Example 1

```
%% Matlab seminar 2016
% Example 1: Discrete Time domain
% *****************************************************************
% Javier Gomez Bolanos 2016 - Dept. Signal Processing and Acoustics
%                   Aalto University
% *****************************************************************
% This is an example of Matlab script that generates a sinusoidal and
% process it to obtain different parameters from the signal.
% Each process is divided in Sections which are explained separately
% Section 1: Generate data
% Section 2: Calculate RMS value of the signal
clear all; clc; % clean everything :)

% ********************************************************************
%% Section 1: Generating signals
% generating a sine signal with frequency f0 and duration s_length sampled
% at fs frequency rate (generate it as column vector):
fs = 48e3; % sampling rate (Hz)
f0 = 1e3; % frequency of the signal (Hz)
s_length = 10; % duration in seconds

% hard way
inc_time = 1/fs; % time intervals between samples
N = ceil(s_length.*fs); % length of the vector in samples
time_axis = (0:N-1).'.*inc_time; % time axis vector
s0 = sin(2*pi.*time_axis.*f0); % generate data vector
% using Matlab functions
s1 = sine(f0,s_length,fs).';

% ********************************************************************
%% Section 2: Root Mean Square value of the signal
% hard way
s0_rms = sqrt(sum(abs(s0).^2,1)./N); %this is the root mean square equation
% using Matlab functions
s1_rms = rms(s1,1); % this is the Matlab function for the RMS of x(n)

% Confirm that the RMS value of periodic signals is independent of duration
s0_rms1 = sqrt(sum(abs(s0(1000:1000+fs-1)).^2,1)./fs);
s1_rms1 = rms(s1(1000:1000+fs-1),1);
```

```
disp(['Signal0 RMS: ' num2str(s0_rms)]);
disp(['Signal1 RMS: ' num2str(s1_rms)]);
disp(['Short Signal0 RMS: ' num2str(s0_rms1)]);
disp(['Short Signal1 RMS: ' num2str(s1_rms1)]);
```

Example 2

```
%% Matlab seminar 2016
% Example 2: Discrete Frequency domain
% ******************************************************************
% Javier Gomez Bolanos 2016 - Dept. Signal Processing and Acoustics
%                   Aalto University
% ******************************************************************
% This is an example of Matlab script that generates two sinusoidals and
% process it to obtain different parameters from the signal.
% Each process is divided in Sections which are explained separately
% Section 1: Generate data
% Section 2: FFT
% Section 3: Calculate RMS in frequency domain
clear all; clc;

% *********************************************************************
%% Section 1: Generate data
% generating two sine signal with frequency [f0 f1] and duration s_length
% sampled at fs frequency rate (generate it as column vectors):
fs = 48e3; % sampling rate (Hz);
f = [1e2 1e3]; % frequency of the signal 1 and 2 (Hz)
s_length = 1.5; % duration in seconds
inc_time = 1/fs; % time intervals between samples
N = ceil(s_length.*fs); % length of the vector in samples
time_axis = (0:N-1).'.*inc_time; % time axis vector (column vector)
s = zeros(N,size(f,2)); % Initialize variable
% in a FOR loop
for i=1:size(f,2)
    s(:,i) = sin(2*pi.*time_axis.*f(i)); % generate data matrix
end;
% in a matrix calculation
time_axis2 = repmat(time_axis,1,size(f,2)); % repeat the time_axis vector
f_2 = repmat(f,size(time_axis,1),1);% repeat the f vector
s = sin(2*pi.*time_axis2.*f_2);

% *********************************************************************
%% FFT
Nfft = 2^12; % size of the FFT
inc_freq = fs/Nfft; % frequency bin size
S0 = fft(s,Nfft); % FFT
```

```
S1 = fft(s,Nfft)/Nfft; % FFT and normalization
N1=Nfft/2; % short length of signal
S2 = fft(s(1:N1,:),Nfft)/N1; % zero-pading FFT and normalization
win = hanning(N1);
win = repmat(win./rms(win),1,size(s,2));
S3 = fft(s(1:N1,:).*win,Nfft)/N1; % zero-pading FFT and normalization

% ************************************************************************
%% Plotting
freq_axis = (0:Nfft-1).*inc_freq; % Frequency vector axis
figure(1);clf;
semilogx(freq_axis,20*log10(abs([sum(S0,2) sum(S1,2)...
    sum(S2,2) sum(S3,2)]))); % Plot in semilogarithmic scale
grid on; % Enable grid
xlim([20 48000-20]);ylim([-50 70]);
xlabel('Frequency (Hz)'); ylabel('Amplitude non corrected');
legend('Non FFT normalization','Normalized N=Nfft',...
    'Normalized zero-padding','Normalized windowing');
% plotting the peak amplitude
figure(2);clf;
semilogx(freq_axis,20*log10(abs([sum(S1,2) sum(S2,2) sum(S3,2)]).*2));
grid on; % Enable grid
xlim([20 20e3]);ylim([-30 0]);
xlabel('Frequency (Hz)'); ylabel('Amplitude');
legend('Normalized N=Nfft','Normalized zero-padding',...
    'Normalized windowing');
% plotting the rms value
figure(3);clf;
semilogx(freq_axis,20*log10(abs([sum(S1,2) sum(S2,2)...
    sum(S3,2)]).*2./sqrt(2)));
grid on; % Enable grid
xlim([20 20e3]);ylim([-30 0]);
xlabel('Frequency (Hz)'); ylabel('Normalized Magnitude RMS');
legend('Normalized N=Nfft','Normalized zero-padding',...
    'Normalized windowing');
```

Example 3

```
%% Matlab seminar 2016
% Example 3: Calibration and Absolute pressure from impulse responses
% ********************************************************************
% Javier Gomez Bolanos 2016 - Dept. Signal Processing and Acoustics
%                   Aalto University
% ******************************************************************** This
% is an example of Matlab script that generates a scaled sinusoidal to be
% used as calibration signal and loads the sweep response of a loudspeaker
% to obtain different parameters from the loudspeaker. Each process is
% divided in Sections which are explained separately Section 1:
% Generate/Load data Section 2: Calibration Section 3: Calculate parameters
clear all; clc;

%% Generate/Load data
% sine signal
fs = 48e3; % sampling rate (Hz)
f0 = 1e3; % frequency of the signal (Hz)
s_length = 10; % duration in seconds
s1 = 0.0043.*sine(f0,s_length,fs).'; % Assume this signal to be the signal
% of a calibrator (94dBSPL/1kHz) is measured by the microphone

% Loudspeaker response
load('LS_response_sweep.mat'); % x contains two vectors: the measured
% sweep and the inverse sweep spectra calculated for a sweep of 1Vrms
ls_sweep = x(:,1);
inv_sweep = x(:,2);

% ********************************************************************
%% Calibration to a sensitivity of 1 Vrms

% Calculate the rms of the signal
g_cal = rms(s1); % g_cal represents the sensitivity of the measurement
% chain (mic+preamp+DAQ)
% ******* Q1: why is g_cal so important?

% Calibrate the input signal
ls_sweep = ls_sweep./g_cal; % after this, if the input signal is a
% 1kHz sine signal with 94dBSPL the rms value of the signal is 1Vrms
% ********************************************************************
%% Calculate parameters

% Impulse response (exp-sweep method)
LS_sweep = fft(ls_sweep,size(inv_sweep,1));
FR = LS_sweep.*inv_sweep; % FFT to the same size of inv_sweep
ir = fftshift(ifft(FR)); % inverse Fourier transform
% ******** Q2: which 'symmetric' FFT?
ir_lin = ir(end/2+1:end); % linear part of IR
ir_har = ir(1:end/2); % Harmonics of IR (there is a way to separate each
% harmonic. You can read Farina's paper on sweeps to know how to implement
% the sweep and separate each component of the reponse

% Frequency response of loudspeaker
Nfft = 2^13; % size of the FFT
% ******** Q3: which size is the best?
inc_freq = fs/Nfft; % frequency bin size (increment of frequency)
FR_ls = fft(ir_lin,Nfft);

% Sensitivity calculated with broadband spectra
freq_axis = inc_freq.*(0:Nfft-1);
S1 = 20*log10(abs(sqrt(sum(abs(FR_ls).^2)/Nfft)));
% using the impulse response
S2 = sqrt(sum(abs(ir_lin).^2)); % S1 and S2 should be equal
% ******** Q4: Why S1 and S2 should be equal?


% Excess group delay
[~,ir_min] = rceps(ir_lin);
FR_min = fft(ir_min,Nfft);
FR_max = FR_ls./FR_min;
% excess group delay in seconds
GD_excess =[angle(FR_max(1)); -diff(unwrap(angle(FR_max)))]./(2*pi*inc_freq);
% ******** Q5: Why "angle(FR_max(1))" is added at the beginning of
% GD_excess?
%% Plotting

figure(1);clf;
 % Plot in semilogarithmic scale
semilogx(freq_axis,20*log10(abs(FR_ls)/2e-5),'Linewidth',2);
grid on; % Enable grid
```

```
xlim([20 20e3]);ylim([75 95]);
xlabel('Frequency (Hz)'); ylabel('Sound Pressure Level (dB)');
% ******** Q6: Why plotting with logarithmic frequency axes?
% ******** Q7: Why plotting with logarithmic SPL axes?
figure(2);clf;
time_axis = (0:size(ir_lin,1)-1)./fs; % time axis
plot(time_axis(1:2048).*1e3,ir_lin(1:2048),'Linewidth',2);
grid on; % Enable grid
xlabel('Time (ms)'); ylabel('Gain');

figure(3);clf;
semilogx(freq_axis,GD_excess.*1e3,'Linewidth',2); % GD in milliseconds
grid on; % Enable grid
xlim([20 20e3]);ylim([0 4]);
xlabel('Frequency (Hz)'); ylabel('Excess delay (ms)');
```

Example 4

```
%% Matlab seminar 2016
% Example 4: Schroeder integration
% *********************************************************************
% Javier Gomez Bolanos 2016 - Dept. Signal Processing and Acoustics
%                   Aalto University
% *********************************************************************
% This is a Matlab script that load the impulse response of a room and
% calculates the Schroeder integral to determine the reverberation time
clear all; clc;
% *********************************************************************
%% Load data
fs = 48000; % Sampling Rate
load('RoomImpulseResponse.mat'); % load data
% ******** Q1: Why not to assign a variable, eg. x = load('xxxx.mat');?
time_axis = (0:size(x,1)-1).'./fs; % time axis in seconds
% *********************************************************************
%% Calculate Schroeder curve
xx=x.^2; % square the response
% with FOR loop
sch1=xx; % initialize output vector
for i=size(x,1)-1:-1:1 % reverse integration
    sch1(i)=sch1(i+1)+xx(i);
end;
% with CUMSUM
sch2 = cumsum(xx,1,'reverse');

% *********************************************************************
%% Plotting

figure(1);clf;
plot(time_axis,20*log10(abs(x)),'b',time_axis,10*log10(sch1),'r',...
    time_axis,10*log10(sch2./max(sch2).*max(abs(x).^2)),'m','Linewidth',2);
xlabel('Time (s)'); ylabel('Level');
grid on;
legend('RIR','Schroeder 1', 'Schroeder 2 (Scaled)');
% You can use polyfit to make a linear regression of the Schroeder curve in
% order to obtain the slope of the curve for automatic calculation of RT.

% **** Q2: Why sch1 and sch2 are plotted with 10*log10 instead of 20*log10?
```

Example 5

```
function H = mag_smoothing(B)
%% Matlab seminar 2016
% Example 5: Smoothing algorithm
% ******************************************************************
% Javier Gomez Bolanos 2016 - Dept. Signal Processing and Acoustics
%                   Aalto University
% ******************************************************************
% This is a Matlab function that implements the 1/3 octave smoothing of the
% input magnitude response B.

stop=length(B)/2+1; % We don’t need the other side of spectra
smooth_factor=1/3; % 1/3 octave bandwidth
Q=sqrt(2^(smooth_factor))/(2^(smooth_factor)-1); % Q-factor
B1=abs(B); % Only magnitude
H=B1(1:stop); % Initialize output vector
q=fix(Q)+1; % pointer to first frequency to be smoothed
for i=q:stop % scan each frequency bin

    N=fix(i/Q); % ammount of frequency bins in the smoothing window
    if mod(N,2)==0 % detect if it is even
        N=N+1;
    end;
    fc=(N+1)/2; % center frequency
    fh=N-fc;

    if i+fh>stop % detect if there is not enough bandwidth
        f=N-(i+fh-stop);
    else
        f=N;
    end;

    H(i)=sqrt(sum(B1(i-fh:i+fh-(N-f)).^2)/f); % smoothing
end;
H = [H; H(end-1:-1:2)];
% ******** Q1: What kind of frequency window can be used?
% ******** Q2: Why negative spectra is not used in the calculations?
% ******** Q3: Why phase is not used in the smoothing?
```

Example 6

```
function [white, pink] = noise_generator(L,fs)
%% Matlab seminar 2016
% Example 6: Noise generator
% *******************************************************************
% Javier Gomez Bolanos 2016 - Dept. Signal Processing and Acoustics
%                   Aalto University
% *******************************************************************
% This is a Matlab function that implements a white and pink noise vectors
% Duration L in seconds and sampled rate fs in Hz
% *******************************************************************
%% Generate white noise vector
vector_length = L*fs; % desired length of our vector
white = randn(vector_length,1);% Gaussian white noise
% ******** Q1: Why randn() and not rand()?

% *******************************************************************
%% Generate pink noise
Nfft = vector_length; % Define FFT length
freq_axis = (0:Nfft-1)./Nfft.*fs; %Generate frequency axis
f = freq_axis(1:end/2+1); % Use only the positive frequency range
% ******** Q2: Why only positive frequencies?

White = fft(white,Nfft); % Go to frequency domain
% ******** Q3: Why we need the white noise here?

% Create a pink spectra envelope (-10dB per decade)
H = [0 sqrt(f(2) ./ f(2:end))];
H = [H.'; H(end-1:-1:2)']; % create negative spectra
H = H./rms(abs(H)); % normalize to unit power P=1
Pink = White.*H; % Multiply spectrums
pink = ifft(Pink,[],1,'symmetric'); % go to time domain
% ******** Q4: What is called the process described in the last two lines?
```