xstd`	Standard deviation of X(t) [x]
`Xvar`	Corresponding variance [x2]
`Xmax`	Maximum peak value in X(t) [x]
`Xmin`	Minimum peak value in X(t) [x]
`Nw`	Number of overlapping sub-windows [-]
`window`	Sub-window length [-]
`nfft`	Filter depth of fft-routine [-]

Sub-series Data

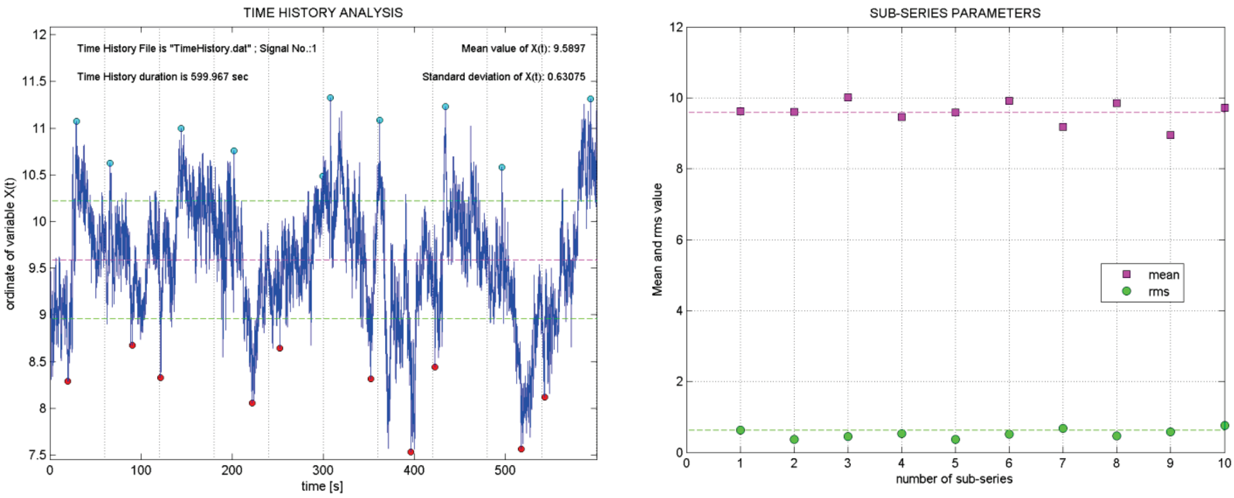


Figure 2.9 Example of Sub-series analysis applied on time history.

`Smean`	Mean value per sub-series
`Srms`	Standard deviation per sub-series
`Smin`	Minimum per sub-series
`Smax`	Maximum per sub-series

Example on displaying the data in the command window for further analysis:

```
>> Smean'

ans =

    9.6134    9.5983   10.0123    9.4570    9.5909    9.9156    9.1759    9.8526    8.9548    9.7260
```

2.3 Time Series Correlation - TSCorr

2.3.1 General Information

The script creates a correlation plot (also referred to as *scatter plot* or *scatter graph*) of two time processes of equal length. The resulting graph illustrates the relation between the two processes, $X_A(t)$ and $X_B(t)$. Running the script creates following graphical output on the screen:

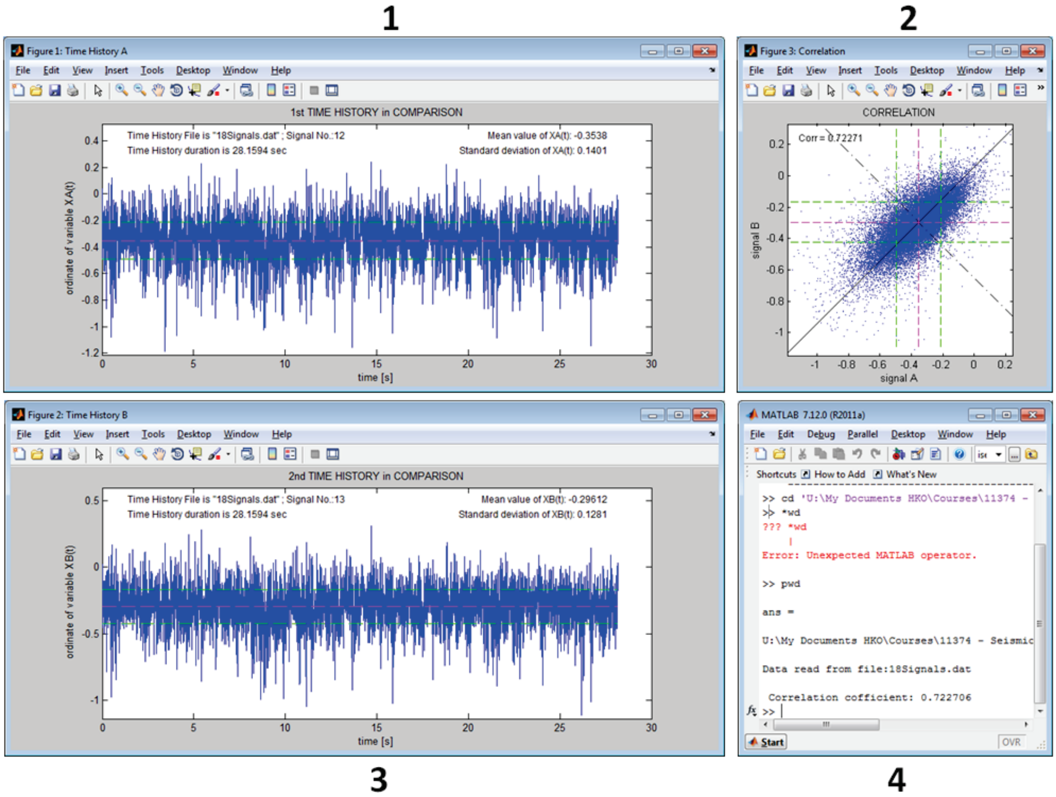


Figure 2.10 Display with different windows created when running "`TSCorr.m`".

Here, windows "Figure 1: Time History A" and "Figure 2: Time History B" display the two time histories next to each other. Any apparent similarity between the time series indicates some level of correlation. Window 2 "Figure 3: Correlation" illustrates the characteristic of the similarity in a correlation plot and window 4 is the Matlab command window. In this example the script reads (between line 58 and 76) the data file "18Signals.dat" with 18 wind load time series measured on a low-rise building (for more detailed description of data file see chapter 2.2.2). Disregarding which data set is read following information shall be provided (see script line 78 to 89):

`FileName` = File name of data file
`Fsamp` = sample frequency in [Hz] used to calculate the time axis vector is there isn't any in the data file
`DT` = time step between data points [s] = 1/Fsamp
`SignalA` = Number of first signal for correlation analysis (if applicable) – saved as `XA`
`SignalB` = Number of second signal for correlation analysis (if applicable) – saved as `XB`
`m1` = number of data points (time steps)
`TAx(i)` = Vector with values for time axis (length = `m1`), separately calculated if necessary
`XA` = Vector with data of first stochastic process (length = `m1`)
`XB` = Vector with data of second stochastic process (length = `m1`)

Figure 2.11 illustrates the construction of a correlation graph. The position of a correlation point is determined by the coordinates x_A and x_B, which are the ordinates of the respective time processes at a specific instant in time.

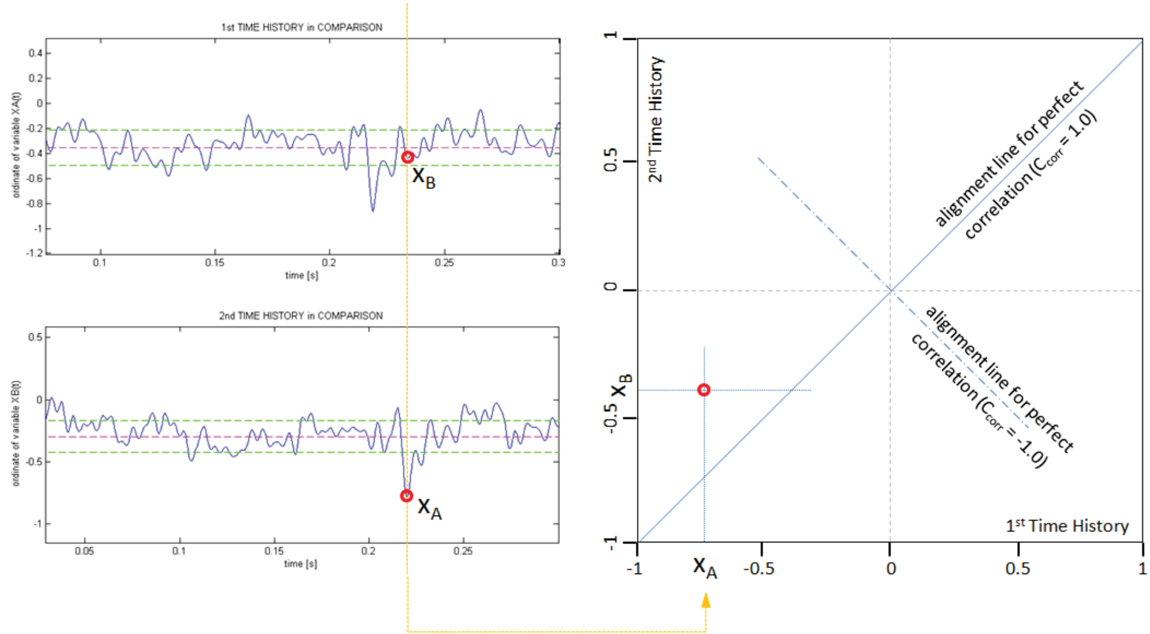


Figure 2.11 Construction of a correlation graph.

The axes of the graph correspond to the abscissa axes of the individual process time histories. If the process of the two time series vary fully synchronized and if the magnitude with which the values vary the same in both series the correlation dots align along a 45° inclined line. In this case we could say that the two series are identical to each other. Any deviation to this "perfect correlation" will appear as different inclination of the alignment line and as scatter of the dots around it. Figure 2.12 shows different examples of possible correlation plots. Alignment along a straight line indicates full correlation with the exception of vertical and horizontal orientation where then correlation will be zero.

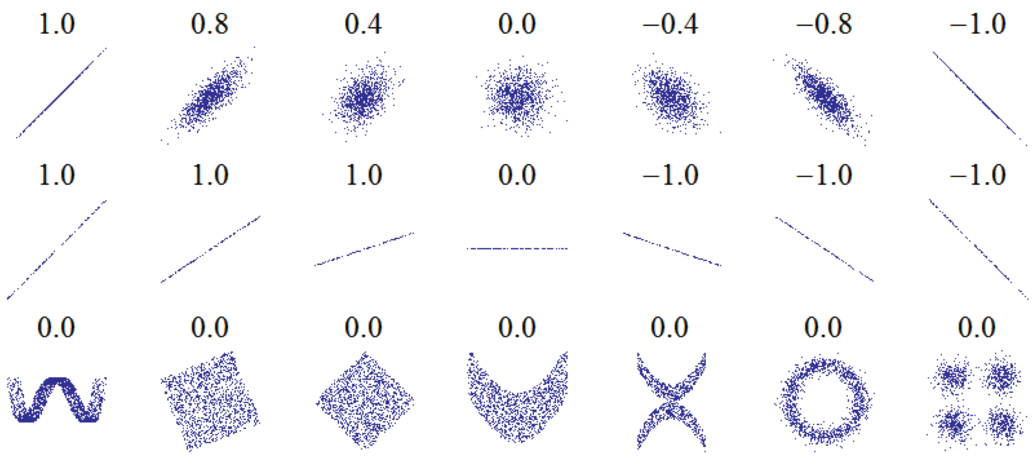


Figure 2.12 Several sets of (x, y) points, with the Pearson correlation coefficient of x and y for each set. Note that the correlation reflects the noisiness and direction of a linear relationship (top row), but not the slope of that relationship (middle), nor many aspects of nonlinear relationships (bottom). N.B.: the figure in the center has a slope of 0 but in that case the correlation coefficient is undefined because the variance of Y is zero (graph: Wikipedia, 2013).

.

2.3.2 Example

Furthermore, the density of the dots indicates similar to the histogram of a single time series the frequency of a specific pair (x_A, x_B) or in other words: it indicates the probability of a combined event where x_A and x_B occur at the same time (*joint probability*). Figure 2.13 shows the details of a correlation graph including the histograms or discrete PDFs (created with THA5) of the two compared signals.

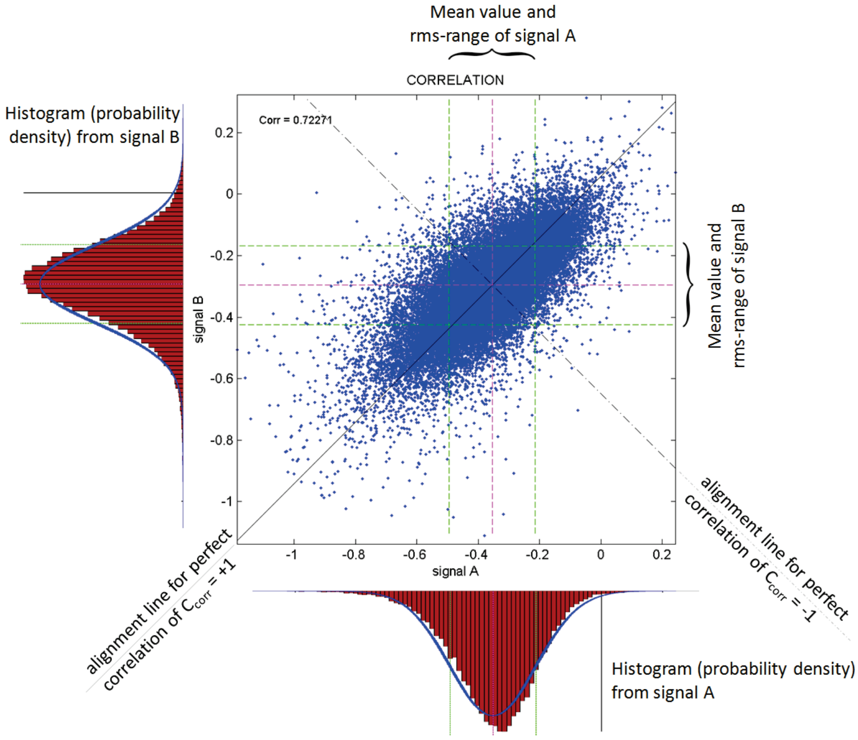


Figure 2.13 Elements of a correlation plot and PDFs of the underlying time histories.

To give a physical context on the above discussion Figure 2.14 shows relation between some time series of measured wind-induced surface pressure on a low-rise building. The characteristic of the correlation plots is now interpretable as a reflection of the load at the considered points is acting together on the building structure. This is vital information if we have to define areas with simultaneous peak loading.

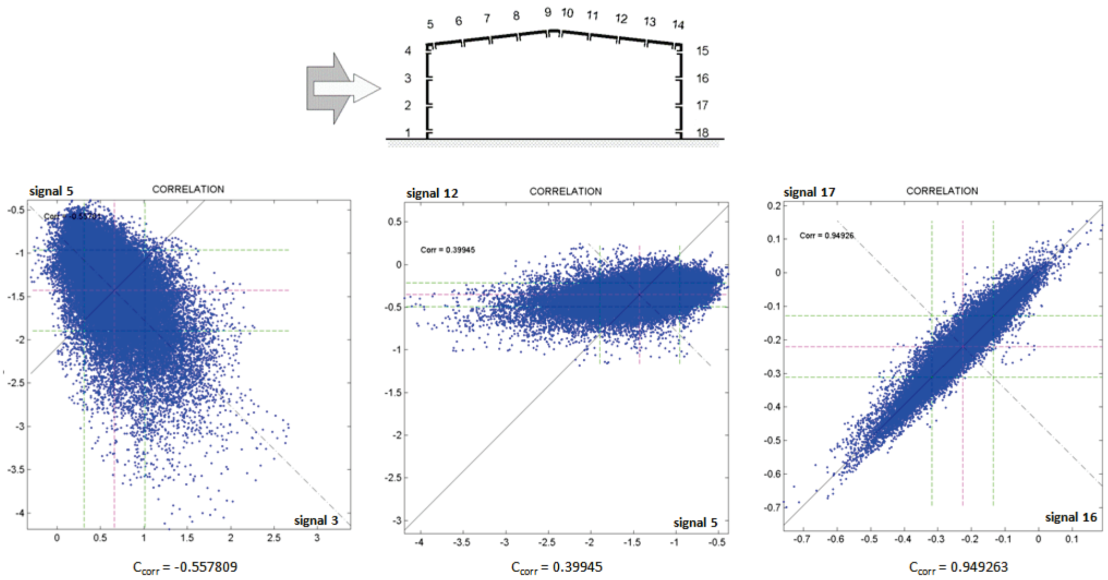


Figure 2.14 Examples for correlation plots of pressure signals measured on a low-rise building.

Table 2.1 contains the correlation coefficients between all measured signals on the low-rise building (diagonal symmetric matrix of correlation coefficients). The magnitude of the correlation coefficients is visualized in Figure 2.13. Structures of areas appear where the wind load on the surface is more or less synchronized, i.e. exhibiting higher or lower correlation to each other.

Table 2.1 Correlation coefficients between measured pressure signals on low-rise building.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	0.9673	0.917	0.7902	-0.5477	-0.6223	-0.5175	-0.3592	-0.2772	-0.2932	-0.2137	-0.1586	-0.094	-0.0337	-0.0215	-0.0131	0.0007	-0.0094
2	0.9673	1	0.9151	0.7538	-0.4969	-0.5928	-0.5101	-0.3527	-0.2563	-0.2558	-0.1838	-0.1333	-0.0784	-0.0238	-0.0106	-0.0021	0.011	0.0004
3	0.917	0.9151	1	0.8745	-0.5578	-0.6468	-0.5267	-0.3544	-0.2731	-0.303	-0.2215	-0.1675	-0.1109	-0.0575	-0.0499	-0.0463	-0.035	-0.043
4	0.7902	0.7538	0.8745	1	-0.4757	-0.5213	-0.4028	-0.269	-0.2179	-0.2628	-0.1825	-0.1281	-0.0732	-0.0255	-0.0158	-0.0121	-0.0092	-0.015
5	-0.5477	-0.4969	-0.5578	-0.4757	1	0.7154	0.365	0.2338	0.3486	0.5459	0.4549	0.3995	0.341	0.2833	0.2894	0.3037	0.292	0.2971
6	-0.6223	-0.5928	-0.6468	-0.5213	0.7154	1	0.5592	0.3034	0.3615	0.4695	0.4066	0.3616	0.3135	0.2637	0.2729	0.286	0.2749	0.2851
7	-0.5175	-0.5101	-0.5267	-0.4028	0.365	0.5592	1	0.5507	0.2688	0.3139	0.33	0.2867	0.2548	0.2259	0.2483	0.2593	0.2496	0.2601
8	-0.3592	-0.3527	-0.3544	-0.269	0.2338	0.3034	0.5507	1	0.5361	0.223	0.2374	0.2599	0.2525	0.2318	0.254	0.2656	0.2604	0.2723
9	-0.2772	-0.2563	-0.2731	-0.2179	0.3486	0.3615	0.2688	0.5361	1	0.6397	0.3222	0.2481	0.2803	0.2858	0.3125	0.3218	0.3138	0.3201
10	-0.2932	-0.2558	-0.303	-0.2628	0.5459	0.4695	0.3139	0.223	0.6397	1	0.6828	0.3683	0.3054	0.3211	0.3734	0.3908	0.385	0.3888
11	-0.2137	-0.1838	-0.2215	-0.1825	0.4549	0.4066	0.33	0.2374	0.3222	0.6828	1	0.6637	0.3719	0.3057	0.3753	0.3984	0.3981	0.4025
12	-0.1586	-0.1333	-0.1675	-0.1281	0.3995	0.3616	0.2867	0.2599	0.2481	0.3683	0.6637	1	0.7227	0.4509	0.4555	0.4676	0.466	0.4701
13	-0.094	-0.0784	-0.1109	-0.0732	0.341	0.3135	0.2548	0.2525	0.2803	0.3054	0.3719	0.7227	1	0.735	0.5926	0.5554	0.5328	0.5286
14	-0.0337	-0.0238	-0.0575	-0.0255	0.2833	0.2637	0.2259	0.2318	0.2858	0.3211	0.3057	0.4509	0.735	1	0.8436	0.7486	0.6947	0.6752
15	-0.0215	-0.0106	-0.0499	-0.0158	0.2894	0.2729	0.2483	0.254	0.3125	0.3734	0.3753	0.4555	0.5926	0.8436	1	0.9378	0.8713	0.8438
16	-0.0131	-0.0021	-0.0463	-0.0121	0.3037	0.286	0.2593	0.2656	0.3218	0.3908	0.3984	0.4676	0.5554	0.7486	0.9378	1	0.9493	0.9204
17	0.0007	0.011	-0.035	-0.0092	0.292	0.2749	0.2496	0.2604	0.3138	0.385	0.3981	0.466	0.5328	0.6947	0.8713	0.9493	1	0.9567
18	-0.0094	0.0004	-0.043	-0.015	0.2971	0.2851	0.2601	0.2723	0.3201	0.3888	0.4025	0.4701	0.5286	0.6752	0.8438	0.9204	0.9567	1

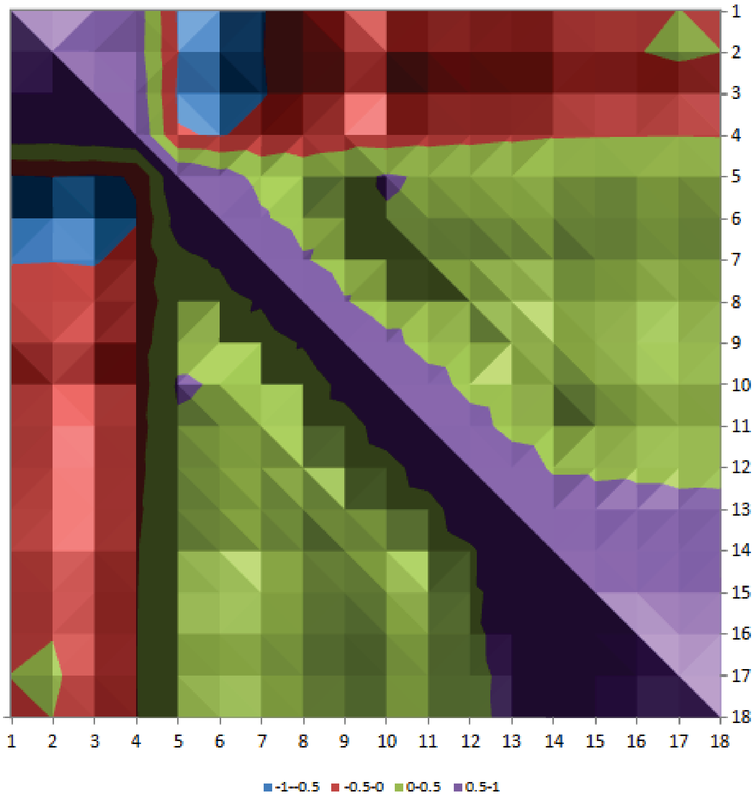


Figure 2.15 Graphical presentation of the Correlation Coefficient matrix.

2.4 Joint Probability Density Function - JPDF

2.4.1 General Information

The script calculates the joint probability of two independent variables, in this case X_A is the wind speed and X_B is the air temperature. Both variables are defined in their probability density function. Furthermore, the calculated joint probability density function (JPDF) can be evaluated with respect to some *decision criteria*. For example a decision criterion for a plume of visible water vapour can be defined by the boundary conditions of air temperature below 2°C and mean wind speed above 6m/s. The integration of the JPDF for combinations of X_A and X_B fulfilling both criteria gives the overall probability of this particular situation.

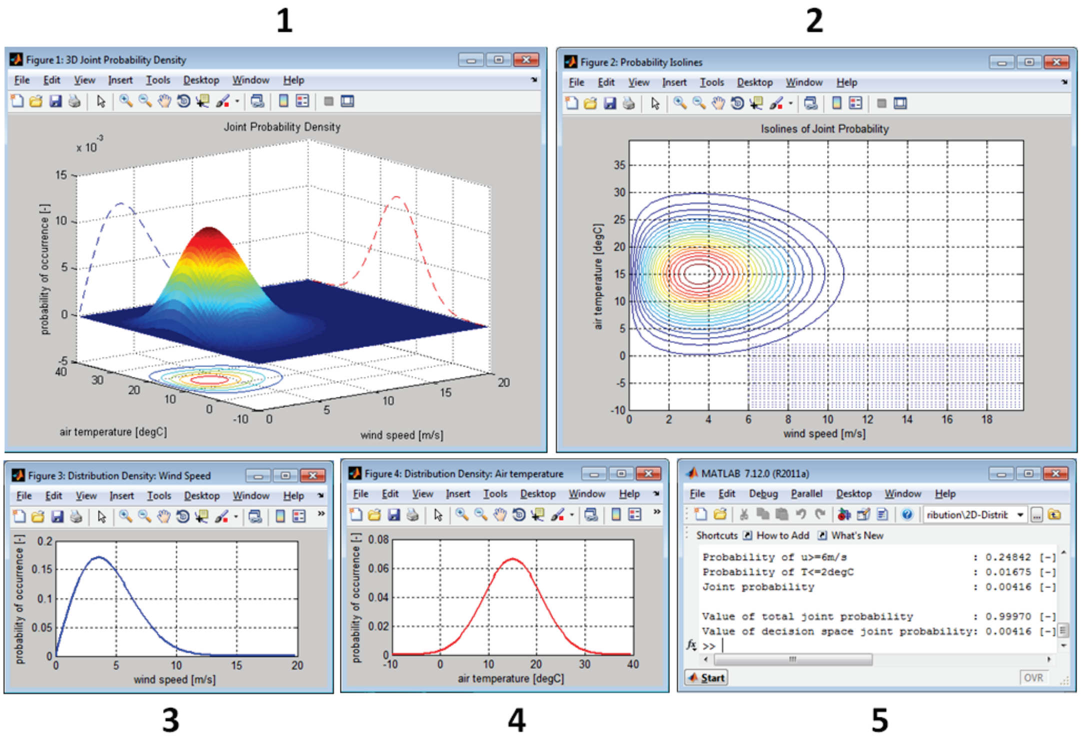


Figure 2.16 Display with different windows created when running "`JPDF.m`".

Figure 2.16 shows the different graphical outputs created by the script.

1. **Figure 1: 3D Joint Probability Density**
   Gives an overview of the resulting JPDF. Switching on the "Rotate 3D" tool allows reviewing the result. The results are calculated on a grid defined by the step width of each variable.

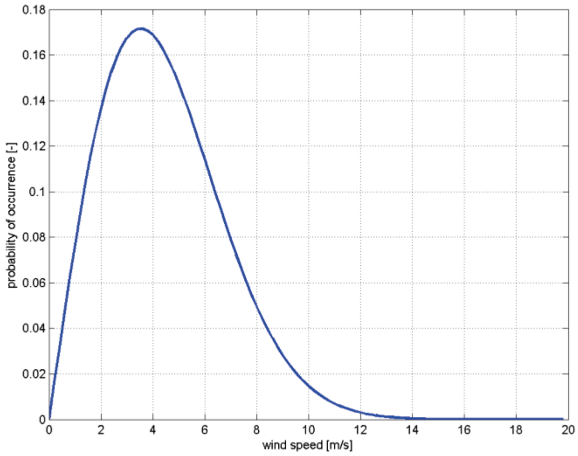
2. **Figure 2: Probability Isolines**
   Presents the JPDF as isolines of joint probability on a 2D plane. Additionally, those points fulfilling pre-defined decision criteria are indicated to indicate the area underneath the JPDF that gets integrated to determine the joint probability of the particular case.

3. **Figure 3: Distribution Density Wind Speed**
   PDF of first variable X_A (here: wind speed)

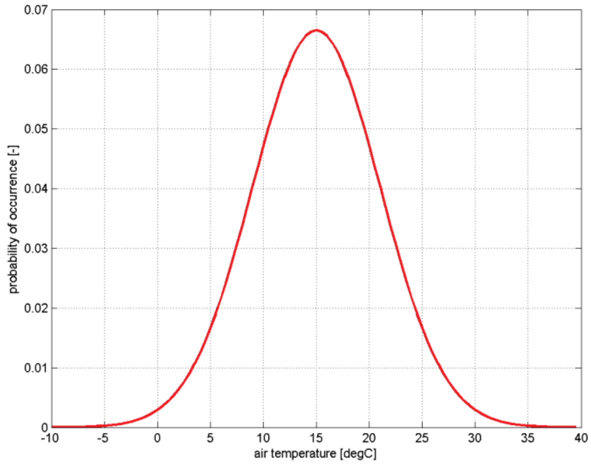
4. **Figure 4: Distribution Density Air Temperature**
   PDF of second variable X_B (here: air temperature)

5. **Matlab Command Window**

2.4.2 Parameter Settings



Probability of 10-minutes mean wind speed.



Probability of air temperature.

Figure 2.17 Main input information is the definition of the two independent variables in their PDF.

The PDFs for each variable are in this case defined by functions. In case of special PDFs the curves can also be defined directly at the discrete steps of the corresponding variable. In our example the ordinates of the PDFs are calculated over a certain range with a certain step width. Below, the syntax to generate the PDF vectors for our example is given:

```
% 2) CONTRUCTION OF DISTRIBUTION DENSITIES:
%    ------------------------------------------
% 2.1 Definition of calculation settings

NU = 100;    % discretisation of the velocity axis 0-30m/s in 0.3m/s steps
NT = 100;    % discretisation of temperature axis -10 to 40degC in 0.5degC steps
dU = 0.2;    % wind velocity step width [m/s]
dT = 0.5;    % air temperature step width [degC]

U0 = 0;      % lowest wind speed [m/s]
T0 = -10;    % lowest air temperature [degC]

U    = zeros(1,NU);  % vector for wind speed range
T    = zeros(1,NT);  % vector for airtemperature range
R    = zeros(NT,NU); % Result matrix
D    = zeros(NT,NU); % Decision matrix
pdfU = zeros(1,NU);  % PDF vector for wind velocity
pdfT = zeros(1,NT);  % PDF vector for air temperature
cdfU = zeros(1,NU);  % CDF vector for wind velocity
cdfT = zeros(1,NT);  % CDF vector for air temperature
```

Most important is that the vectors `pdfU` and `pdfT` are defined for all values in the range of the two variables (here: `U` and `T`), either by calculation or point-by-point. The area underneath the PDFs is usually unity but can also be scaled in case of dependent events. For better orientation of the JPDF the PDFs of the individual variables are projected on the side walls of the graph (Figure 2.18).

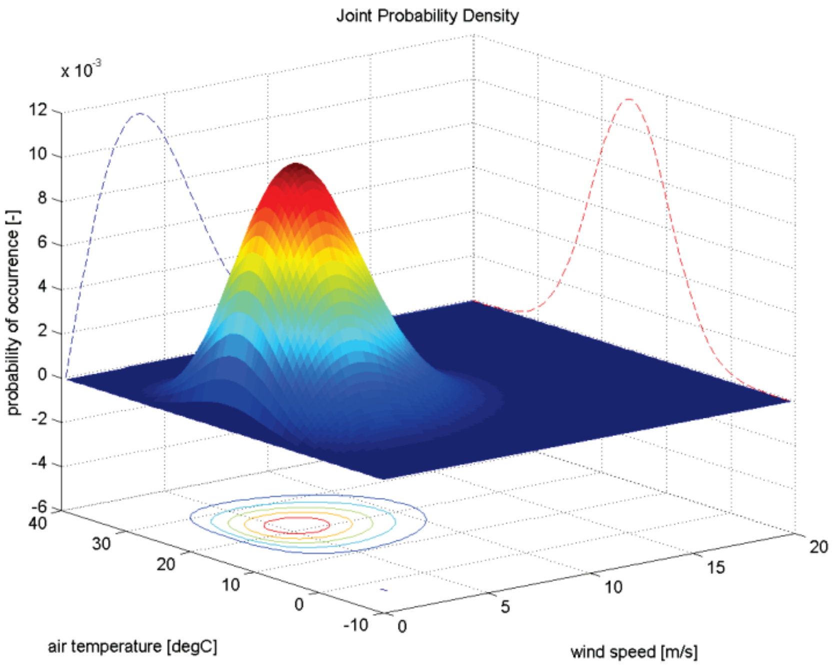


Figure 2.18 3D presentation of Joint Probability Density.

The JPDF can be used to estimate the probability of situations consisting of the simultaneous occurrence of X_A and X_B within certain boundary conditions. In Figure 2.19 a situation or case is defined by wind speeds larger than 6m/s and air temperatures below 2°C:

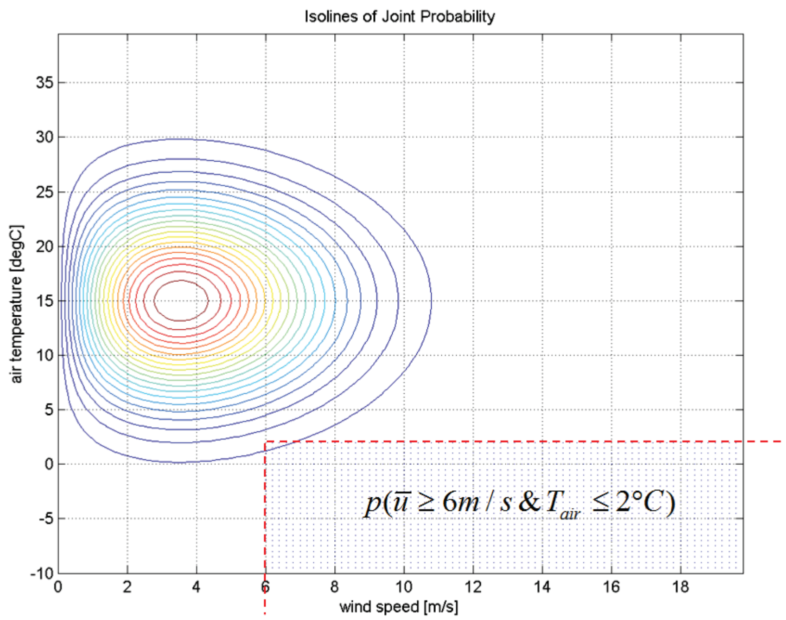


Figure 2.19 Isolines of JPD and integration points fulfilling decision criterion.

The calculated probabilities are printed on the screen (needs to be adjusted for other cases):

```
Probability of u>=6m/s                : 0.24842 [-]
Probability of T<=2degC               : 0.01675 [-]
Joint probability                     : 0.00416 [-]

Value of total joint probability      : 0.99970 [-]
Value of decision space joint probability: 0.00416 [-]
```

3. Matlab Function Descriptions

3.1 "detrend"

detrend

Remove linear trends

Syntax

```
y = detrend(x)
y = detrend(x,'constant')
y = detrend(x,'linear',bp)
```

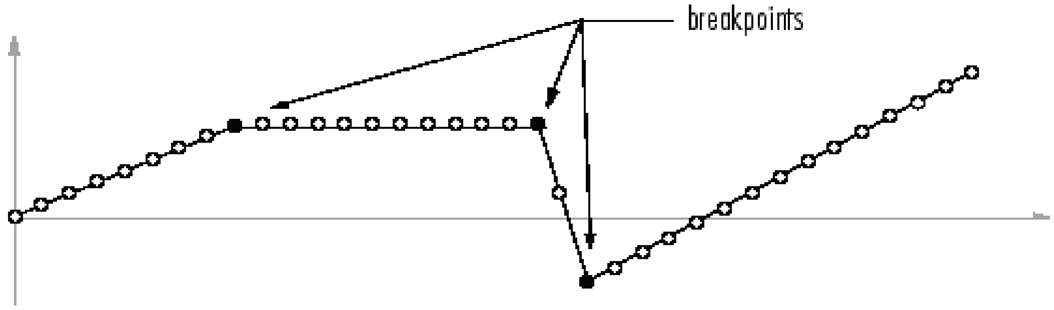
Description

`detrend` removes the mean value or linear trend from a vector or matrix, usually for FFT processing.

`y = detrend(x)` removes the best straight-line fit from vector `x` and returns it in `y`. If `x` is a matrix, `detrend` removes the trend from each column.

`y = detrend(x,'constant')` removes the mean value from vector `x` or, if `x` is a matrix, from each column of the matrix.

`y = detrend(x,'linear',bp)` removes a continuous, piecewise linear trend from vector `x` or, if `x` is a matrix, from each column of the matrix. Vector `bp` contains the indices of the breakpoints between adjacent linear segments. The breakpoint between two segments is defined as the data point that the two segments share.



`detrend(x,'linear')`, with no breakpoint vector specified, is the same as `detrend(x)`.

Examples

```
sig = [0 1 -2 1 0 1 -2 1 0];        % signal with no linear trend
trend = [0 1 2 3 4 3 2 1 0];        % two-segment linear trend
x = sig+trend;                      % signal with added trend
y = detrend(x,'linear',5)           % breakpoint at 5th element

y =

   -0.0000
    1.0000
   -2.0000
    1.0000
    0.0000
    1.0000
   -2.0000
    1.0000
   -0.0000
```

Note that the breakpoint is specified to be the fifth element, which is the data point shared by the two segments.

Algorithms

`detrend` computes the least-squares fit of a straight line (or composite line for piecewise linear trends) to the data and subtracts the resulting function from the data. To obtain the equation of the straight-line fit, use `polyfit`.

3.2 "butter"

butter

Butterworth filter design

Syntax

```
[z,p,k]=butter(n,Wn)
[z,p,k] = butter(n,Wn, 'ftype')
[b,a]=butter(n,Wn)
[b,a]=butter(n,Wn, 'ftype')
[A,B,C,D]=butter(n,Wn)
[A,B,C,D] = butter(n,Wn,' ftype')
[z,p,k]=butter(n,Wn,' s')
[z,p,k] = butter(n,Wn,' ftype','s')
[b,a]=butter(n,Wn,' s')
[b,a]=butter(n,Wn, 'ftype','s')
[A,B,C,D]=butter(n,Wn,' s')
[A,B,C,D] = butter(n,Wn,' ftype','s')
```

Description

`butter` designs lowpass, bandpass, highpass, and bandstop digital and analog Butterworth filters. Butterworth filters are characterized by a magnitude response that is maximally flat in the passband and monotonic overall.

Butterworth filters sacrifice rolloff steepness for monotonicity in the pass- and stopbands. Unless the smoothness of the Butterworth filter is needed, an elliptic or Chebyshev filter can generally provide steeper rolloff characteristics with a lower filter order.

Digital Domain

`[z,p,k] = butter(n,Wn)` designs an order `n` lowpass digital Butterworth filter with normalized cutoff frequency `Wn`. It returns the zeros and poles in length `n` column vectors `z` and `p`, and the gain in the scalar `k`.

`[z,p,k] = butter(n,Wn, 'ftype')` designs a highpass, lowpass, or bandstop filter, where the string `'ftype'` is one of the following:

- `'high'` for a highpass digital filter with normalized cutoff frequency `Wn`
- `'low'` for a lowpass digital filter with normalized cutoff frequency `Wn`
- `'stop'` for an order `2*n` bandstop digital filter if `Wn` is a two-element vector, `Wn = [w1 w2]`. The stopband is $w1 < \omega < w2$.

Cutoff frequency is that frequency where the magnitude response of the filter is $\sqrt{1/2}$. For `butter`, the normalized cutoff frequency `Wn` must be a number between 0 and 1, where 1 corresponds to the Nyquist frequency, π radians per sample.

If `Wn` is a two-element vector, `Wn = [w1 w2]`, `butter` returns an order `2*n` digital bandpass filter with passband $w1 < \omega < w2$.

With different numbers of output arguments, `butter` directly obtains other realizations of the filter. To obtain the transfer function form, use two output arguments as shown below.

> **Note** See Limitations below for information about numerical issues that affect forming the transfer function.

`[b,a] = butter(n,Wn)` designs an order `n` lowpass digital Butterworth filter with normalized cutoff frequency `Wn`. It returns the filter coefficients in length `n+1` row vectors `b` and `a`, with coefficients in descending powers of *z*.

$$H(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(n+1)z^{-n}}{1 + a(2)z^{-1} + \ldots + a(n+1)z^{-n}}$$

`[b,a] = butter(n,Wn, 'ftype')` designs a highpass, lowpass, or bandstop filter, where the string `'ftype'` is `'high'`, `'low'`, or `'stop'`, as described above.

To obtain state-space form, use four output arguments as shown below:

`[A,B,C,D] = butter(n,Wn)` or

`[A,B,C,D] = butter(n,Wn,' ftype')` where `A`, `B`, `C`, and `D` are

$$x[n+1] = Ax[n] + Bu[n]$$
$$y[n] = Cx[n] + Du[n]$$

and *u* is the input, *x* is the state vector, and *y* is the output.

Analog Domain

`[z,p,k] = butter(n,Wn,' s')` designs an order `n` lowpass analog Butterworth filter with angular cutoff frequency `Wn` rad/s. It returns the zeros and poles in length `n` or `2*n` column vectors `z` and `p` and the gain in the scalar `k`. `butter`'s angular cutoff frequency `Wn` must be greater than 0 rad/s.

If `Wn` is a two-element vector with `w1 < w2`, `butter(n,Wn,'s')` returns an order `2*n` bandpass analog filter with passband `w1` < ω < `w2`.

`[z,p,k] = butter(n,Wn,' ftype','s')` designs a highpass, lowpass, or bandstop filter using the `ftype` values described above.

With different numbers of output arguments, `butter` directly obtains other realizations of the analog filter. To obtain the transfer function form, use two output arguments as shown below:

`[b,a] = butter(n,Wn,' s')` designs an order `n` lowpass analog Butterworth filter with angular cutoff frequency `Wn` rad/s. It returns the filter coefficients in the length `n+1` row vectors `b` and `a`, in descending powers of *s*, derived from this transfer function:

$$H(s) = \frac{B(s)}{A(s)} = \frac{b(1)s^n + b(2)s^{n-1} + \ldots + b(n+1)}{s^n + a(2)s^{n-1} + \ldots + a(n+1)}$$

`[b,a] = butter(n,Wn, 'ftype','s')` designs a highpass, lowpass, or bandstop filter using the `ftype` values described above.

To obtain state-space form, use four output arguments as shown below:

`[A,B,C,D] = butter(n,Wn,' s')` or

`[A,B,C,D] = butter(n,Wn,' ftype','s')` where `A`, `B`, `C`, and `D` are

$$x = Ax + Bu$$
$$y = Cx + Du$$

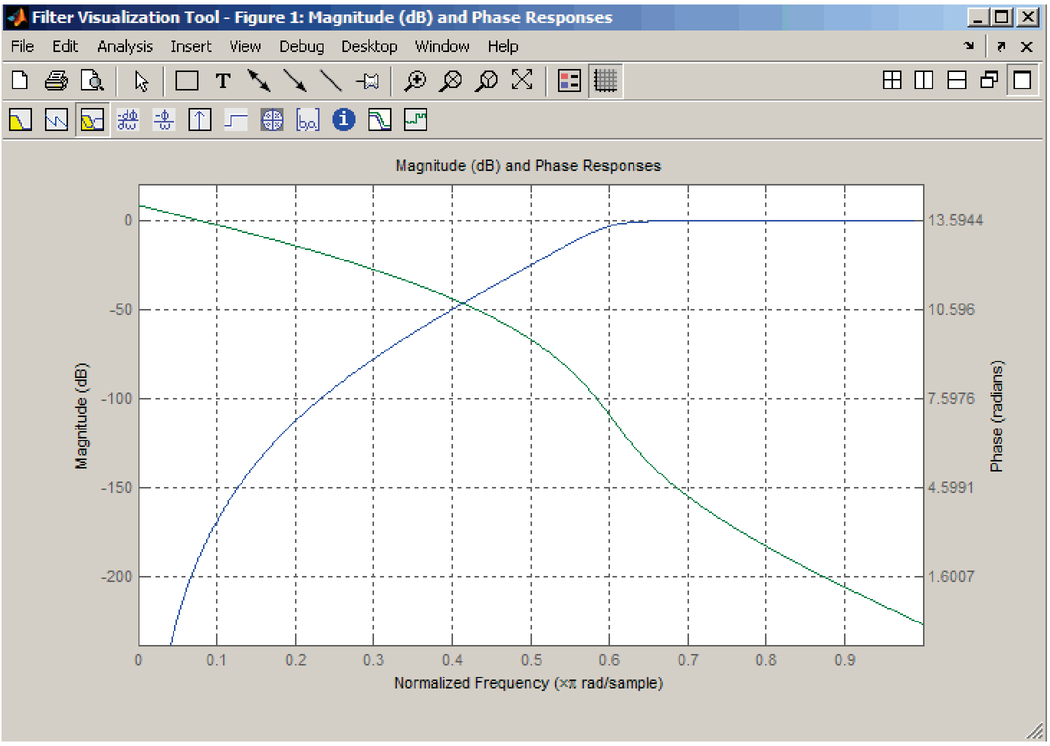
and *u* is the input, *x* is the state vector, and *y* is the output.

Examples

Highpass Filter

For data sampled at 1000 Hz, design a 9th-order highpass Butterworth filter with cutoff frequency of 300 Hz, which corresponds to a normalized value of 0.6:

```
[z,p,k] = butter(9,300/500,'high');
[sos,g] = zp2sos(z,p,k);        % Convert to SOS form
Hd = dfilt.df2tsos(sos,g);     % Create a dfilt object
h = fvtool(Hd);                % Plot magnitude response
set(h,'Analysis','freq')        % Display frequency response
```



Limitations

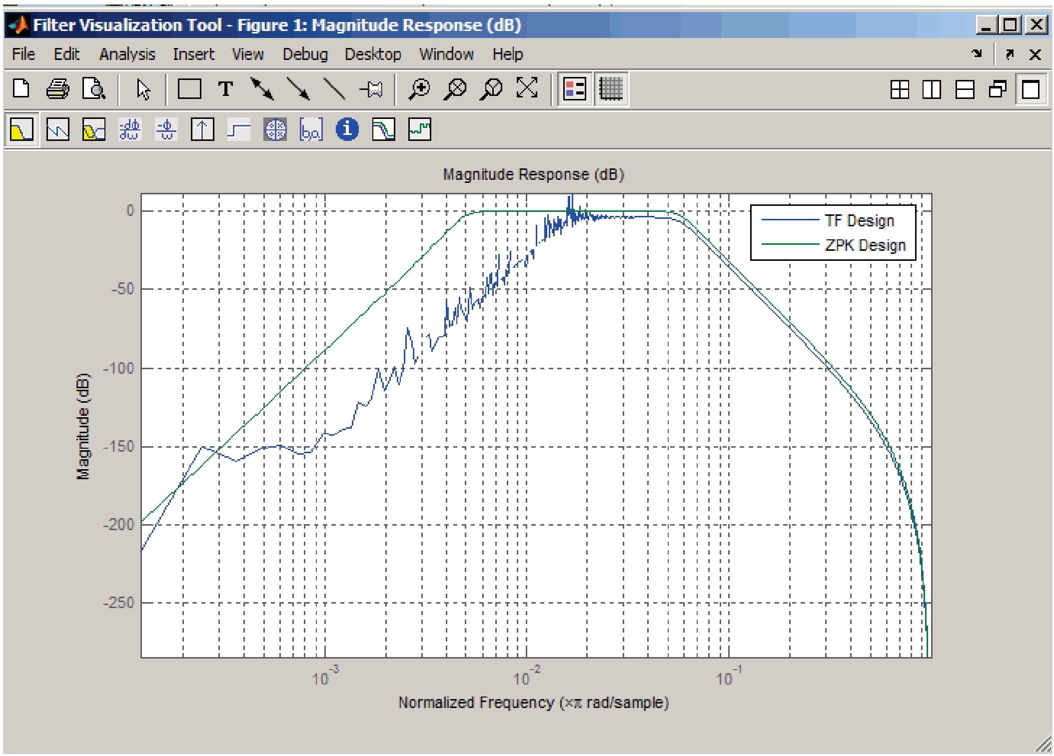
In general, you should use the `[z,p,k]` syntax to design IIR filters. To analyze or implement your filter, you can then use the `[z,p,k]` output with [zp2sos]() and an sos [dfilt]() structure. For higher order filters (possibly starting as low as order 8), numerical problems due to roundoff errors may occur when forming the transfer function using the `[b,a]` syntax. The following example illustrates this limitation:

```
n = 6; Wn = [2.5e6 29e6]/500e6;
ftype = 'bandpass';

% Transfer Function design
[b,a] = butter(n,Wn,ftype);
h1=dfilt.df2(b,a);       % This is an unstable filter.

% Zero-Pole-Gain design
[z, p, k] = butter(n,Wn,ftype);
[sos,g]=zp2sos(z,p,k);
h2=dfilt.df2sos(sos,g);

% Plot and compare the results
hfvt=fvtool(h1,h2,'FrequencyScale','log');
legend(hfvt,'TF Design','ZPK Design')
```



Algorithms

`butter` uses a five-step algorithm:

1. It finds the lowpass analog prototype poles, zeros, and gain using the `buttap` function.
2. It converts the poles, zeros, and gain into state-space form.
3. It transforms the lowpass filter into a bandpass, highpass, or bandstop filter with desired cutoff frequencies, using a state-space transformation.
4. For digital filter design, `butter` uses `bilinear` to convert the analog filter into a digital filter through a bilinear transformation with frequency prewarping. Careful frequency adjustment guarantees that the analog filters and the digital filters will have the same frequency response magnitude at `Wn` or `w1` and `w2`.
5. It converts the state-space filter back to transfer function or zero-pole-gain form, as required.

3.3 "filtfilt"

filtfilt

Zero-phase digital filtering

Syntax

```
y = filtfilt(b,a,x)
y = filtfilt(SOS,G,x)
```

Description

`y = filtfilt(b,a,x)` performs zero-phase digital filtering by processing the input data, `x`, in both the forward and reverse directions[1]. The vector `b` provides the numerator coefficients of the filter and the vector `a` provides the denominator coefficients. If you use an all-pole filter, enter `1` for `b`. If you use an all-zero filter (FIR), enter `1` for `a`. After filtering the data in the forward direction, `filtfilt` reverses the filtered sequence and runs it back through the filter. The result has the following characteristics:

- Zero-phase distortion
- A filter transfer function, which equals the squared magnitude of the original filter transfer function
- A filter order that is double the order of the filter specified by `b` and `a`

`filtfilt` minimizes start-up and ending transients by matching initial conditions, and you can use it for both real and complex inputs. Do not use `filtfilt` with differentiator and Hilbert FIR filters, because the operation of these filters depends heavily on their phase response.

> **Note** The length of the input `x` must be more than three times the filter order defined as `max(length(b)-1,length(a)-1)` .

`y = filtfilt(SOS,G,x)` zero-phase filters the data `x` using the second-order section (biquad) filter represented by the matrix `SOS` and scale values `G`. The matrix `SOS` is an L-by-6 matrix containing the L second-order sections. The matrix `SOS` must be of the form:

$$\begin{pmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{pmatrix}$$

where each row are the coefficients of a biquad filter. The vector of filter scale values,`G`, must have a length between 1 and L+1.

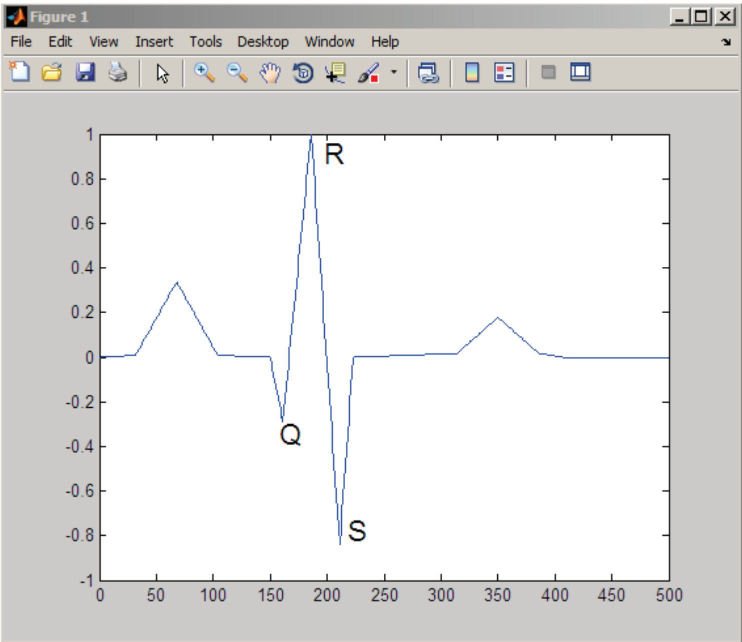
> **Note** When implementing zero-phase filtering using a second-order section filter, the length of the input `x` must be more than 6 samples.

Examples

Zero-phase filtering helps preserve features in the filtered time waveform exactly where those features occur in the unfiltered waveform. To illustrate the use of `filtfilt` for zero-phase filtering, consider an electrocardiogram waveform as an example.

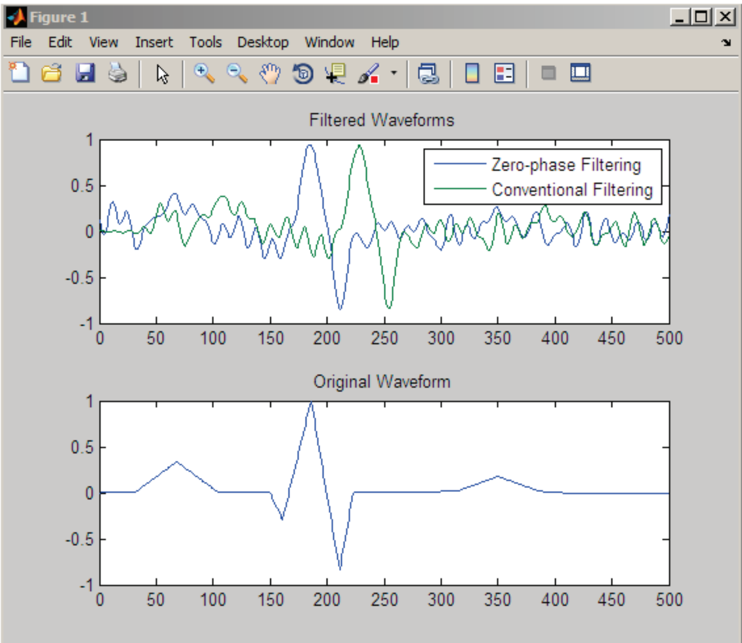
```
plot(ecg(500)); %plot ECG signal
```

The QRS complex is an important feature in the ECG waveform beginning around time point 160 in this example.



The following sample corrupts the ECG waveform with additive noise, constructs a lowpass FIR equiripple filter, and filters the noisy waveform using both zero-phase and conventional filtering. Because the filter is an all-zero (FIR) filter, the input `a` equals 1.

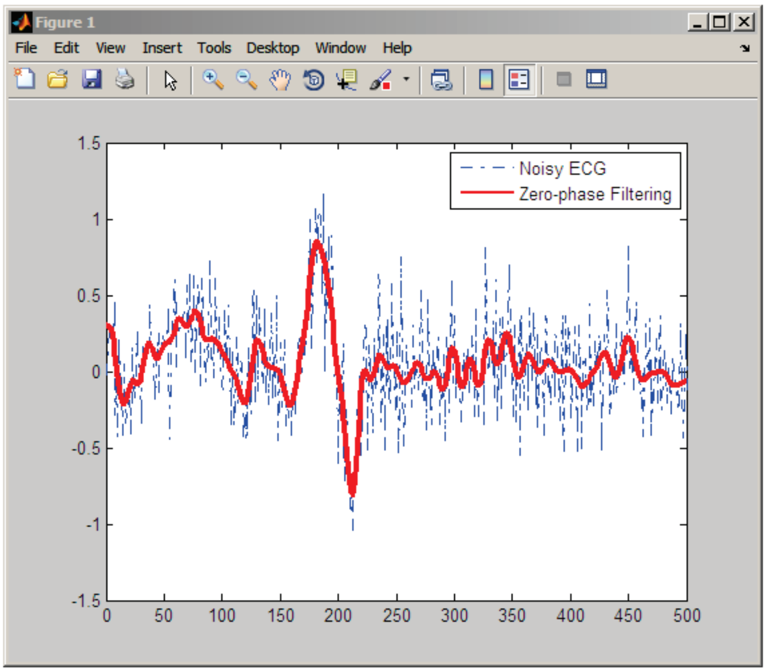
```
x=ecg(500)'+0.25*randn(500,1); %noisy waveform
h=fdesign.lowpass('Fp,Fst,Ap,Ast',0.15,0.2,1,60);
d=design(h,'equiripple'); %Lowpass FIR filter
y=filtfilt(d.Numerator,1,x); %zero-phase filtering
y1=filter(d.Numerator,1,x); %conventional filtering
subplot(211);
plot([y y1]);
title('Filtered Waveforms');
legend('Zero-phase Filtering','Conventional Filtering');
subplot(212);
plot(ecg(500));
title('Original Waveform');
```



Zero-phase filtering reduces noise in the signal and preserves the QRS complex at the same time it occurs in the original signal. Conventional filtering reduces noise in the signal, but delays the QRS complex.

Repeat the above using a Butterworth second-order section filter:

```
h=fdesign.lowpass('N,F3dB',12,0.15);
d1 = design(h,'butter');
y = filtfilt(d1.sosMatrix,d1.ScaleValues,x);
plot(x,'b-.'); hold on;
plot(y,'r','linewidth',3);
legend('Noisy ECG','Zero-phase Filtering','location','NorthEast
```



References

[1] Oppenheim, A.V., and R.W. Schafer, *Discrete-Time Signal Processing,* Prentice-Hall, 1989, pp.284–285.

[2] Mitra, S.K., *Digital Signal Processing, 2nd ed.,* McGraw-Hill, 2001, Sections 4.4.2 and 8.2.5.

[3] Gustafsson, F., Determining the initial states in forward-backward filtering, *IEEE Transactions on Signal Processing*, April 1996, Volume 44, Issue 4, pp.988–992.

3.4 "pwelch"

pwelch

PSD using Welch's method

Syntax

```
[Pxx,w] = pwelch(x)
[Pxx,w] = pwelch(x,window)
[Pxx,w] = pwelch(x,window,noverlap)
[Pxx,w] = pwelch(x,window,noverlap,nfft)
[Pxx,w] = pwelch(x,window,noverlap,w)
[Pxx,f] = pwelch(x,window,noverlap,nfft,fs)
[Pxx,f] = pwelch(x,window,noverlap,f,fs)
[...] = pwelch(x,window,noverlap,..., 'range')
pwelch(x,...)
```

Description

`[Pxx,w] = pwelch(x)` estimates the power spectral density `Pxx` of the input signal vector `x` using Welch's method. Welch's method splits the data into overlapping segments, computes modified periodograms of the overlapping segments, and averages the resulting periodograms to produce the power spectral density estimate.

- The vector `x` is segmented into eight sections of equal length, each with 50% overlap.
- Any remaining (trailing) entries in `x` that cannot be included in the eight segments of equal length are discarded.
- Each segment is windowed with a Hamming window (see `hamming`) that is the same length as the segment.

The power spectral density is calculated in units of power per radians per sample. The corresponding vector of frequencies `w` is computed in radians per sample, and has the same length as `Pxx`.

A real-valued input vector `x` produces a full power one-sided (in frequency) PSD (by default), while a complex-valued `x` produces a two-sided PSD.

In general, the length *N* of the FFT and the values of the input `x` determine the length of `Pxx` and the range of the corresponding normalized frequencies. For this syntax, the (default) length *N* of the FFT is the larger of 256 and the next power of 2 greater than the length of the segment. The following table indicates the length of `Pxx` and the range of the corresponding normalized frequencies for this syntax.

PSD Vector Characteristics for an FFT Length of N (Default)

Real/Complex Input Data	Length of Pxx	Range of the Corresponding Normalized Frequencies
Real-valued	$(N/2)+1$	$[0, \pi]$
Complex-valued	N	$[0, 2\pi)$

`[Pxx,w] = pwelch(x,window)` calculates the modified periodogram using either:

- The window length `window` for the Hamming window when `window` is a positive integer
- The window weights specified in `window` when `window` is a vector

With this syntax, the input vector `x` is divided into an integer number of segments with 50% overlap, and each segment is the same length as the window. Entries in `x` that are left over after it is divided into segments are discarded. If you specify `window` as the empty vector `[]`, then the signal data is divided into eight segments, and a Hamming window is used on each one.

`[Pxx,w] = pwelch(x,window,noverlap)` divides `x` into segments according to `window`, and uses the integer `noverlap` to specify the number of signal samples (elements of `x`) that are common to two adjacent segments. `noverlap` must be less than the length of the window you specify. If you specify `noverlap` as the empty vector `[]`, then `pwelch` determines the segments of `x` so that there is 50% overlap (default).

`[Pxx,w] = pwelch(x,window,noverlap,nfft)` uses Welch's method to estimate the PSD while specifying the length of the FFT with the integer `nfft`. If you specify `nfft` as the empty vector `[]`, the number of points used in the PSD estimate defaults to a maximum of 256 or the next power of two greater than the length of `window`. For a window length less than or equal to 256, `nfft` defaults to 256. For a window length greater than 256, `nfft` defaults to the next power of two.

The length of `Pxx` and the frequency range for `w` depend on `nfft` and the values of the input `x`. The following table indicates the length of `Pxx` and the frequency range for `w` for this syntax.

PSD and Frequency Vector Characteristics

Real/Complex Input Data	nfft Even/Odd	Length of Pxx	Range of w
Real-valued	Even	`(nfft/2 + 1)`	$[0, \pi]$
Real-valued	Odd	`(nfft + 1)/2`	$[0, \pi)$
Complex-valued	Even or odd	`nfft`	$[0, 2\pi)$

`[Pxx,w] = pwelch(x,window,noverlap,w)` estimates the two-sided PSD at the normalized frequencies specified in the vector `w` using the Goertzel algorithm. The frequencies of `w` are rounded to the nearest DFT bin commensurate with the resolution of the signal. The units of `w` are rad/sample.

`[Pxx,f] = pwelch(x,window,noverlap,nfft,fs)` uses the sampling frequency `fs` specified in hertz (Hz) to compute the PSD vector (`Pxx`) and the corresponding vector of frequencies (f). In this case, the units for the frequency vector are in Hz. The spectral density produced is calculated in units of power per Hz. If you specify `fs` as the empty vector `[]`, the sampling frequency defaults to 1 Hz.

The frequency range for `f` depends on `nfft`, `fs`, and the values of the input `x`. The length of `Pxx` is the same as in the PSD and Frequency Vector Characteristics above. The following table indicates the frequency range for `f` for this syntax.

PSD and Frequency Vector Characteristics with fs Specified

Real/Complex Input Data	nfft Even/Odd	Range of f
Real-valued	Even	[0,fs/2]
Real-valued	Odd	[0,fs/2)
Complex-valued	Even or odd	[0,fs)

[Pxx,f] = pwelch(x,window,noverlap,f,fs) estimates the two-sided PSD at the normalized frequencies specified in the vector f using the Goertzel algorithm. The f vector returned is the same vector as the input f vector. The frequencies of f are rounded to the nearest DFT bin commensurate with the resolution of the signal.

[...] = pwelch(x,window,noverlap,..., *'range'*) specifies the range of frequency values. This syntax is useful when x is real. The string *'range'* can be either:

- 'twosided': Compute the two-sided PSD over the frequency range [0,fs). This is the default for determining the frequency range for complex-valued x.
  - If you specify fs as the empty vector, [], the frequency range is [0,1).
  - If you don't specify fs, the frequency range is [0, 2π).
- 'onesided': Compute the one-sided PSD over the frequency ranges specified for real x. This is the default for determining the frequency range for real-valued x.

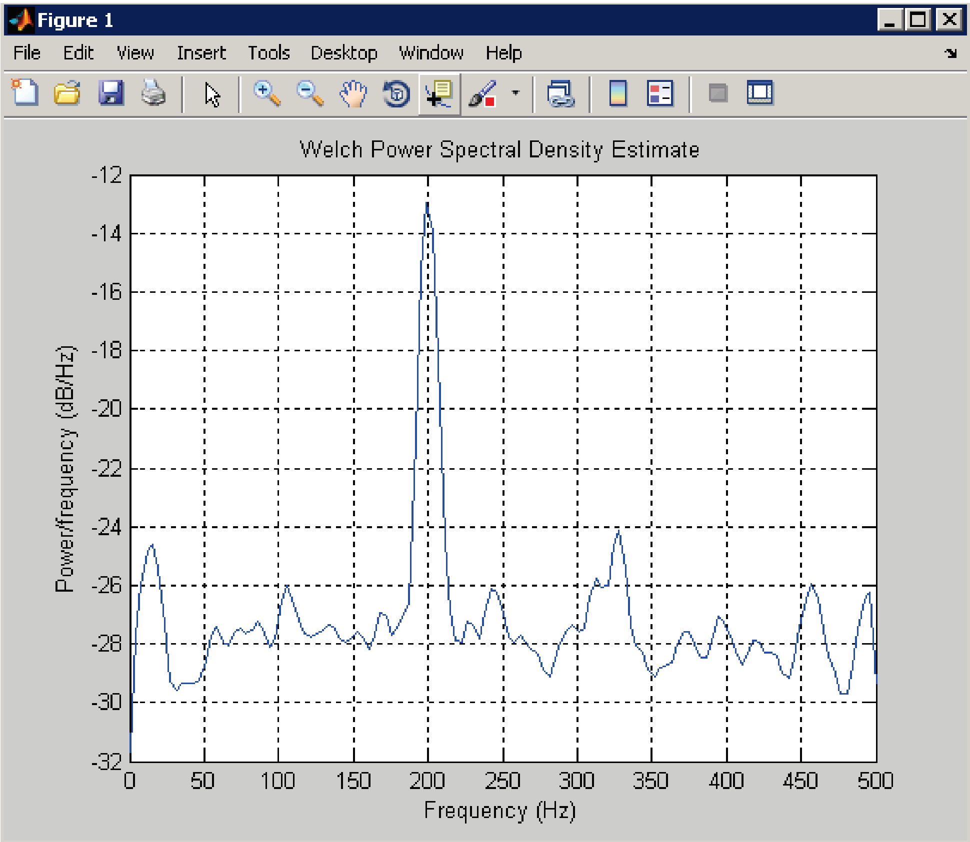
The string *'range'* can appear anywhere in the syntax after noverlap.

pwelch(x,...) with no output arguments plots the PSD estimate in dB per unit frequency in the current figure window.

Examples

Estimate the PSD of a signal composed of a sinusoid plus noise, sampled at 1000 Hz. Use 33-sample windows with 32-sample overlap, and the default FFT length, and display the two-sided PSD estimate:

```
Fs = 1000;
t = 0:1/Fs:1;
% 200Hz cosine + noise
randn('state',0);
x = cos(2*pi*t*200) + randn(size(t));
pwelch(x,128,120,[],Fs,'onesided')
```



Algorithms

`pwelch` calculates the power spectral density using Welch's method (see References below):

1. The input signal vector `x` is divided into k overlapping segments according to `window` and `noverlap` (or their default values). If the window size is larger than the number of FFT points (`NFFT`) , the signal is divided into `NFFT`-length segments and then, the last segment is padded with zeros.
2. The specified (or default) window is applied to each segment of `x`. (No preprocessing is done before applying the window to each segment.)
3. An `nfft`-point FFT is applied to the windowed data.
4. The (modified) periodogram of each windowed segment is computed.
5. The set of modified periodograms is averaged to form the spectrum estimate $S(e^{j\omega})$.
6. The resulting spectrum estimate is scaled to compute the power spectral density as $S(e^{j\omega})/F$, where F is
   - 2π when you do not supply the sampling frequency
   - `fs` when you supply the sampling frequency

The number of segments k that `x` is divided into is calculated as:

- Eight if you don't specify `window`, or if you specify it as the empty vector `[]`
- $k = \frac{m-o}{l-o}$ if you specify window as a nonempty vector or a scalar

  In this equation, m is the length of the signal vector `x`, o is the number of overlapping samples (`noverlap`), and l is the length of each segment (the window length).

References

[1] Hayes, M., *Statistical Digital Signal Processing and Modeling*, John Wiley & Sons, 1996.

[2] Stoica, P., and R.L. Moses, *Introduction to Spectral Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1997, pp. 52-54.

[3] Welch, P.D, "The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms,"*IEEE Trans. Audio Electroacoustics, Vol. AU-15 (June 1967)*, pp.70-73.

4. Matlab Codes

4.1 "THA5.m"


```
prog='THA5';
%---------------------------------------------------------------------------------------
%                TIME HISTORY ANALYSIS - Detrend, Filter and Display
%=======================================================================================
%
% DESCRIPTION:      This program has been developed to view and analyse a time series of
% ------------      a measured signal. The main features are:
%
%                   - viewing the time steries as a graph.
%                   - presentation of data points as histogram (discrete probability density).
%                   - calculation of power spectral density.
%                   - calculation and displaying the mean values and standard deviations of sub-series.
%                   - Identification of maximum and minimum in sub-series.
%
%                   Furthermore, some signal processing can be performed on the original data,
%                   namely detrending and high or low-pass filtering. The difference between the
%                   initial and the altered signal can be visualised in the graph of the time
%                   series and the spectral density the spectral density.
%
% Program ID:
% -----------
% File name       : THA5.m
% Author          : Holger Koss (hko)
% Development Log : 2009-04-28  hko   Basic structure of the program
%                   2012-05-04  hko   Adoption to a general tool to first analysis and
%                                     treatment of a time history
%                   2012-10-29  hko   Adoption for course 11374
%                   2012-11-07  hko   Reading additional types of files
%                   2013-11-11  hko   Different formats of PSD
%
% Necessary files  : "TimeHistory"  - Ascii file containing in the first column the
%                                     time axis and in the second column the time history
%                                     of the investigated quantity. Both columns are in
%                                     model scale.
%
%                   "18Signals.dat" - File with 18 time series of pressure coefficients
%                                     measured in a wind tunnel test on a model low-rise
%                                     building (based on cpcent.00).
%
%                   "cpcent.00"     - Fragmented file with cp time series from 12 storm
%                                     events, each lead by the velocity pressure [kPa]
%                                     in the wind tunnel test at eaves height.
%                                     (Exercise: "Characteristic Wind Load Statistic on Low-rise
%                                     Buildings")
%
%                   "BendTS.txt"    - Time series of the base bending moment [N] of a
%                                     high-rise building.
%                                     (Exercise: "Dynamic Response of a High-rise Building
%                                     to Wind Loading")
%=======================================================================================
 close all
 clear all
%=======================================================================================
% 1) READING OF INPUT DATA:
%    ----------------------
%    This version of the program is prepared to read two different input files.
%    A simple file with two columns, time axis (t) and variable X(t), and a
%    file with a matrix of 18 time series of measured pressure coefficients.
%    In this case the time series needs to be generated to plot the series.
%
%    1 = Read simple input file
%    2 = Read matrix of cp-series
%    3 = Read observation 10min-mean data Reykjavik harbour 2002 (hole year)
%    4 = Reading on hour of anemometer data from Oeresund Bridge (30Hz)
%    5 = Reading fragmented CpCent data files with wind tunnel speeds inbetween blocks

Iread = 5;

% 1.1) Reading simple data set:
%      ----------------------------------------------------------
%      File with two columns: time axis and data of variable X(t). The
%      values in this file have been recorded at top of the pylon of the
%      Great Belt Bridge (TimeHistory).

if Iread == 1

    FileName = 'TimeHistory.dat';          % Name of input file (10min wind Great Belt Bridge)
%   FileName = 'BendTS.txt';               % Name of input file (Base bending moment CDJ)
    fid      = fopen(FileName,'r');
    Series   = fscanf(fid,'%e',[2 inf]);   % Matrix with time series
    Series   = Series';                    % Transposed matrix
    [m1 n1]  = size(Series);               % Number of rows (m1) and columns (n1)
    status   = fclose(fid);

    disp(['Data read from file:',FileName])

    TAx      = Series(:,1);                % Time series
    X0       = Series(:,2);                % Saving selected data to variable vector
    SignalNo = 1;                          % Default value since there is only one signal to analyse

    NFFTcase = 3*1024;                     % Filter depth of the fft-routine N*1024
    Nwindow  = 8;                          % Number of windows (default = 8) for pwelch SFD calculation
    DT       = (TAx(10)-TAx(1))/9;         % Calculation of time step
    Fsamp    = 1/DT;                       % Calculation of sample frequency
    Nbin     = 30;                         % Number of bins to generate histogram
```

THA5.m

```
    Nsub     = 30;                          % Number of sub-series

end

% 1.2) Reading Data from Pressure Measurements on Low-rise Building:
%      -----------------------------------------------------------
%      Here, we read a file with 18 time series of pressure coefficients
%      measured on a wind tunnel model of a low-rise building. Each of these
%      time series is a signal of a stochastic process. The input file does
%      not contain the time axis. To plot the process the time axis needs
%      to be generated separately.
%

if Iread == 2

    FileName = '18Signals.dat';              % Name of input file
    fid      = fopen(FileName,'r');         % Echo print on screen
    qhmwk    = fscanf(fid,'%e',[1 1]);      % velocity pressure [kN/m^2]
    cp       = fscanf(fid,'%e',[18 inf]);   % Matrix with pressure coefficient time series
    Series   = cp';                         % Transposed matrix where each column is a signal
    [m1 n1]  = size(Series);                % Number of rows (m1) and columns (n1)
    status   = fclose(fid);

    disp(['Data read from file:',FileName])

    SignalNo = 12;                          % Number of signal to be analysed (1-18)
    X0       = Series(:,SignalNo);          % Saving selected data to input vector

    NFFTcase = 2*1024;                      % Filter depth of the fft-routine N*1024
    Nwindow  = 8;                           % Number of windows (default = 8) for pwelch SFD calculation
    Fsamp    = 1600;                        % Sample frequency in
    DT       = 1/Fsamp;                     % Calculation of time step
    Nbin     = 100;                         % Number of bins to generate histogram
    Nsub     = 11;                           % Number of sub-series

    for i=1:m1                              % Generation of time axis with m1 steps
        TAx(i) = (i-1)*DT;
    end


end

% 1.3) Reading Data from Wind Records at Reykjavik Harbour, entire year 2002:
%      ----------------------------------------------------------------------
%      The data are provided by a private weather station located in the harbour of Reykjavik.
%      The file contains amogst other 10 minutes mean and gust wind speeds continuously
%      recorded throughout the year 2002. The file format is given in the list below.
%
%      Column  Content
%      -----------------------------------------------------
%         1    Hours
%         2    Minutes
%         3    Day
%         4    Month
%         5    Year
%         6    10 Minutes mean wind speed [m/s]
%         7    Gust wind speed [m/s]
%         8    Wind Direction [deg]
%         9    Standard deviation of wind direction [deg]
%        10    Atmospheric pressure [mbar]
%        11    Air temperature [degC]
%

if Iread == 3

    FileName = 'Reykjavik2002.txt';          % Name of the 1st time history file
    fid1     = fopen(FileName,'r');
    Series   = fscanf(fid1,'%g',[11 inf]);
    Series   = Series';
    [m1 n1]  = size(Series);
    status   = fclose(fid1);

    SignalNo = 6;                            % Number of signal to be analysed (1-18)
    X0       = Series(:,SignalNo);           % Saving selected data to input vector

    NFFTcase = 10*1024;                      % Filter depth of the fft-routine N*1024
    Nwindow  = 8;                            % Number of windows (default = 8) for pwelch SFD calculation
    Fsamp    = 0.001667;                     % Sample frequency in [Hz]
    DT       = 600;                          % 10min mean data
    Nbin     = 50;                           % Number of bins to generate histogram
    Nsub     = 12;                           % Number of sub-series

    % Calculating a continous time index:
    % ----------------------------------
    % (time index is generated in 10-minutes steps as the smallest time unit available)

    DayMon   = [31 28 31 30 31 30 31 31 30 31 30 31]; % days per month for time axis
    for i=1:m1
        if Series(i,4)==1;  Month=0      ; end
        if Series(i,4)==2;  Month=44640  ; end
        if Series(i,4)==3;  Month=84960  ; end
        if Series(i,4)==4;  Month=129600; end
        if Series(i,4)==5;  Month=172800; end
        if Series(i,4)==6;  Month=217440; end
        if Series(i,4)==7;  Month=260640; end
        if Series(i,4)==8;  Month=305280; end
        if Series(i,4)==9;  Month=349920; end
        if Series(i,4)==10; Month=393120; end
        if Series(i,4)==11; Month=437760; end
        if Series(i,4)==12; Month=480960; end
        TAx(i) = Series(i,1)*60+Series(i,2)+(Series(i,3)-1)*24*60+Month;  % time axis in minutes
    end

end
```

THA5.m

```
% 1.4) Reading record file from Oeresund Bridge Monitoring:
%      ------------------------------------------------------------------
%      The data are recorded on the Oresund Bridge and cover a period of one hour.
%      The input file has one sub array "wind" with 8 columns of following structure:
%
%      Anemometer A: located at second shortest cable
%      1st   U component
%      2nd   V component
%      3rd   W component
%      4th   Azimuth
%
%      Anemometer B: located at midspan
%      5st   U component
%      6nd   V component
%      7rd   W component
%      8th   Azimuth
%
%      With V direction coinciding with geographical north, U coincide with Eastand W
%      is vertical direction Azimuth is the angle that define the position of the wind
%      flow along the hoorizontal plane. It is calculated from 0 to 360 clockwise along
%      the horizontal plane. 0 is the Geographical north.
%      Each columns is 1 hour long (108000) sampled at 30Hz.


if Iread == 4

%   FileName = 'wind_RawData_20120320_162217.mat'; % Starting 4.22pm
%   FileName = 'wind_RawData_20120320_172250.mat'; % Starting 5.22pm
%   FileName = 'wind_RawData_20120320_182323.mat'; % Starting 6.22pm
    FileName = '3 records of consecutive hours'; % Starting 6.22pm
    load 'wind_RawData_20120320_162217.mat';
    Series1 = wind;
    [m1 n] = size(Series1);
    load 'wind_RawData_20120320_172250.mat';
    Series2 = wind;
    [m1 n] = size(Series2);
    load 'wind_RawData_20120320_182323.mat';
    Series3 = wind;
    [m1 n] = size(Series3);

    m3  = 3*m1; % All three files are of the same length!

    X0a1 = zeros(m1,1);
    X0a2 = zeros(m1,1);
    X0a3 = zeros(m1,1);
    X0b1 = zeros(m1,1);
    X0b2 = zeros(m1,1);
    X0b3 = zeros(m1,1);
    X0a  = zeros(m3,1); % vector for combined series
    X0b  = zeros(m3,1); % vector for combined series
    % Horizontal resulting conponent for anemometer A:
    for i=1:m1
        X0a1(i)      = sqrt(Series1(i,1)^2+Series1(i,2)^2);
        X0a2(i)      = sqrt(Series2(i,1)^2+Series2(i,2)^2);
        X0a3(i)      = sqrt(Series3(i,1)^2+Series3(i,2)^2);
        X0a(i)       = X0a1(i);
        X0a(i+m1)    = X0a2(i);
        X0a(i+2*m1)  = X0a3(i);
    end

    % Horizontal resulting conponent for anemometer B:
    for i=1:m1
        X0b1(i)= sqrt(Series1(i,5)^2+Series1(i,6)^2);
        X0b2(i)= sqrt(Series2(i,5)^2+Series2(i,6)^2);
        X0b3(i)= sqrt(Series3(i,5)^2+Series3(i,6)^2);
        X0b(i)       = X0b1(i);
        X0b(i+m1)    = X0b2(i);
        X0b(i+2*m1)  = X0b3(i);
    end

%   X0        = X0a;                          % Three hours wind speeds at second shortest cable
%   X0        = X0b;                          % Three hours wind speed at midspan
%   X0        = X0a1;                         % First hour wind speed at short cable
%   X0        = X0a2;                         % Second hour wind speed at short cable
%   X0        = X0a3;                         % Third hour wind speed at short cable
    X0        = X0b1;                         % First hour wind speed at midspan
%   X0        = X0b2;                         % Second hour wind speed at midspan
%   X0        = X0b3;                         % Third hour wind speed at midspan
%   X0        = Series1(:,4);                 % One hour wind direction
    m1        = m1;                           % Record length: m1 for one hour
                                              %                m3 for three hours

    SignalNo = 9999;                          % Combination of different signals

    NFFTcase = 20*1024;                       % Filter depth of the fft-routine N*1024
    Nwindow  = 8;                             % Number of windows (default = 8) for pwelch SFD calculation
    Fsamp    = 30;                            % Sample frequency in [Hz]
    DT       = 1/Fsamp;                       % Time step [s]
    Nbin     = 100;                           % Number of bins to generate histogram
    Nsub     = 1;                            % Number of sub-series

    for i=1:m1                                % Generation of time axis with m1 steps
        TAx(i) = (i-1)*DT;
    end

end

% 1.5) Reading orginal formated data files for CpCent time series:
%      ----------------------------------------------------------------------
%      The input file consists of time series from 18 signals. The time
%      series have a record length 4096 steps sampled at 1600Hz in a wind
%      tunnel test. The mean wind speed at which the test has been
%      performed is given as mean velocity pressure [kPa] at the start of the
```

THA5.m

```
%      block with 4096x18 data entries. All-in-all 12 of the above
%      described data sets, i.e. mean velocity pressure and signal time
%      series, are contained in the input file. The data reading accounts
%      for the special structure of the input file.
%

if Iread == 5

    FileName = 'cpcent.00';                 % Name of input file
    %FileName = 'cpcent.01';                % Name of input file

    Nstorm   = 12;                          % number of sub-series
    Ntap     = 18;                          % No of taps
    length   = 4096;                        % number of time steps per storm
    SignalNo = 3;                           % Number of signal to be analysed (1-18)

    disp(['Data read from file:',FileName])
    Series   = zeros(Nstorm*length,Ntap);
    fid      = fopen(FileName,'r');         % Echo print on screen
    index=0;
    for istorm = 1:12
        f1               = (istorm-1)*length+1;
        f2               = istorm*length;
        qhmwk(istorm)    = fscanf(fid,'%e',[1 1]);    % velocity pressure [kN/m^2]
        cp               = fscanf(fid,'%e',[18 length]);
        Series(f1:f2,:) = cp'; % saving the 12 data sets as continuous times series
        fprintf(1,'Storm Number considered: %g %g\n',istorm,qhmwk(istorm))
    end
    status   = fclose(fid);

    [m1 n1]  = size(Series);                % Number of rows (m1) and columns (n1)
    X0       = Series(:,SignalNo);          % Saving selected data to input vector

    NFFTcase = 10*1024;                      % Filter depth of the fft-routine N*1024
    Nwindow  = 8;                           % Number of windows (default = 8) for pwelch SFD calculation
    Fsamp    = 1600;                        % Sample frequency in
    DT       = 1/Fsamp;                     % Calculation of time step
    Nbin     = 100;                         % Number of bins to generate histogram
    Nsub     = 12;                           % Number of sub-series

    for i=1:m1                              % Generation of time axis with m1 steps
        TAx(i) = (i-1)*DT;
    end

end

%=========================================================================================
% NOTE: At this point in the program you should have following information available:
%
%       SignalNo = Number of signal that has been chosen to be analysed - saved as X0(i)
%       X0(i)    = Vector with data of stochastic process (length = m1)
%       TAx(i)   = Vector with values for time axis (length = m1)

%       m1       = number of data points (time steps)
%       NFFTcase = FFT filter length, determine by trial, shall not exceed m1
%       Nwindow  = Number of windows (default = 8) for pwelch SFD calculation
%       Fsamp    = sample frequency in [Hz]
%       DT       = time step between data points [s] = 1/Fsamp
%       Nbin     = Number of bins to generate histogram
%       Nsub     = Number of sub-series in which the signal can be divided
%                  to calculate sub-mean and rms-values
%
%       Disregarding what data you want to analyse, just make sure that after reading the
%       input file you define the parameters and vectors listed above.
%
%=========================================================================================
% 2) ANALYSIS SETTING:
%    -----------------
% 2.2) Parameter Definitions:
%      The setting of the parameter switches different options for the analysis on
%      and off. The detailed setting for the different actions are defined in
%      section 1.4.                      <0> = no action
%                                        <1> = action activated
%
%      The program allows for following data modification and analysis:
%
%      1. DETREND
%      2. STATSITSICS part 1
%      3. SPECTRAL DENSITY part 1
%      4. DIGITAL FILTERING
%      5. STATSITSICS part 2
%      6. SPECTRAL DENSITY part 2
%

DoAct1  = 0;   % Linear detrending the time series (no break points)
DoAct2  = 0;   % Identify, display and safe sub-series maxima and minima in vector
DoAct3  = 0;   % Digital filtering
DoAct4  = 0;   % Saving modified data in external file "Data2.dat"

%     Adjustments for display
%     Displ = 1 displaying modified data, if applied, and related results (set as default)
%     Displ = 2 displaying both initial and modified data and results for comparison

Displ   = 2;   % Display parameter


% 2.2) Parameter Definitions:
%      ---------------------
%      General:
%      --------

pi     = 4*atan(1.);   % Circular constant

%      Spectral Density:
```

```
%      -----------------
%      The spectral density is calculated using Welch's method in following format:
%
%      [Pxx,f] = pwelch(x,window,noverlap,nfft,fs)
%
%      Description (from the Matlab Desktop Help):
%      ------------------------------------------
%      [Pxx,w] = pwelch(x) estimates the power spectral density Pxx of the input signal
%      vector x using Welch's method. Welch's method splits the data into overlapping segments,
%      computes modified periodograms of the overlapping segments, and averages the resulting
%      periodograms to produce the power spectral density estimate.
%
%     - The vector x is segmented into eight sections of equal length, each with 50% overlap.
%     - Any remaining (trailing) entries in x that cannot be included in the eight segments of
%       equal length are discarded.
%     - Each segment is windowed with a Hamming window (see hamming) that is the same length
%     - as the segment.
%
%     [Pxx,f] = pwelch(x,window,noverlap,nfft,fs) uses the sampling frequency fs specified in
%     hertz (Hz) to compute the PSD vector (Pxx) and the corresponding vector of frequencies (f).
%     In this case, the units for the frequency vector are in Hz. The spectral density produced
%     is calculated in units of power per Hz. If you specify fs as the empty vector [], the
%     sampling frequency defaults to 1 Hz.

Nw       = Nwindow;              % Number of windows (default = 8)
window   = floor(m1*2/(1+Nw)); % length of windows assumed 50% overlap (still automatic default)
nfft     = NFFTcase;             % Filter depth (should not exceed numer of time steps!)
fs       = Fsamp;                % Sample frequency [Hz]

%      Digital filtering of the signal:
%      -------------------------------
%      Fn      = order of the filter (using standard 6th-order Butterworth filter)
%      Ftype  = 'high' for a highpass digital filter with cutoff frequency CutOff
%               (high-frequency signals pass)
%      Ftype  = 'low' for a lowpass digital filter with cutoff frequency CutOff
%               (low-frequency signals pass)
%      CutOff = Frequency [Hz] below/above which the frequency content will be filtered out

Fn     = 6;
Ftype  ='low';
CutOff = 0.5;

%==========================================================================================
% 3) DATA PROCESSING:
%    ---------------
%    3.1 Detrending unmodified X0(t):
%    -------------------------------
%    The function "detrend" removes the mean value or linear trend from a vector or matrix.
%
%    detrend(x,'linear')     - removing of a linear trend
%    detrend(x,'linear',bp) - removing of linear trend between break points
%
%    A breakpoint between two segments is defined as the data point that the two segments share.
%    The break points are given in vector "bp".

if DoAct1 == 1
    X1 = detrend(X0,'linear'); % no breakpoints defined
else
    X1 = X0;
end

%    3.2 Digital Filtering of X1(t):
%    ------------------------------
% Calculation of unfiltered spectrum for later comparison

X1d       = detrend(X1,'constant'); % Removing mean value before performing FFT
[Sxx,f]   = pwelch(X1d,window,[],nfft,fs);
Df        = (f(10)-f(1))/9;          % Frequency resolution of calculated spectrum
XvarG     = trapz(f(:),Sxx(:,1));    % Area underneath calcultaed SDF-curve (geometrical variance)
Xvar      = (std(X1d))^2;            % statistical variance
SNx1(:,1) = Sxx(:,1).*f./XvarG;      % Ordinate normalised Spectral Curves
SNx1(:,2) = Sxx(:,1)./XvarG;         % Normalizing Spectral Curves to Unit Area (SNxx=Sxx/XvarG)
SNx1(:,3) = Sxx(:,1)./XvarG*Xvar;    % Spectrum with statistical variance as area
SNx1(:,4) = Sxx(:,1);                % Spectrum as calculated with pwelch (geometric variance underneath)



% Performance of digital filtering
if DoAct3 == 1
    Nyquist = Fsamp/2;
    [b,a]   = butter(Fn,CutOff/Nyquist,Ftype);
    Xf      = filtfilt(b,a,X1);
    X2      = Xf;
else
    X2      = X1;
end

%    The stochastic process is now saved as X2(t) on which all subsequent
%    analysis will be peformed.
%
%    3.3 Statistical Parameter of Time History (X2):
%    ----------------------------------------------
%    Echo print on print is default

Xmean     = mean(X2);               % mean value of the parent time history
Xstd      = std(X2);                % standard deviation of the parent time history
Xvar      = Xstd^2;                 % corresponding variance
Xmax      = max(X2);                % maximum peak value occuring in parent time history
Xmin      = min(X2);                % corresponding minimum peak value
Tend      = TAx(m1);                % duration of parent time history

%    3.4 Statistics on Sub-Series:
%    ----------------------------
%
```

THA5.m

```
if DoAct2==0; Nsub=1; end

Lsub = floor(m1/Nsub);     % Length of sub-series in number of time steps

Smean = zeros(Nsub,1);
Srms  = zeros(Nsub,1);
Sx    = zeros(Nsub,1);
St    = zeros(Nsub,1);
Smin  = zeros(Nsub,1);
Smax  = zeros(Nsub,1);
SXmin = zeros(Nsub,1);
SXmax = zeros(Nsub,1);

for k=1:Nsub
    Smin(k) = +10E10;  %Value to compare with data points in sub-series to identify sub-minimum.
    Smax(k) = -10E10;  %Value to compare with data points in sub-series to identify sub-maximum.
    k1 = 1+(k-1)*Lsub; % Step number where sub-series starts
    k2 = k*Lsub;       % Step number where sub-series ends
    Smean(k)= mean(X2(k1:k2));  % mean value of sub-series
    Srms(k) = std(X2(k1:k2));   % standard deviation of sub-series

    if DoAct2==1
        for j=k1:k2
            if X2(j)<=Smin(k) ; Smin(k)=X2(j) ; SXmin(k) = TAx(j) ; end
            if X2(j)>=Smax(k) ; Smax(k)=X2(j) ; SXmax(k) = TAx(j) ; end
        end
    end
    Sx(k) = k;                  % Number of sub-series
    St(k) = TAx(k*(Lsub-1));    % Location of sub-series boundaries on time axis
end


%    3.5 Calculating a histogram and probability density on detrended data:
%    ---------------------------------------------------------------------

RangeX = (Xmax-Xmin)*1.03;        % Expanding range about 3%
BinLow = Xmin-0.03*(Xmax-Xmin)/2; % Lower start point for bin grid
DBin   = RangeX/Nbin;             % Bin width

% Generating vector with Nbin+1 bin boundaries:
BIN = zeros(Nbin+1,1);
BIN(1) = BinLow;
for i=1:Nbin
    BIN(i+1)=BinLow+i*DBin;
end

% Counting data points per bin:
BinCount=zeros(Nbin,1);
for i=1:Nbin
    BinL=BIN(i);
    BinU=BIN(i+1);
    for j=1:m1
        if(X2(j)>BinL)&&(X2(j)<=BinU)
            BinCount(i)=BinCount(i)+1;
        end
    end
end

% Conversion to relative bin frequency
RelFreq=zeros(Nbin,1);
for i=1:Nbin
    RelFreq(i)=BinCount(i)/(m1*DBin);
end

Dx = DBin/10;
i=0;
for x = Xmin:Dx:Xmax
    i=i+1;
    pdf(i,1) = x;
    pdf(i,2) = 1/(Xstd*sqrt(2*pi))*exp(-0.5*((x-Xmean)/Xstd)^2);
end

%
%    Printing basic parameter of the analysis
%    ----------------------------------------

fprintf(1,'TIME HISTORY of Variable X\n');
fprintf(1,'  Number of time steps in  time history: %10.0f [-]\n',m1);
fprintf(1,'  Duration of parent time history      : %10.2f [s]\n',Tend);
fprintf(1,'  Number of sub-series                 : %10.4g [s]\n',Nsub);
fprintf(1,'  Duration of sub-series               : %10.2f [s]\n',Tend/Nsub);
fprintf(1,'  Time step width DT                   : %10.4g [s]\n',DT);
fprintf(1,'  Sample frequency  (if [T]=s)         : %10.4g [Hz]\n',Fsamp);
fprintf(1,'  Mean value of X(t)                   : %10.4g [x]\n',Xmean);
fprintf(1,'  Standard deviation of X(t)           : %10.4g [x]\n',Xstd);
fprintf(1,'  Corresponding variance               : %10.4g [x^2]\n',Xvar);
fprintf(1,'  Maximum peak value in X(t)           : %10.4g [x]\n',Xmax);
fprintf(1,'  Minimum peak value in X(t)           : %10.4g [x]\n',Xmin);
fprintf(1,' \n');
fprintf(1,'SPECTRAL DENSITY parameters for Sxx\n');
fprintf(1,' Number of overlapping sub-windows     : %10.0f [-]\n',Nw);
fprintf(1,' Sub-window length                     : %10.0f [-]\n',window);
fprintf(1,' Filter depth of fft-routine           : %10.0f [-]\n',nfft);
fprintf(1,' \n');
fprintf(1,' \n');

%    3.6 Spectral Density of X(t):
%    -----------------------------

X2d       = detrend(X2,'constant'); % Removing mean value before performing FFT
[Sxx,f]   = pwelch(X2d,window,[],nfft,fs);
XvarG     = trapz(f(:),Sxx(:,1));   % Area underneath calcultaed SDF-curve (geometrical variance)
SNx2(:,1) = Sxx(:,1).*f./XvarG;     % Ordinate normalised Spectral Curves
```

THA5.m

```
SNx2(:,2) = Sxx(:,1)./XvarG;         % Normalizing Spectral Curves to Unit Area (SNxx=Sxx/XvarG)
SNx2(:,3) = Sxx(:,1)./XvarG*Xvar;    % Spectrum with statistical variance as area
SNx2(:,4) = Sxx(:,1);                % Spectrum as calculated with pwelch (geometric variance underneath)

%
%    3.7 Saving modified data to external file:
%    ------------------------------------------

if DoAct4 ==1
    fid1 = fopen('Data2.dat','w');
    for i=1:m1
        fprintf(fid1,' %g %g\n',TAx(i),X2(i));
    end
    status = fclose(fid1);
end

%=========================================================================================
% 4) GRAPHICAL DISPLAY OF THE EXTREME VALUE ANALYSIS:
%---------------------------------------------------
% Display Definitions
% -------------------

scrsz = get(0,'ScreenSize');

%=====================================================================================
figure('Name','Time History','Position',[5 0.50*scrsz(4) 0.7*scrsz(3) 0.45*scrsz(4)])

% Setting axis range

Y1max = Xmin+(Xmax-Xmin)*1.2;
Y1min = Xmin-(Xmax-Xmin)*0.02;
DY   = Y1max-Xmax;

X1min = TAx(1);
X1max = TAx(m1);

if Displ == 2
    plot(TAx,X1,'-c');
    hold on
end

if DoAct2==1
    for j=1:Nsub
        plot(SXmax(j),Smax(j),'o','MarkerEdgeColor','k','MarkerFaceColor','c','MarkerSize',5);hold on
        plot(SXmin(j),Smin(j),'o','MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',5);hold on
    end
end

plot(TAx,X2);
hold on

plot3([TAx(1),TAx(m1)],[Xmean,Xmean],[1,1],'--m');
hold on
plot3([TAx(1),TAx(m1)],[Xmean+Xstd,Xmean+Xstd],[1,1],'--g');
hold on
plot3([TAx(1),TAx(m1)],[Xmean-Xstd,Xmean-Xstd],[1,1],'--g');
hold on

plot([0 0],[Y1min Y1max],':k');
for j=1:Nsub
    plot([St(j) St(j)],[Y1min Y1max],':k');
end

text(Tend/20,Y1max-0.3*DY,['Time History File is "',FileName,'"',' ; Signal No.:',num2str(SignalNo)],'FontSize',9)
text(Tend/20,Y1max-0.7*DY,['Time History duration is ',num2str(Tend),' sec'],'FontSize',9)

text(0.98*Tend,Y1max-0.3*DY,['Mean value of X(t): ',num2str(Xmean)],'FontSize',9,'HorizontalAlignment','right')
text(0.98*Tend,Y1max-0.7*DY,['Standard deviation of X(t): ',num2str(Xstd)],'FontSize',9,'HorizontalAlignment','right')

xlabel('time [s]');
ylabel('ordinate of variable X(t)');
title('TIME HISTORY ANALYSIS');

axis([X1min X1max Y1min Y1max]);

eval(['print -dtiff -zbuffer TimeHist']);

%=====================================================================================
figure('Name','Histogram','Position',[0.715*scrsz(3) 0.50*scrsz(4) 0.29*scrsz(3) 0.45*scrsz(4)])

XX=1.05*max(RelFreq);

for i=1:Nbin
    area([0 RelFreq(i) RelFreq(i) 0],[BIN(i) BIN(i) BIN(i+1) BIN(i+1)],'FaceColor',[.7 0 0]);
    hold on
end

plot([0 XX],[Xmean Xmean],'--m'); hold on
plot([0 XX],[Xmean+Xstd Xmean+Xstd],'--g'); hold on
plot([0 XX],[Xmean-Xstd Xmean-Xstd],'--g'); hold on

plot(pdf(:,2),pdf(:,1),'-b','LineWidth',2); hold on

title('HISTOGRAM');
xlabel('relative frequency');
ylabel('Data value');

axis([0 XX Y1min Y1max]);

eval(['print -dtiff -zbuffer Histogram']);

%=====================================================================================
figure('Name','Spectral Density','Position',[5 35 0.33*scrsz(3) 0.395*scrsz(4)])

```

THA5.m

```
if Displ == 2
    loglog(f,SNx1(:,1),'.','MarkerSize',5,'Color','c');
    hold on
end

loglog(f,SNx2(:,1),'.','MarkerSize',5,'Color','b');
hold on

title('SPECTRAL DENSITY of X(t)');
xlabel('frequency [Hz]');
ylabel('Normalised Spectrum S_x_x(f) * f / \sigma_x^2');
grid on

eval(['print -dtiff -zbuffer SDF']);

%=================================================================================
figure('Name','Spectral Density 2','Position',[5 35 0.33*scrsz(3) 0.395*scrsz(4)])

loglog(f,SNx2(:,2),'.','MarkerSize',5,'Color','b');
hold on

title('SPECTRAL DENSITY of X(t)');
xlabel('frequency [Hz]');
ylabel('Normalised Spectrum S_x_x(f) / \sigma_x^2');
grid on

eval(['print -dtiff -zbuffer SDF2']);

%=================================================================================
figure('Name','Spectral Density 3','Position',[5 35 0.33*scrsz(3) 0.395*scrsz(4)])

loglog(f,SNx2(:,3),'.','MarkerSize',5,'Color','b');
hold on

title('SPECTRAL DENSITY of X(t)');
xlabel('frequency [Hz]');
ylabel('Spectrum S_x_x(f)');
grid on

eval(['print -dtiff -zbuffer SDF3']);

%=================================================================================
figure('Name','Spectral Density 4','Position',[5 35 0.33*scrsz(3) 0.395*scrsz(4)])

loglog(f,SNx2(:,4),'.','MarkerSize',5,'Color','b');
hold on

title('SPECTRAL DENSITY of X(t)');
xlabel('frequency [Hz]');
ylabel('Spectrum S_x_x(f)');
grid on

eval(['print -dtiff -zbuffer SDF4']);


%=================================================================================
figure('Name','SubSeries Parameters','Position',[0.34*scrsz(3) 35 0.33*scrsz(3) 0.395*scrsz(4)])

plot(Sx,Smean,'s','MarkerEdgeColor','k','MarkerFaceColor','m','MarkerSize',7); hold on
plot(Sx,Srms,'o','MarkerEdgeColor','k','MarkerFaceColor','g','MarkerSize',7); hold on

title('SUB-SERIES PARAMETERS');
xlabel('number of sub-series');
ylabel('Mean and rms value');
legend('mean','rms','Location','Best');

plot([0 Nsub],[Xmean Xmean],'--m'); hold on
plot([0 Nsub],[Xstd Xstd],'--g'); hold on


grid on

eval(['print -dtiff -zbuffer SubSeriesParam']);
```

4.2 "TSCorr.m"


```
prog='TSCorr';
%-------------------------------------------------------------------------------------
%                 TIME SERIES CORRELATION - relation between two time signals
%=====================================================================================
%
% DESCRIPTION:      This program has been developed to view and analyse a time series of
% ------------      a measured signal. The main features are:
%
%                   - viewing the time steries as a graph.
%                   - presentation of data points as histogram (discrete probability density).
%                   - calculation of power spectral density.
%                   - calculation and displaying the mean values and standard deviations of sub-series.
%                   - Identification of maximum and minimum in sub-series.
%
%                   Furthermore, some signal processing can be performed on the original data,
%                   namely detrending and high or low-pass filtering. The difference between the
%                   initial and the altered signal can be visualised in the graph of the time
%                   series and the spectral density the spectral density.
%
% Program ID:
% -----------
% File name       : THA4.m
% Author          : Holger Koss (hko)
% Development Log : 2009-04-28  hko   Basic structure of the program
%                   2012-05-04  hko   Adoption to a general tool to first analysis and
%                                     treatment of a time history
%                   2012-10-29  hko   Adoption for course 11374
%
% Necessary files  : "TimeHistory"  - Ascii file containing in the first column the
%                                     time axis and in the second column the time history
%                                     of the investigated quantity. Both columns are in
%                                     model scale.
%
%                   "cpcent00.dat"  - File with 18 time series of pressure coefficients
%                                     measured in a wind tunnel test on a model low-rise
%                                     building.
%=====================================================================================
 close all
 clear all
%=====================================================================================
% 1) READING OF INPUT DATA:
%    ----------------------
%    This version of the program is prepared to read a file with a matrix
%    of 18 time series of measured pressure coefficients. In this case the
%    time series needs to be generated to plot the series.
%

% 1.1) Reading Data from Pressure Measurements on Low-rise Building:
%      -------------------------------------------------------------
%      Here, we read a file with 18 time series of pressure coefficients
%      measured on a wind tunnel model of a low-rise building. Each of these
%      time series is a signal of a stochastic process. The input file does
%      not contain the time axis. To plot the process the time axis needs
%      to be generated separately.
%


FileName = '18Signals.dat';             % Name of input file
SignalA = 1;                            % Number of 1st signal to be compared (1-18)
SignalB = 18;                           % Number of 2nd signal to be compared (1-18)
fid      = fopen(FileName,'r');
qhmwk    = fscanf(fid,'%e',[1 1]);      % velocity pressure [kN/m^2]
cp       = fscanf(fid,'%e',[18 inf]);   % Matrix with pressure coefficient time series
Series   = cp';                         % Transposed matrix where each column is a signal
[m1 n1]  = size(Series);                % Number of rows (m1) and columns (n1)
status   = fclose(fid);

Fsamp    = 1600;                        % Sample frequency in [Hz]
DT       = 1/Fsamp;                     % Calculation of time step
for i=1:m1                              % Generation of time axis with m1 steps
    TAx(i) = (i-1)*DT;
end

XA = Series(:,SignalA);                 % Saving selected data to input vector
XB = Series(:,SignalB);                 % Saving selected data to input vector


% NOTE: At this point in the program you should have following information available:
%
%       Fsamp    = sample frequency in [Hz]
%       DT       = time step between data points [s] = 1/Fsamp
%       SignalNo = Number of signal that has been chosen to be analysed - saved as X0(i)
%       m1       = number of data points (time steps)
%       NFFTcase = FFT filter length, determine by trial, shall not exceed m1
%       TAx(i)   = Vector with values for time axis (length = m1)
%       X0(i)    = Vector with data of stochastic process (length = m1)
%
%       Disregarding what data you want to analyse, just make sure that after reading the
%       input file you define the parameters and vectors listed above.

%=====================================================================================
% 2) ANALYSIS SETTING:
%    -----------------
%    The stochastic process is now saved as X2(t) on which all subsequent
%    analysis will be peformed.
%
%    2.1 Statistical Parameter of 1st Time History (XA):
%    ---------------------------------------------------
```

TSCorr.m

```
XAmean    = mean(XA);              % mean value of the parent time history
XAstd     = std(XA);               % standard deviation of the parent time history
XAvar     = XAstd^2;               % corresponding variance
XAmax     = max(XA);               % maximum peak value occurring in parent time history
XAmin     = min(XA);               % corresponding minimum peak value
Tend      = TAx(m1);               % duration of (both!) time histories

%    2.2 Statistical Parameter of 2nd Time History (XB):
%    ---------------------------------------------------

XBmean    = mean(XB);              % mean value of the parent time history
XBstd     = std(XB);               % standard deviation of the parent time history
XBvar     = XBstd^2;               % corresponding variance
XBmax     = max(XB);               % maximum peak value occurring in parent time history
XBmin     = min(XB);               % corresponding minimum peak value

%    2.3 Statistical Parameter of 2nd Time History (XB):
%    ---------------------------------------------------

ABcorr = corr(XA,XB);

% R = corrcoef(X) returns a matrix R of correlation coefficients calculated
% from an input matrix X whose rows are observations and whose columns are variables
Mcorr  = corr(Series);


disp(['Data read from file:',FileName])
fprintf(1,' \n');
fprintf(1,' Correlation coefficient: %g\n',ABcorr);


%=========================================================================================
% 3) GRAPHICAL DISPLAY OF THE EXTREME VALUE ANALYSIS:
%----------------------------------------------------
% Display Definitions
% -------------------

scrsz = get(0,'ScreenSize');

%=====================================================================================
figure('Name','Time History A','Position',[5 0.50*scrsz(4) 0.7*scrsz(3) 0.45*scrsz(4)])

% Setting axis range

Y1max = XAmin+(XAmax-XAmin)*1.2;
Y1min = XAmin-(XAmax-XAmin)*0.02;
DY1   = Y1max-XAmax;

Ex    = floor(log10(Tend));
X1min = 0.;
X1max = ceil(Tend/(10^Ex))*10^Ex;

plot(TAx,XA,'-b');
hold on

plot3([TAx(1),TAx(m1)],[XAmean,XAmean],[1,1],'--m');
hold on
plot3([TAx(1),TAx(m1)],[XAmean+XAstd,XAmean+XAstd],[1,1],'--g');
hold on
plot3([TAx(1),TAx(m1)],[XAmean-XAstd,XAmean-XAstd],[1,1],'--g');
hold on

text(Tend/20,Y1max-0.3*DY1,['Time History File is "',FileName,'"',' ; Signal No.:',num2str(SignalA)],'FontSize',9)
text(Tend/20,Y1max-0.7*DY1,['Time History duration is ',num2str(Tend),' sec'],'FontSize',9)

text(0.98*Tend,Y1max-0.3*DY1,['Mean value of XA(t): ',num2str(XAmean)],'FontSize',9,'HorizontalAlignment','right')
text(0.98*Tend,Y1max-0.7*DY1,['Standard deviation of XA(t): ',num2str(XAstd)],'FontSize',9,'HorizontalAlignment','right')

xlabel('time [s]');
ylabel('ordinate of variable XA(t)');
title('1st TIME HISTORY in COMPARISON');

axis([X1min X1max Y1min Y1max]);

eval(['print -dtiff -zbuffer TimeHistA']);

%=====================================================================================
figure('Name','Time History B','Position',[5 35 0.7*scrsz(3) 0.45*scrsz(4)])

Y2max = XBmin+(XBmax-XBmin)*1.2;
Y2min = XBmin-(XBmax-XBmin)*0.02;
DY2   = Y2max-XBmax;

Ex    = floor(log10(Tend));
X2min = 0.;
X2max = ceil(Tend/(10^Ex))*10^Ex;

plot(TAx,XB,'-b');
hold on

plot3([TAx(1),TAx(m1)],[XBmean,XBmean],[1,1],'--m');
hold on
plot3([TAx(1),TAx(m1)],[XBmean+XBstd,XBmean+XBstd],[1,1],'--g');
hold on
plot3([TAx(1),TAx(m1)],[XBmean-XBstd,XBmean-XBstd],[1,1],'--g');
hold on

text(Tend/20,Y2max-0.3*DY2,['Time History File is "',FileName,'"',' ; Signal No.:',num2str(SignalB)],'FontSize',9)
text(Tend/20,Y2max-0.7*DY2,['Time History duration is ',num2str(Tend),' sec'],'FontSize',9)

text(0.98*Tend,Y2max-0.3*DY2,['Mean value of XB(t): ',num2str(XBmean)],'FontSize',9,'HorizontalAlignment','right')
text(0.98*Tend,Y2max-0.7*DY2,['Standard deviation of XB(t): ',num2str(XBstd)],'FontSize',9,'HorizontalAlignment','right')

```

TSCorr.m

```
xlabel('time [s]');
ylabel('ordinate of variable XB(t)');
title('2nd TIME HISTORY in COMPARISON');

axis([X1min X1max Y2min Y2max]);

eval(['print -dtiff -zbuffer TimeHistB']);

%=================================================================================
figure('Name','Correlation','Position',[0.715*scrsz(3) 0.50*scrsz(4) 0.29*scrsz(3) 0.45*scrsz(4)])


plot(XA,XB,'.b','MarkerSize',3)
hold on

%
% Calculation of coordinate values for graph design

X1 = min(XA);
X2 = max(XA);
Y1 = min(XB);
Y2 = max(XB);
DX = abs((X2-X1)/20);
DY = abs(Y2-Y1)/20;

%
% Calculating the line coordinates for full correlation (corr=1) with
% refernce in intersection point of both mean values.
%

plot ([X1 X2],[XBmean-(XAmean-X1) XBmean+(X2-XAmean)],'-k');
hold on
%plot ([X1 X2],[XBmean+(XAmean-X1) XBmean-(X2-XAmean)],'-.k');
plot ([XAmean+(XBmean-Y1) XAmean-(Y2-XBmean)],[Y1 Y2],'-.k');
hold on

title('CORRELATION');
xlabel('signal A');
ylabel('signal B');

text(X1+DX,max(XB)-DY,['Corr = ',num2str(ABcorr)],'FontSize',9)

%
% Plotting the MEAN value and STANDARD DEVIATION of 1st signal XA
%
plot3([XAmean,XAmean],[Y1,Y2],[1,1],'--m');
hold on
plot3([XAmean+XAstd,XAmean+XAstd],[Y1,Y2],[1,1],'--g');
hold on
plot3([XAmean-XAstd,XAmean-XAstd],[Y1,Y2],[1,1],'--g');
hold on

%
% Plotting the MEAN value and STANDARD DEVIATION of 2nd signal XB
%
plot3([X1,X2],[XBmean,XBmean],[1,1],'--m');
hold on
plot3([X1,X2],[XBmean+XBstd,XBmean+XBstd],[1,1],'--g');
hold on
plot3([X1,X2],[XBmean-XBstd,XBmean-XBstd],[1,1],'--g');
hold on


axis([X1 X2 Y1 Y2]);
axis equal;

eval(['print -dtiff -zbuffer Correlation']);
```

4.3 “JPDF.m”


```
prog='JPDF'
%-----------------------------------------------------------------------------------------
%          Visualisation of Interaction between two independent Variables
%          ==============================================================
%
% DESCRIPTION:
% ------------
% This program calculates and visualises the joint probability between two
% independent variable. Since the here treated topics fall in the area of
% Wind Engineering on variable will quite likely be the wind. In priciple
% both variables can individually be defined.
%
% Program ID:
% -----------
% File name       : JPDF.m
% Author          : hko
% Development Log : 2011-01-24  hko   Basic structure of the program
%                   2013-03-14  hko   Visualisation of JPDF in four graphs
%                                     Decision function and JP over decision area.
%
%
%=========================================================================================
 close all
 clear all
%=========================================================================================
% 1) DEFINITIONS AND CENTRAL ADJUSTMENTS OF THE PROGRAM:
%    ---------------------------------------------------
% 1.1) Description of Text for Plots:
%      ------------------------------------------

pi     = 4*atan(1.);

DoPDF  = 1; % switch for plotting the PROJECTED shape of PDFs for both
            % variables on the side walls of the 3D graph.

% 1.2) Distribution densities of variables:
%      ------------------------------------
%      M1, M2 = mean value of both variables
%      S1, S2 = standard deviation of both variables

% Variable 1: Daily Mean Wind Speeds (10-minutes mean)
%             2-parametric Weibull Distribution

A = 5;
k = 2;

% Variable 2: Air Temperatures (degC)
%             Normal Distribution

M2 = 15;
S2 = 6;


%=========================================================================================
% 2) CONTRUCTION OF DISTRIBUTION DENSITIES:
%    -------------------------------------
% 2.1 Definition of calculation settings

NU = 100;   % discretisation of the velocity axis 0-30m/s in 0.3m/s steps
NT = 100;   % discretisation of temperature axis -10 to 40degC in 0.5degC steps
dU = 0.2;   % wind velocity step width [m/s]
dT = 0.5;   % air temperature step width [degC]

U0 = 0;     % lowest wind speed [m/s]
T0 = -10;   % lowest air temperature [degC]

U    = zeros(1,NU);  % vector for wind speed range
T    = zeros(1,NT);  % vector for airtemperature range
R    = zeros(NT,NU); % Result matrix
D    = zeros(NT,NU); % Decision matrix
pdfU = zeros(1,NU);  % PDF vector for wind velocity
pdfT = zeros(1,NT);  % PDF vector for air temperature
cdfU = zeros(1,NU);  % CDF vector for wind velocity
cdfT = zeros(1,NT);  % CDF vector for air temperature

% 2.2 Probility Densities Functions (PDF)of individual Variables
% ---------------------------------------------------------------

for i=1:NU   % Wind Density Distribution
    U(i)    = U0+(i-1)*dU;
    pdfU(i) = k/A*(U(i)/A)^(k-1)*exp(-(U(i)/A)^k);
end

cdfU(1) = pdfU(1)*dU;
for i=2:NU
    cdfU(i)=pdfU(i)*dU+cdfU(i-1);
end

for i=1:NT   % Temperature Density Distribution
    T(i)    = T0+(i-1)*dT;
    pdfT(i) = 1/(S2*sqrt(2*pi))*exp(-0.5*((T(i)-M2)/S2)^2);
end

cdfT(1) = pdfT(1)*dT;
for i=2:NT
    cdfT(i)=pdfT(i)*dT+cdfT(i-1);
end
```

JPDF.m

```
% 2.3 Joint Probability Matrix R
% -----------------------------

for i=1:NU
   for j=1:NT
      R(j,i)=pdfU(i)*pdfT(j);
   end
end

% 2.4 Decision function and probability of decision area
% -----------------------------------------------------

JP  = 0; % Value of jount probability in decision area
JPt = 0; % Value of total joint probability

for i=1:NU
   for j=1:NT
      JPt = JPt+R(j,i)*dU*dT;
      if (U(i)>=6)&&(T(j)<=2)       % DECISION FUNCTION
         D(j,i) = 1.;
         JP     = JP+R(j,i)*dU*dT;
      end
   end
end

% Check for probability beyond decision point in individual PDFs:

pU6=0;
for i=1:NU
    if U(i)>=6;pU6=pU6+pdfU(i)*dU;end
end

pT2=0;
for i=1:NT
    if T(i)<=2;pT2=pT2+pdfT(i)*dT;end
end

fprintf(1,'Probability of u>=6m/s                   : %7.5f [-]\n',pU6)
fprintf(1,'Probability of T<=2degC                  : %7.5f [-]\n',pT2)
fprintf(1,'Joint probability                        : %7.5f [-]\n',pU6*pT2)
fprintf(1,' \n')
fprintf(1,'Value of total joint probability         : %7.5f [-]\n',JPt)
fprintf(1,'Value of decision space joint probability: %7.5f [-]\n',JP)


%=========================================================================================
% 3) GRAPHICAL DISPLAY OF THE EXTREME VALUE ANALYSIS:
%----------------------------------------------------
% Display Definitions
% ------------------

scrsz = get(0,'ScreenSize');

%=========================================================================================

figure('Name','3D Joint Probability Density','Position',[5 0.40*scrsz(4) 0.5*scrsz(3) 0.5*scrsz(4)])

R(1,1)=-0.000001; % This point in the joint probability matrix is assigned
                  % artificially with a negative value to activate the
                  % offset for the isolines below the 3D graph.
surfc(U,T,R,'EdgeColor','none');hold on

if DoPDF==1
    X1 = zeros(1,NT); X1 = X1+U(NU);
    Y1 = zeros(1,NU); Y1 = Y1+T(NT);
    fact1 = max(pdfU);
    fact2 = max(pdfT);
    plot3(X1,T,pdfT*fact1,'--r','MarkerSize',1); hold on
    plot3(U,Y1,pdfU*fact2,'--b','MarkerSize',1); hold on
end


view(-38,18);
title('Joint Probability Density');
xlabel('wind speed [m/s]');
ylabel('air temperature [degC]');
zlabel('probability of occurrence [-]');

eval(['print -dtiff -zbuffer JointProbability3D']);

%-----------------------------------------------------------------------------------------

figure('Name','Probability Isolines','Position',[0.515*scrsz(3) 0.4*scrsz(4) 0.48*scrsz(3) 0.5*scrsz(4)])
contour(U,T,R,20)
hold on
for i=1:NU
   for j=1:NT
      if D(j,i)==1
         plot(U(i),T(j),'.b','MarkerSize',2); hold on
      end
   end
end

title('Isolines of Joint Probability');
xlabel('wind speed [m/s]');
ylabel('air temperature [degC]');
grid on

eval(['print -dtiff -zbuffer JPDisolines']);

%-----------------------------------------------------------------------------------------

figure('Name','Distribution Density: Wind Speed','Position',[5 0.06*scrsz(4) 0.3*scrsz(3) 0.24*scrsz(4)])
plot(U,pdfU,'-b','LineWidth',2)
```

```
xlabel('wind speed [m/s]');
ylabel('probability of occurrence [-]');
grid on

eval(['print -dtiff -zbuffer pdfX1']);

%---------------------------------------------------------------------------------------

figure('Name','Distribution Density: Air temperature','Position',[0.316*scrsz(3) 0.06*scrsz(4) 0.3*scrsz(3) 0.24*scrsz(4)])
plot(T,pdfT,'-r','LineWidth',2)
xlabel('air temperature [degC]');
ylabel('probability of occurrence [-]');
grid on

eval(['print -dtiff -zbuffer pdfX2']);
```

5. References

Abbe, C., 1888. *Treatise on Meteorological Apparatus and Methods*. Annual Report of the Chief Signal Officer for 1887, Washington D.C.

Multhauf, R.P., 1961. *The Introduction of Self-Registering Meteorological Instruments*. United States National Museum, Bulletin 228, Smithsonian Institution, Washington D.C. [online] Available at The Project Gutenberg Ebook < http://www.gutenberg.org/files/32482/32482-h/32482-h.htm> [Accessed 19 November 2012]

Moriarty, W.W., 1985. *On the Directional Uncertainty of Strong Wind Gusts*. Journal of Wind Engineering and Industrial Aerodynamics, 21, pp.166 155

Wikipedia, 2013. *Correlation and dependence*. [online] Available at: < http://en.wikipedia.org/wiki/Correlation > [Accessed November 10, 2013]

DTU Civil Engineering
Department of Civil Engineering
Technical University of Denmark

Brovej, Building 118
2800 Kgs. Lyngby
Telephone 45 25 17 00

www.byg.dtu.dk