```
% README.m
% Information for files illustrating Sound & Vibration article:
% Brandt & Ahlin, Sampling and Time-Domain Analysis, May 2010.
%
% Installation:
% Unpack the files into a directory and go inside MATLAB (or Octave if you
% are using that)
% Make this directory your working directory
%
% Run the following examples (look inside each example m file for comments)
% 1. dataqex.m      Data quality, Example 1
% 2. interpex.m     Interpolation example, similar to that used in article
% 3. octshapes.m    Third octaves, Example 2
```

```
function K = akurtosis(x);
% AKURTOSIS     Calculates kurtosis of (columns in) x
%
%       K = akurtosis(x);
%
%       K           kurtosis (third moment over sigma^3)
%
%       x           Time data, if more than one column, each column is
%                   computed and S is a row vector
%
% Note: The kurtosis here is defined so that it equals exactly 3 for
% Gaussian data (normal distribution).
%


% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

Sigma=std(x);
K=amoment(x,4)./(Sigma.^4);
```

```
function M = amoment(x,n);
% AMOMENT   Calculate central statistical moments of columns in x
%
%           M = amoment(x,n);
%
%           M           Central moment of order n (mean of x removed)
%
%           x           Time signal(s), in column(s)
%           n           Statistical order (2 for variance, ...)


% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk


m=mean(x);
M=zeros(1,length(m));
for col=1:length(m)
   M(col)=mean((x(:,col)-m(col)).^n);
end
```

```
function [Pdf, XAx, G, mu, sigma] = apdf(x, N, NPlot);
% APDF      Calculate and plot probability density function, PDF
%
%           [Pdf, XAx, G, mu, sigma] = gaussian(x, N, NPlot);
%
%           Pdf         Probability density function of x
%           XAx         Amplitude (x-) axis for Pdf
%           G           Gaussian distribution with same mu and sigma as x
%           mu          mean of x
%           sigma       Standard deviation of x
%
%           x           Time data, in column vector
%           N           Number bins in histogram (number bars in Pdf)
%           NPlot       If = 0, no plot is produced
%                       If = 1, a (lin-lin) plot with G is produced,
%                       If = 2, a semilogy plot with G is produced,
%                       which is often better for classification
%                       purposes (to check if x is Gaussian or not)
%
% Calculates and plots the PDF of x using N bins, and if NPlot=1 or NPlot=2
% also plots the "theoretical" Gauss curve using mean and standard
% deviation of x. The scaling is so that the surface under Pdf = 1.
%
% With no input parameters, apdf() plots a theoretical Gauss curve for a
% normalized variable with zero mean and standard deviation of unity,
% z=(x-mu)/sigma.
%
% [Pdf, Xax, G]=apdf(x,N) returns the histogram in Pdf and
% the gauss curve at the center values of Pdf, in G and x axis in XAx.
% If N is not given, a default of 30 bins is used.
%---------------------------------------------------------


% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

if nargout == 0
   z=(-5:0.01:5)';
   pz=exp(-z.^2/2)/sqrt(2*pi);
   plot(z,pz)
   ylabel('p(z)')
   xlabel('standardized variable z')
elseif nargout >= 2                  % at least Pdf and XAx requested
    if nargin == 1
        N=30;
        NPlot=0;
    elseif nargin == 2
        NPlot=0;
    elseif nargin ~= 3
        error('Too many input parameters!')
    end
    mu=mean(x);
    sigma=std(x);
    XMax=max(x);
    XMin=min(x);
    XMax=max(abs([XMax XMin]));
    XAx=(-XMax:XMax*2/N:XMax)';
    dx=XAx(2)-XAx(1);
    Pdf=hist(x,XAx);
    Pdf=Pdf(:)/length(x)/dx;                 % Force to column and
    G=1./(sqrt(2*pi)*sigma).*exp(-0.5*((XAx-mu)./sigma).^2);
    if NPlot == 1
```

```
        figure
        bar(XAx,Pdf);
        if NPlot == 1
            hold on
            G=G;
            plot(XAx,G,'r:')
            hold off
        end
        xlabel('Units of x')
        ylabel('Probability Density')
    elseif NPlot == 2
        figure
        semilogy(XAx,[Pdf G])
        xlabel('Units of x')
        ylabel('Probability Density')
    end
end
```

```
function S = askewness(x);
% ASKEWNESS     Calculates skewness of (columns in) x
%
%       S = askewness(x);
%
%       S           Skewness (third moment over sigma^3)
%
%       x           Time data, if more than one column, each column is
%                   computed and S is a row vector
%
%


% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

Sigma=std(x);
S=amoment(x,3)./(Sigma.^3);
```

```
% Data quality example
% See article in Sound & Vibration:
% Brandt & Ahlin, Sampling and Time-Domain Analysis, May 2010.
%
% This is an example of computing and displaying statistical measures in
% order to assess the quality of a set of measurements. This example uses
% synthesized data, as opposed to the plot in the % article that was
% produced based on a real case.

% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

clear
close all

warning off
% Produce 8 channels of (Gaussian) random data
x=randn(50000,8);       % 50,000 samples on each 'channel'
fs=5000;                % Used for scaling only
% Now 'contaminate' channels 3 and 5 with a sine and a triangle wave,
% respectively, to yield a kurtosis other than 3
t=(0:1/fs:(length(x(:,1))-1)/fs)';  % Time axis for disturbances
x(:,3)=x(:,3)+10*sin(2*pi*50*t);   % 50 Hz disturbance
T=square(2*pi*0.5*t);
T=timeint(T,fs);
x(:,5)=x(:,5)+5*T;

% Produce standard statistics and print on screen
warning off         % Warning issued about no. channels > 1, see statchk
statchk(x,50,1,'DataQual')      % This saves results in file DataQual.mat
fprintf('printing file DataQual.log to screen:\n\n')
type DataQual.log
fprintf('\n\n')
warning on

% Plot normalized kurtosis, normalized by channel 1
figure
load DataQual
bar(Kurtosis/Kurtosis(1));
title('Kurtosis Normalized to Channel #1')
xlabel('Channel')

% Apparently channels 3 and 5 are not like the others (surprise!)
% Plot to see what they look like
figure
plot(t,x(:,3))
title('Channel 3 - zoom in to see!')
xlabel('Time [s]')
figure
plot(t,x(:,5))
title('Channel 5')
xlabel('Time [s]')
```

```
% Demo script to illustrate interpolation.
% See Sound & Vibration article:
% Brandt & Ahlin, Sampling and Time-Domain Analysis, May 2010.
%
% This example is similar to Example 1 in file filterex.m, but gives more
information.

% The example produces lowpass filtered Gaussian random noise, with an
% oversampling rate of 10. Then it resamples this data to 4 times lower
% sampling frequency, i.e. to an oversampling ratio of 2.5 (still
% fulfilling the sampling theorem!). From those new data points, the signal
% is then upsampled to the original sampling frequency, and the original
% and the new, resampled signal are compared. The results show a good
% agreement, illustrating that interpolation works.

% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

clear
close all

% Produce 'original' random noise, oversampled by a factor of 10, i.e. the
% sampling frequency is 10 times the bandwidth of the data.
% If you zoom in on the data in this plot, you will find that the curve is
% smoothly moving up and down, as you would suspect an accelerometer signal
% to behave.
x=randn(10000,1);              % Data vector
fs=1000;
t=(0:1/fs:(length(x)-1)/fs)';
% Low pass filter data to 10 times oversampling
fc=100;
wc=0.1;
[B,A] = ellip(8,0.1,70,wc);
x0=filtfilt(B,A,x);
plot(t,x0)
xlabel('Time [sec.]')
title(['Original data, bandwidth ' num2str(fc) ', sampling freq. ' num2str(fs) '
Hz'])

% Now resample data to a sampling frequency of 2.5 times fc. 'Resampling'
% here simply means picking every 4th sample, as we are still below the
% nyquist frequency, and there will be no aliasing.
% If you zoom in on these data, you will find they look 'sharper', similar
% to what we normally see on a data acquisition system, because these data
% are oversampled with only a factor 2.5. These data are still 'correct'
% but give no good description of the actual motion/acceleration of the
% measured structure.
x25=x0(1:4:end);
fs25=fs/4;
t25=(0:1/fs25:(length(x25)-1)/fs25)';
figure
plot(t25,x25)
xlabel('Time [sec.]')
title(['Data vector resampled at ' num2str(fs25) ' Hz'])

% Plot the original and downsampled signals with asterisks '*' and '+' signs,
% respectively, to indicate the process of decimating the data.
i25=find(t25<.2);
idx=find(t<.2);
figure
plot(t(idx),x0(idx),t(idx),x0(idx),'*',t25(i25),x25(i25),'+')
xlabel('Time [sec.]')
```

```
title('Original and downsampled signal, first 200 samples')
legend('Original','Orig. Samples','Downsampled Samples')

% Now interpolate up to original sampling and plot overlaid with the
% original samples. You should see that the two sets agree very well. If
% you look closely at the first few samples, you will notice they do not
% fit, because of end phenomena (no data to the left).
xr=resample(x25,4,1);
%xr=interp(x25,4);             % This is a different way of interpolation
%                                which is, in this case, actually a little
%                                bit more accurate. RESAMPLE is more
%                                general, though, as it works for up and
%                                downsampling alike.
figure
plot(t(idx),x0(idx),'o',t(idx),xr(idx),'+')
xlabel('Time [sec.]')
legend('Original','resampled')
title('Original and resampled signals, first 200 samples')
```

```
function [b,a] = noctfilt(n,fc,fs)
%NOCTFILT   Create filter coefficients for 1/n octave-band filter
%
%           [b,a] = noctfilt(n,fc,fs)
%
%           b       Filter coefficients as used by, e.g., FILTER command
%           a       Ditto. Can be one or more rows, see fc below.
%
%           n       Fractional octave number (for 1/n octave filter)
%           fc      Center frequency (frequencies). If length > 1, b,a will
%                   be rows in matrices with one row per center frequency.
%           fs      Sampling frequency
%
% NOTE n must be 1, 2, 3, 6, 12, or 24
%
% This command produces coefficients for a filter that conforms with
% IEC 61260:1995, and ANSI S1.11-2004, if possible. If the sampling
% frequency is unsuitable, so that the filter shape is not within the
% limits, a warning is issued.
%
% suitable ratios for fs/fc is approximately 5 < fs/fc < 500.

% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

% Check that the requested n is supported
N=[1 2 3 6 12 24];
if ~ismember(n,N)
    error('Unsupported fractional octave number');
end

G=2;                % Base two definition
OctRatio=G^(0.5/n);
fl = fc/OctRatio;
fh = fc*OctRatio;
fnyq = fs/2;                % Nyquist frequency for BUTTER command

[b,a] = butter(3,[fl/fnyq fh/fnyq]);  % Third order butterworth bandpass filter as
specified by standards.

% Check that filter corresponds to IEC/ANSI standard, or issue warning
[fu,fl,f]=noctlimits(n,fc);          % Calculate limits
[Hf,ff]=freqz(b,a,1000,fs);         % Filter frequency response, 100 values
Hf=Hf(2:end);                       % Remove zero frequency
ff=ff(2:end);
fu=interp1(f,fu,ff,'linear','extrap');               % Interpolate fu onto ff axis
fl=interp1(f,fl,ff,'linear','extrap');
if ~isempty(find(20*log10(abs(Hf))>fu)) | ~isempty(find(20*log10(abs(Hf))<fl))
    warning('Filter shape does not conform with IEC/ANSI standard. Resample data to
different sampling frequency!')
%     figure
%     loglog(ff,fl,ff,fu,ff,db20(Hf))
end
function [Lu,Ll,f] = noctlimits(n,fc);
% NOCTLIMITS    Calculate upper and lower limits of IEC/ANSI octave filter(s)
%
%           [Lh, Ll,f] = noctlimits(n,fc)
%
%           Lh          Upper limit in dB, in column vector
%           Ll          Lower limit in dB, in column vector
%           f           Frequency axis in Hz for Lh and Ll, in column
%                       vector(s). If fc contains several frequencies, each
%                        column in f corresponds to the same column in fc.
%
%           n            Fractional octave number, 1 for full octaves, 3 for 1/3 oct.
etc...
%           fc           Center frequency of 1/n octave filter. can be a row
%                        or a column vector for calculating limits for several bands
%
% This function calculates upper and lower limits of a fractional octave
% band as specified in the standards IEC 61260:1995, and ANSI S1.11-2004.
% Particularly note that the limits are always the same values, only the
% frequencies changes depending on the chosen center frequency/frequencies.
% Thus, Lh and Ll are always single columns, even if several center
% frequencies fc are calculated.

% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

% Filter class 0 is used, as there is no reason in MATLAB to make the accuracy
% less than the best.
% Base two is used (the standard gives you the option of base ten or base
% two).

% G=10^(3/10);    % This is the constant for base ten definition
G=2;    % This is the constant for base ten definition

% Calculate base frequency break points
OmegaUpper=[G^(1/8) G^(1/4) G^(3/8) G^(1/2)*(1-1e-11) G^(1/2) G G^2 G^3 G^4]-1;
OmegaUpper=1+((G^(1/2/n)-1)/(sqrt(G)-1))*OmegaUpper;

if length(fc) == 1
    f=fc*[fliplr(1./OmegaUpper) 1 OmegaUpper]';
else
    f=zeros(19,length(fc));
    for n=1:length(fc)
        f(:,n)=fc(n)*[fliplr(1./OmegaUpper) 1 OmegaUpper]';
    end
end

% Create limits
Lu=[-75 -62 -42.5 -18 -2.3 .15 .15 .15 .15 .15 .15 .15 .15 .15 -2.3 -18 -42.5 -62 -
75]';
Ll=[-inf -inf -inf -80 -4.5 -4.5 -1.1 -.4 -.2 -.15 -.2 -.4 -1.1 -4.5 -4.5 -80 -inf -
inf -inf]';
```

```
% Script showing effects of sampling freq. on (third) octave digital filter
% See Example 2 in Sound and Vibration article:
% Brandt & Ahlin, Sampling and Time-Domain Analysis, May 2010.

% The example calculates upper and lower limits for the filter shapes
% according to the standards by IEC and ANSI (see the OCTLIMITS command).
% Three sampling frequencies are used, and the center frequency is kept
% constant at 500 Hz. The three sampling frequencies are chosen so that
% the first is too low to fulfill the filter shape limits, the second
% gives filter coefficients that fulfill the filter shapes, and the last
% sampling frequency is (way too) large, creating a filter that does not
% fulfill the shape.

% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

clear
close all

% Parameter values
fc=500;            % 1/1 octave center freq.
flo = fc/sqrt(2);   % low cutoff freq. (definition)
fhi = fc*sqrt(2);   % high cutoff freq. (definition)

% Calculate filter coefficients for three different fs
c={'b','g','k','m'};        % Line colors for plotting
fsl=[2 5 1000]*1000;        % sampling frequency list
warning off                 % This suppresses the check made by noctfilt
% which actually checks if the computed filter conforms with standards
% Now loop through the three sampling frequencies and for each calculate a
% third octave filter frequency response in H
for n = 1:3
    fs=fsl(n);
    N=16*1024;
    f=(0:fs/2/N:fs/2-1/N)';
    [b,a] = noctfilt(3,fc,fs);
    H=freqz(b,a,f,fs);
    semilogx(f,20*log10(abs(H)),c{n})
    if n == 1
        hold on
    end
end
warning on

% Calculate IEC/ANSI standard limits for the octave filter and plot
% those limits on top of the frequency responses (FRF) from above. Only the
% middle FRF in H will be within the bounds.
[Lu Ll fl]=noctlimits(3,fc);
semilogx(fl,Lu,'r',fl,Ll,'r')
legend('fs=2K','fs=5K','fs=1000K')
title('Filter shape limits of ANSI 1.11 and IEC 1260 in red')
axis([200 1000 -60 5])
grid
set(gca,'XTick',[200 500 1000])
```

```
function statchk(x, N, NPlot, FileName);
% STATCHK       Produce some standard statistics of data in (columns of) x
%
%           statchk(x, N, NPlot, FileName);
%
%           x           Time data in column(s)
%           N           Number of bins for PDF computation
%           NPlot       If NPlot=1 a plot with the PDF of x is plotted with
%                       the equivalent Gaussian distribution overlaid
%                       If NPlot=2, the same is plotted with logy scale
%                       If NPlot=0 no plot is produced.
%           FileName    If this string is given, the output of statchk is
%                       redirected to a log file FileName.log in the
%                       current directory (or in the directory indicated in
%                       the string FileName if a full path is given).
%                       Also, the actual statistical vectors are stored in
%                       mat file FileName.mat.
%
% This function performs some standard statistical analysis of time data in x.
% Also, a histogram is plotted with N bins centered around 0. If Nplot=1 a plot of
% the Gaussian distribution using mean and std of x is overlayed on the histogram.
% If N is not given a default of 30 bins are used for the histogram. After calling
% this function, a number of variables with generic names are logged in text file
FileName.
% If no FileName is given the values are listed on the screen.
%
% WARNING! This command will overwrite an existing log file with the name
% in FileName!
%-----------------------------------------------------------------------------------
---


% Copyright (c) 2009 by Anders Brandt
% Email: abra@ib.sdu.dk

if nargin == 2
   NPlot=0;
   fid=1;
elseif nargin == 1
   NPlot=0;
   N=30;
   fid=1;
elseif nargin == 3
       fid=1;
elseif nargin == 4
    fid=fopen(strcat(FileName,'.log'),'w');
elseif nargin > 4
   error('Wrong number of parameters!')
end

[mx,nx]=size(x);

m=mean(x);
Sigma2=diag(cov(x))';
Sigma=sqrt(Sigma2);
Skewness=askewness(x);
Kurtosis=akurtosis(x);
RMS=sqrt(1/mx*sum(x.^2));
Crest=max(abs(x))./RMS;
XMax=max(x);
XMin=min(x);
kol=[1:nx];
```

```
if fid ~= 1
    fprintf(fid,'Output of command STATCHK\n');
    fprintf(fid,'Date: %s\n',datestr(now));
end
fprintf(fid,'Statistical parameters:\n');
fprintf(fid,'===========================\n');
[S,E]=sprintf('%5s%12s%12s%12s%12s%12s%12s%12s%12s%12s\n','Col.','Max','Min','Mean',
'Crest','RMS','Std dev',...
'Variance','Skewness','Kurtosis');
fprintf(fid,'%s',S);
for col=1:nx,

S2=sprintf('%5d%12.2g%12.2g%12.2g%12.2g%12.2g%12.2g%12.2g%12.2g%12.2g',kol(col),XMax
(col),XMin(col),...
   m(col),Crest(col),RMS(col),Sigma(col),Sigma2(col),Skewness(col),Kurtosis(col));
   fprintf(fid,'%s\n',S2);
end

[row,col]=size(x);
if (col > 1) & NPlot
   fprintf(fid,' Stat: more than one column in vector, graphing only first
column!\n\n');
end
[H, XAx, G]=apdf(x(:,1),N,NPlot);
%
if nargin == 4
   S=['save ' FileName ' XMax XMin m Crest RMS Sigma Sigma2 Skewness Kurtosis'];
   eval(S)
   fclose(fid);
end
```

```
function y = timeint(x,fs,type,fc);
% TIMEINT Integrate time signal
%
%       y = timeint(x,fs,type,fc);
%
%       y       Time integration of x
%
%       x       Input time signal
%       fs      Sampling frequency
%       type    'Simple' (default), or 'HPFilter'
%       fc     Cutoff frequency for HP filter, 5 Hz default
%

% Copyright (c) 2009 by Anders Brandt
% Email: anders.brandt@abravibe.com

% Check parameters
if nargin == 2
    type='SIMPLE';
    fc=[];
elseif nargin == 3
    if strcmp(upper(type),'SIMPLE')
        fc=[];
    elseif strcmp(upper(type),'HPFILTER')
        fc=5;
    else
        fprintf('Wrong parameter ''type'', se help')
        return;
    end
elseif nargin == 4
    if strcmp(upper(type),'SIMPLE')
        fprintf('Wrong number of parameters for type ''SIMPLE''')
        return;
    end
end

% Process each case separately
if strcmp(upper(type),'SIMPLE')
    dx=1/fs;
    x=x-mean(x);
    y=dx*cumsum(x);
elseif strcmp(upper(type),'HPFILTER')
    % Create a 1. order HP filter with cutoff frequency fc
    fnyq=fs/2;               % Nyquist frequency
    frel=fc/fnyq;           % Relative cutoff frequency as wanted by butter
    Amp=1/(2*pi*fc);        % Butterworth has a unity numerator, HP filter does not
    [B,A]=butter(1,frel);    % 1. order HP filter coefficients
    y=Amp*filter(B,A,x);     % Integrate time signal
end
```