```
% MATLAB session for BRUFACE :11/10/2016 regarding: Systems,
Pole, Zero, Step, Impulse, lsim

clear       % clear all variables in the workspace
clc         % clear only the screen
close all   % close all the figures (if any)

% define a first order system
num1 = 100;          % numerator of the transfer function
den1 = [1 0.95 100]; % denominator of the transfer function

sys1 = tf(num1,den1);

% poles and zeros of the system
p1   = pole(sys1);
z1   = zero(sys1);

% ------------------------------------------------------------------------ %
% -------------------------- STEP RESPONSE  ---------------------------- %
% ------------------------------------------------------------------------ %

% step response of the system for 16 seconds using the step command of
% Matlab
figure()
step(sys1,16)
grid on
ylim([0 2])  % this command limits the y-axis from 0 to 2

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

% step response of the system for 16 seconds using the lsim command of
% Matlab

% first we define the step excitation u that will be applied to the system

t      = 0:0.01:16; % t is the time vector on which the step response will
                    % be computed with the lsim

u_step = ones(1,length(t));   % u is a vector full with ones in order to
                              % excite the system with a unit step

y_step = lsim(sys1,u_step,t); % evaluate the response of the system to the
                              % step excitation u

hold on
plot(t,y_step,'r--')
legend('The ''step'' command','The ''lsim'' command')
% Observe that the blue line coincides with the red one

% ------------------------------------------------------------------------ %
% ---------------------- RANDOM EXCITATION  ---------------------------- %
% ------------------------------------------------------------------------ %

% response of the system to a random Gaussian white noise input of mean 2
```

```
% and variance 0.1 for 30 seconds
tt     = 0:0.01:30;
u_rand = 2 + sqrt(0.1)*randn(length(tt),1);
y_rand = lsim(sys1,u_rand,tt);

figure()
subplot(1,2,1)
plot(tt,u_rand)
grid on
xlabel('time (sec)')
ylabel('Input excitation')
legend('u')
ylim([0 3.5])
subplot(1,2,2)
plot(tt,y_rand,'r')
grid on
xlabel('time (sec)')
ylabel('Output (response of the system)')
legend('y')
ylim([0 3.5])

% ------------------------------------------------------------------------ %
% ----------------------- IMPULSE EXCITATION  ---------------------------- %
% ------------------------------------------------------------------------ %

% impulse of a stable system

figure()
impulse(tf(10,[1 0.4 1]))
hold on

% impulse of an unstable system
impulse(tf(0.01,[1 0.04 -0.1]))
legend('Stable system','Unstable system')
ylim([-10 10])
xlim([0 60])
grid on
```

% MATLAB session for BRUFACE: 18/10/2016 regarding: Bode, Filter, Transients and Estimate of TF (Transfer Function)

```
clear       % clear all variables in the workspace
clc         % clear only the screen
close all   % close all the figures (if any)

% define a second order system
num1 = 100;          % numerator of the transfer function
den1 = [1 0.95 100]; % denominator of the transfer function

sys1 = tf(num1,den1);

% poles and zeros of the system
p1   = pole(sys1);
z1   = zero(sys1);

% ------------------------------------------------------------------------ %
% --------------------------- BODE PLOT  ----------------------------------- %
% ------------------------------------------------------------------------ %

figure()
bode(sys1)  % computes the frequency response of the system sys1
grid on
hold on

% define a first order system and compute the frequency response again
sys2 = tf(2,[1 10]);
bode(sys2,'r-')
legend('2nd order system','1st order system')

%% This part shows how a system can amplify, preserve or attenuate a signal

clear
clc
close all

% we define two sinusoidal signals with same amplitude and different
% frequencies

t     = 0:0.01:2000;
w1    = 0.01;
w2    = 10;

u1    = 10*sin(w1*t);
u2    = 10*sin(w2*t);

% we plot the two signals together with the summation of the two signals

figure()
plot(t,u1,'Linewidth',2)
legend('u1')
grid on
```

```
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
figure()
plot(t,u2,'r','Linewidth',2)
legend('u2')
grid on
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
figure()
plot(t,u1+u2,'Linewidth',2)
legend('u1 + u2')
grid on
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)

% define a second order system with damping ratio 0.1 and natural frequency
% 0.3
zeta  = 0.1;
wn    = 0.3;

g     = tf(wn^2,[1 2*zeta*wn wn^2]);

% compute the ratio of amplitudes and phase for the second order system
[mag,phase,wout] = bode(g);
mag   = squeeze(mag);  % observe the dimensions of "mag" and "phase"...
phase = squeeze(phase); % before and after the command "squeeze"

figure
subplot(2,1,1)
semilogx(wout,db(mag),'Color','k','Linewidth',2)
xlabel('Frequency (rad/s)')
ylabel('Magnitude(dB)')
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
grid on
hold on

subplot(2,1,2)
semilogx(wout,phase,'Color','k','Linewidth',2)
xlabel('Frequency (rad/s)')
ylabel('Phase(deg)')
grid on
hold on
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)

% excite the system with the two sinusoids as well as with the summation of
% the two signals

y1  = lsim(g,u1,t);
y2  = lsim(g,u2,t);

y12 = lsim(g,u1+u2,t);

figure()
```

```
plot(t,y1,'Linewidth',2)
legend('y1')
grid on
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
figure()
plot(t,y2,'Linewidth',2)
legend('y2')
grid on
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)
figure()
plot(t,y12,'Linewidth',2)
legend('y12')
grid on
set(gca,'FontSize',16)
set(findall(gcf,'type','text'),'FontSize',16)

%%

% ------------------------------------------------------------------------ %
% ---------------------------- FILTER  ----------------------------------- %
% ------------------------------------------------------------------------ %

% excite the system sys1 again with u_rand but now compute the response of
% the system with the filter command of Matlab... NOTE!!! FILTER WORKS
% WITH DISCRETE SYSTEMS SO WE FIRST HAVE TO DISCRETIZE THE SYSTEM!!!

% define a second order system
num1 = 100;          % numerator of the transfer function
den1 = [1 0.95 100]; % denominator of the transfer function

sys1 = tf(num1,den1);

% discretize the system with a sampling frequency of 0.01
sys1_d   = c2d(sys1,0.01);

% extract the numerator and the denominator of the system and convert it to
% format of type double
[b,a]    = tfdata(sys1_d,'v');

% response of the system to a random Gaussian white noise input of mean 2
% and variance 0.1 for 30 seconds
tt       = 0:0.01:30;
u_rand   = 2 + sqrt(0.1)*randn(length(tt),1);
y_rand   = lsim(sys1,u_rand,tt);

% excite the discrete system with the random input excitation
y_rand_f = filter(b,a,u_rand);

% create a time vector that will be used to plot the response of the
% discrete system. Observe the diference between tt used in 'lsim' and ttt
% used in 'filter'
ttt      = linspace(0,30,length(y_rand_f));
```

```
figure()
subplot(1,2,1)
plot(tt,u_rand)
grid on
xlabel('time (sec)')
ylabel('Input excitation')
legend('u')
ylim([0 3.5])
subplot(1,2,2)
plot(tt,y_rand,'r')
ylim([0 3.5])
hold on
plot(ttt,y_rand_f,'k--')
grid on
xlabel('time (sec)')
ylabel('Output (response of the system)')
legend(' y with ''lsim'' ',' y with ''filter'' ')

% ------------------------------------------------------------------------ %
% ------------------------- Transients  ---------------------------------- %
% ------------------------------------------------------------------------ %

% at this part of the code we will illustrate the transient phenomenon when
% you excite a system. We will do this by exciting a discrete time system
% with a sinusoidal input. In the beginning we expect to see a strange
% behavior and after some time the output will become also sinusoidal, as
% suggested by the linear system theory.

% create a time vector in order to excite the system for 10 seconds
t      = 0:0.01:10;

% create a sinusoidal input signal of amplitude 2 and frequency 1.5
% rad/sample
u_sin  = 2*sin(1.5*t);

% excite the system with the sinusoidal input
y_sin  = filter(b,a,u_sin);

figure()
subplot(1,2,1)
plot(t,u_sin)
legend('Input')
grid on
subplot(1,2,2)
plot(t,y_sin,'r')
legend('Output')
grid on

% Observe that the strange behavior is gone after approximately 5 seconds

%%
% ------------------------------------------------------------------------ %
% ------------------------- TFESTIMATE  ---------------------------------- %
% ------------------------------------------------------------------------ %
```

```
% create new random input of mean 0 and variance 0.1
N      = 10440;
u_rand = sqrt(0.1)*randn(N,1);

% create first a sufficiently long input excitation (here we excite the
% system with the same input 5 times in a row (5 periods)
u      = repmat(u_rand,5,1);

% excite the system and compute the response
y      = filter(b,a,u);

% remove the transient effects
u_est  = u(end/2 + 1:end,1);
y_est  = y(end/2 + 1:end,1);

% plot the estimate of the Frequency Response Function (FRF)
figure()
subplot(1,3,1)
tfestimate(u_est,y_est)
grid on

% compare with the estimate of the FRF when the cpsd command in Matlab is
% used
Suy    = cpsd(u_est,y_est);
Suu    = cpsd(u_est,u_est);

subplot(1,3,2)
plot(db(Suy./Suu))
grid on
legend('cpsd command')

% compare with the estimate of the FRF when the freqz command in Matlab is
% used. This is another way to compute the FRF of a discrete time system
[H,W] = freqz(b,a,size(Suy,1));

subplot(1,3,3)
plot(W,db(H))
grid on
legend('freqz command')
```

```
% MATLAB session for BRUFACE: 19/10/2016
clear
clc
close all

% We define the system
g = tf(10,[1 0.1 10]);

% ------------------------------------------------------------------------ %
% ------------------------------ PART 1 ---------------------------------- %
% ------------------------------------------------------------------------ %

% 1) The system is continuous (s-domain)
% ------------------------------------------------------------------------ %

% 2) The order of the denominator is 2 --> 2nd order system (two poles)
% ------------------------------------------------------------------------ %

% 3) The system is stable because the real part of the poles is negative

p = pole(g);
z = zero(g);

% 4)the gain of the system is 1 (Remember the trick of computing g(0))

% 5) ------------------------------------------------------------------------ %

% step response of the system for 100 seconds
figure()
step(g,100)
grid on

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

% step response of the system for 100 seconds using the lsim command of
% Matlab

% first we define the step excitation u that will be applied to the system

t      = 0:0.005:100; % t is the time vector on which the step response will
                      % be computed with the lsim

u_step = ones(1,length(t));   % u is a vector full with ones in order to
                              % excite the system with a unit step

y_step = lsim(g,u_step,t);    % evaluate the response of the system to the
                              % step excitation u

hold on
plot(t,y_step,'r--')
legend('The ''step'' command','The ''lsim'' command')

% Observe that the blue line coincides with the red one
```

```
% ------------------------------------------------------------------------ %
% 6) Yes we observe oscillations -- > the poles have a non-zero imaginary
% part

% ------------------------------------------------------------------------ %
% 7) ------------------------------------------------------------------- %

figure()
impulse(g)
xlim([0 100])
grid on

% ------------------------------------------------------------------------ %
% 8) Frequency response of the continuous system g with the bode command of
% Matlab

figure()
bode(g)
grid on

% ------------------------------------------------------------------------ %
% 9) ------------------------------------------------------------------- %
% Understanding the bode plot

% create a time vector in order to excite the system for 400 seconds
t     = 0:0.005:400;

% create a sinusoidal input signal of amplitude 4 and frequency 2.04
% rad/sec
u_sin = 4*sin(2.04*t);

y_sin = lsim(g,u_sin,t);

figure
subplot(1,2,1)
plot(t,u_sin,'b')
xlabel('time (sec)')
ylabel('Input excitation u')
legend('u')
grid on
ylim([-5 5])
subplot(1,2,2)
plot(t,y_sin,'r--')
xlabel('time (sec)')
ylabel('Output response y')
legend('y')
grid on

% ------------------------------------------------------------------------ %
% ----------------------------- PART 2 ----------------------------------- %
% ------------------------------------------------------------------------ %

% define the sampling period
Ts    = 0.05;
```

```
% discretize the system with a sampling period of 0.05 seconds
g_d   = c2d(g,Ts);

% extract the numerator and the denominator of the system and convert it to
% format of type double
[b,a] = tfdata(g_d,'v');

% ------------------------------------------------------------------------ %
% 1) ---------------------------------------------------------------------- %
p_d = pole(g_d);
z_d = zero(g_d);

% ------------------------------------------------------------------------ %
% 2) ---------------------------------------------------------------------- %

% First way -- > the poles of the discrete system should be inside the unit
% circle
pzmap(g_d) % or p_d = pole(g_d) and then observe that norm(p_d(1,1)) < 1

% Second way -- > impulse response should not explode

figure()
impulse(g_d)
grid on

% Third way -- > step response should also not explode

figure()
step(g_d)
grid on

% ------------------------------------------------------------------------ %
% 3) ---------------------------------------------------------------------- %

% create a time vector in order to excite the system for 300 seconds
t       = 0:Ts:300;

% create a sinusoidal input signal of amplitude 2 and frequency 3
% rad/sample
u_sin_d = 2*sin(3*t);

% excite the system with the sinusoidal input
y_sin_d = filter(b,a,u_sin_d);

figure()
subplot(1,2,1)
plot(t,u_sin_d)
set(gca,'FontSize',12)
xlabel('time (sec)')
ylabel('Input excitation u')
legend('u')
grid on
subplot(1,2,2)
```

```
plot(t,y_sin_d,'r')
set(gca,'FontSize',12)
xlabel('time (sec)')
ylabel('Output response y')
legend('y')
grid on

% ------------------------------------------------------------------------ %
% 4) Frequency response of the dicrete time system g_d with the bode
% command of Matlab

figure()
bode(g_d)
grid on
```

```
% MATLAB session for BRUFACE: 26/10/2016 regarding DFT

%% This code is focused on the lecture of 26_10_2016 and deals with the
% implementation of the Discrete Fourier Transform (DFT) in MATLAB

clear
clc
close all

% we construct a sinusoid and we compute the DFT of the signal

% frequency of the continuous signal
f1      = 1;

% sampling frequency
fs      = 10;

% sampling period Ts
Ts      = 1/fs;

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

% number of data for the construction of the continuous signal
Nc      = 1000;

% defining the time vector to plot the continuous signal
tstart = 0;
tend   = 1;

tc      = linspace(tstart,tend,Nc)';

% continuous sine (in the sense that a large number of samples (Nc) in time
% domain has been chosen
uc      = sin(2*pi*f1*tc);

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

% defining the time vector at which the signal is sampled
td      = (tstart:Ts:tend-Ts)'; %(or td     = (tstart:1/fs:tend-1/fs)';

% number of samples for the discrete signal
N       = length(td);

% the discrete time signal
ud      = sin(2*pi*f1*td);

% computing the DFT of the signal with the command fft. Observe that we
% normalize with the number of samples because the signal is periodic
Ud      = 1/N*fft(ud);

% this is the frequency frid we can access with the DFT once we know fs and
```

```
% N
f       = linspace(0,fs-fs/N,N);

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

figure()

subplot(1,2,1)
plot(tc,uc,'Linewidth',3)
hold on
plot(td,ud,'ro','Linewidth',3,'Markersize',20)
xlabel('time (sec)')
ylabel('Sine')
grid on
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)

subplot(1,2,2)
stem(f,abs(Ud),'r','Linewidth',3,'Markersize',20)
xlabel('Frequency (Hz)')
ylabel('Amplitude')
grid on
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)

%% we construct a cosine and we compute the DFT of the signal

clear
clc
close all

% frequency of the continuous signal
f1      = 1;

% sampling frequency
fs      = 10;

% sampling period Ts
Ts      = 1/fs;

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

% number of data for the construction of the continuous signal
Nc      = 1000;

% defining the time vector to plot the continuous signal
tstart = 0;
tend   = 1;

tc      = linspace(tstart,tend,Nc)';

% continuous cosine (in the sense that a large number of samples (Nc) in
```

```
% time domain has been chosen
uc      = cos(2*pi*f1*tc);

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

% defining the time vector at which the signal is sampled
td      = (tstart:Ts:tend-Ts)'; %(or td     = (tstart:1/fs:tend-1/fs)';

% number of samples for the discrete signal
N       = length(td);

% the discrete time signal
ud      = cos(2*pi*f1*td);

% computing the DFT of the signal with the command fft. Observe that we
% normalize with the number of samples because the signal is periodic
Ud      = 1/N*fft(ud);

% this is the frequency frid we can access with the DFT once we know fs and
% N
f       = linspace(0,fs-fs/N,N);

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

figure()

subplot(1,2,1)
plot(tc,uc,'Linewidth',3)
hold on
plot(td,ud,'ro','Linewidth',3,'Markersize',20)
xlabel('time (sec)')
ylabel('Cosine')
grid on
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)

subplot(1,2,2)
stem(f,abs(Ud),'r','Linewidth',3,'Markersize',20)
xlabel('Frequency (Hz)')
ylabel('Amplitude')
grid on
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)

%% here we show different ways to plot the spectrum of a signal
% (DFT line numbers, amplitude, dB, frequency)

clear
clc
close all

% frequency of continuous signal
```

```
fc  = 62.5;

% sampling frequency
fs  = 1000;

% number of samples
N   = 16;

% phase of continuous signal
phi = pi/2;

% amplitude of continuous signal
A   = 1;

% discrete time vector
td = (0:1/fs:(N-1)*1/fs)';

% discrete time signal (sampling the continuous signal at the discrete time
% instants contained in td)
ud = A*sin(2*pi*fc*td + phi);

% compute the DFT of the discrete time signal
U  = 1/size(ud,1)*fft(ud);

% frequency grid
f  = linspace(0,(N-1)*fs/N,N);

figure()
subplot(2,2,1)
plot(td,ud,'ro','Linewidth',4,'Markersize',22)
xlabel('time')
ylabel('Discrete signal')
grid on
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)
subplot(2,2,2)
plot(f,db(U),'rx','Linewidth',4,'Markersize',22);
xlabel('frequency (Hz)')
ylabel('Amplitude (dB)')
grid on
ylim([-40 0])
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)
subplot(2,2,3)
stem(1:size(U,1),abs(U),'r','Linewidth',4,'Markersize',22);
xlabel('DFT line number')
ylabel('Amplitude')
grid on
ylim([0 1])
xlim([0 17])
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)
subplot(2,2,4)
stem(f,abs(U),'r','Linewidth',4,'Markersize',22);
xlabel('frequency (Hz)')
```

```
ylabel('Amplitude')
grid on
ylim([0 1])
hold on
h = plot(fc,0.5,'rx','Markersize',22);
set(h,'Linewidth',4)
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)

%% here we show an alternative way to go from frequency to time domain by
% defining only half the signal spectrum

clear
clc
close all

% number of periods of the sinusoid measured
Nper             = 1;

% frequency of the continuous signal
fc               = 1000/16;

% 16 measurements in one period of the underlying signal
fs               = 16*fc;

% the fft of the sinusoid has the size of the window in the time domain
% (practically the number of measurements)
U1               = zeros(16,1);

% the frequency grid
f                = linspace(0,(size(U1,1)-1)*fs/(size(U1,1)),size(U1,1));

% 1 period is measured --> line number 1 will be excited
U1(Nper+1,1)     = 1/2*exp(-i*pi/2);
U1(end-Nper+1,1) = 1/2*exp(i*pi/2);

% the time domain signal is obtain with the inverse DFT
ud1              = size(U1,1)*ifft(U1);

fc               = f(1,2);
Tc               = 1/fc;
td               = 0:1/fs:Tc - 1/fs;

% alternative way to obtain the time domain signal by defining only half
% the signal spectrum

U2               = zeros(16,1);
U2(Nper+1,1)     = 1/2*exp(-i*pi/2);

ud2              = 2*real(size(U1,1)*ifft(U2));

figure()
plot(td,ud1,'o','Linewidth',4,'Markersize',30);
xlabel('time (sec)')
```

```
ylabel('discrete signal')
grid on
hold on
plot(td,ud2,'rx','Linewidth',4,'Markersize',30);
set(gca,'FontSize',30)
set(findall(gcf,'type','text'),'FontSize',30)
```

```
% MATLAB session for BRUFACE: 8/11/2016 regarding DFT

clear
clc
close all

% ------------------------------------------------------------------------ %
% ------------------------------ PART 1 ---------------------------------- %
% ------------------------------------------------------------------------ %

%% 1)
% The frequency resolution is given by fs / N where fs is the sampling
% frequency and N is the number of measurements. The frequency grid we can
% access is 0 : fs/N : fs-fs/N (N samples in time correspond to N
% frequencies in the grid).

% ------------------------------------------------------------------------ %
%% 2)
% Sampling period of 0.01 seconds / sample or 100 samples / second (Hz)
% (Sampling frequency)
% So in Tmeas = 1 minute we obtain
% N = fs * Tmeas = 100 samples/sec * 60 sec = 6000 samples and the
% frequency resolution is fs / N = 100/6000 = 1/60.

% ------------------------------------------------------------------------ %
%% 3)
% Resolution fs / N = 1 and fs = 1000 Hz (samples/sec)
% so N = fs * 1 = 1000 samples. The measurement time in order to obtain
% 1000 samples is Tmeas = N * Ts = N / fs = 1 second. We will not observe
% leakage in the amplitude spectrum because in 1 second we measure
% 1 sec / 0.25 (sec/period) = 4 periods so we measure an integer number of
% periods.

%%

% ------------------------------------------------------------------------ %
% ------------------------------ PART 2 ---------------------------------- %
% ------------------------------------------------------------------------ %

%% 1)
% In this exercise we study the ALIAS effect in case when the Nyquist
% criterion for the sampling frequency is not satisfied.

clear
clc
close all

% frequencies of the two signals
f1     = 2;
f2     = 8;
```

```
% sampling frequency
fs     = 10;

% phase for the first signal
ph1    = pi;

% ------------------------------------------------------------------------- %
% number of measurements in the time interval in order to create the
% continuous time signal (NOTE: this is NOT the number of samples of the
% discrete time signals. The number of samples will depend on the sampling
% frequency)
Nc     = 1000;

% starting and ending time of measurements
tstart = 0;
tend   = 1;

% building the continuous time signals
tc     = linspace(tstart,tend,Nc)';
uc1    = sin(2*pi*f1*tc + ph1);
uc2    = sin(2*pi*f2*tc);

% ------------------------------------------------------------------------- %
% time vector for the discete time signals
td     = (tstart:1/fs:tend-1/fs)';

% Now can extract the number of samples of the discrete signals
N      = length(td);

% the discrete time signals
ud1    = sin(2*pi*f1*td + ph1);
ud2    = sin(2*pi*f2*td);

% ------------------------------------------------------------------------- %
% Computing the DFT of the discrete time signals (expected to be equal
% since the discrete time versions are equal)
Ud1    = 1/size(ud1,1)*fft(ud1);
Ud2    = 1/size(ud2,1)*fft(ud2);

% the frequency grid we can access with the DFT
f      = linspace(0,(size(ud1,1)-1)*fs/(size(ud1,1)),size(ud1,1));

% ------------------------------------------------------------------------- %
figure()
subplot(2,1,1)
plot(tc,uc1,'Linewidth',4)
hold on
plot(td,ud1,'ko','Linewidth',4,'Markersize',15)
xlabel('time')
ylabel('Slow Sinus')
grid on
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)
```

```
subplot(2,1,2)
plot(tc,uc2,'Linewidth',4)
hold on
plot(td,ud2,'rx','Linewidth',4,'Markersize',15)
xlabel('time')
ylabel('Fast Sinus')
grid on
set(gca,'FontSize',20)
set(findall(gcf,'type','text'),'FontSize',20)

figure
plot(td,ud1,'ko','Linewidth',4,'Markersize',15)
hold on
plot(td,ud2,'rx','Linewidth',4,'Markersize',15)
xlabel('time')
ylabel('Discretized Sinusoids')
grid on
set(gca,'FontSize',20)
set(findall(gcf,'type','text'),'FontSize',20)

figure
stem(f,abs(Ud1),'Linewidth',4,'Markersize',15)
xlabel('Frequency (Hz)')
ylabel('Amplitude')
grid on
xlim([0 11])
set(gca,'FontSize',20)
set(findall(gcf,'type','text'),'FontSize',20)

% ------------------------------------------------------------------------ %
%% 2)
% In this exercise we study the leakage effect

clear
clc
close all

fc    = 2;
fs    = 10;
Ndata = [13 15]';
tc    = linspace(0,2,1000);
uc    = sin(2*pi*fc*tc);

% ------------------------------------------------------------------------ %
figure()

for k = 1:length(Ndata)

    N  = Ndata(k,1);

    td = (0:1/fs:(N-1)*1/fs)';
    %alternative td = linspace(0,(N-1)*1/fs,N)

    ud = sin(2*pi*fc*td);
    U  = 1/size(ud,1)*fft(ud);
```

```
    f  = linspace(0,(size(ud,1)-1)*fs/(size(ud,1)),size(ud,1));

    subplot(length(Ndata),2,2*k-1)
    plot(tc,uc,'Linewidth',4)
    hold on
    plot(td,ud,'ro','Linewidth',4,'Markersize',15)
    xlabel('time')
    ylabel('sinusoid')
    set(gca,'FontSize',15)
    set(findall(gcf,'type','text'),'FontSize',15)
    grid on

    subplot(length(Ndata),2,2*k)
    stem(f,abs(U),'Linewidth',4,'Markersize',15);
    xlabel('frequency (Hz)')
    ylabel('Amplitude')
    grid on
    ylim([0 1])
    hold on
    h = plot(fc,0.5,'rx','Linewidth',4,'Markersize',15);
    set(gca,'FontSize',15)
    set(findall(gcf,'type','text'),'FontSize',15)

end

% ------------------------------------------------------------------------- %
%% 3)

clear
clc
close all

% N = 16 samples in one period means 16*Ts = Tc or 16/fs = 1/fc or
% fc = fs/16 where Tc is the period of the continuous time signal and fc is
% the frequency of the continuous time signal
fc  = 1000/16;

% sampling frequency
fs  = 1000;

% number of measurements for the first signal (one period)
N1  = 16;

% number of measurements for the second signal (two periods)
N2  = 32;

% amplitude for both signals
A   = 1;

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

% constructing the first signal and its DFT
td1 = (0:1/fs:(N1-1)*1/fs)';
ud1 = A*sin(2*pi*fc*td1);
U1  = 1/N1*fft(ud1);
```

```
f1  = linspace(0,(N1-1)*fs/N1,N1);

% constructing the second signal and its DFT
td2 = (0:1/fs:(N2-1)*1/fs)';
ud2 = A*sin(2*pi*fc*td2);
U2  = 1/N2*fft(ud2);
f2  = linspace(0,(N2-1)*fs/N2,N2);

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

figure()
subplot(2,3,1)
plot(td1,ud1,'o','Linewidth',4,'Markersize',15)
xlabel('time')
ylabel('sinusoid')
xlim([0 0.04])
grid on
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

subplot(2,3,2)
stem(0:size(U1,1)-1,abs(U1),'Linewidth',4,'Markersize',15);
xlabel('DFT line number')
ylabel('Amplitude')
grid on
ylim([0 1])
xlim([0 16])
hold on
h = plot(1,0.5,'rx','Markersize',15);
set(h,'Linewidth',4)
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

subplot(2,3,3)
stem(f1,abs(U1),'Linewidth',4,'Markersize',15);
xlabel('frequency (Hz)')
ylabel('Amplitude')
grid on
ylim([0 1])
hold on
h = plot(fc,0.5,'rx','Markersize',15);
set(h,'Linewidth',4)
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

subplot(2,3,4)
plot(td2,ud2,'o','Linewidth',4,'Markersize',15)
xlabel('time')
ylabel('sinusoid')
xlim([0 0.04])
grid on
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

subplot(2,3,5)
stem(0:size(U2,1)-1,abs(U2),'Linewidth',4,'Markersize',15);
```

```
xlabel('DFT line number')
ylabel('Amplitude')
grid on
ylim([0 1])
xlim([0 32])
hold on
h = plot(2,0.5,'rx','Markersize',15);
set(h,'Linewidth',4)
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

subplot(2,3,6)
stem(f2,abs(U2),'Linewidth',4,'Markersize',15);
xlabel('frequency (Hz)')
ylabel('Amplitude')
grid on
ylim([0 1])
hold on
h = plot(fc,0.5,'rx','Markersize',15);
set(h,'Linewidth',4)
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

%%

% ------------------------------------------------------------------------ %
% ------------------------------ PART 3 ---------------------------------- %
% ------------------------------------------------------------------------ %

%% 1)

clear
clc
close all

% number of periods measured
Nper             = 1;

% the frequency of the underlying continuous signal
fc               = 62.5;

% sampling frequency
fs               = 1000;

% Number of samples in one period = Tc / Ts = fs / fc
N                = fs/fc;

% the fft of the sinusoid has the size of the window in the time domain
% (practically the number of measurements)
U1               = zeros(N*Nper,1);

% the frequency grid
f                = linspace(0,(size(U1,1)-1)*fs/(size(U1,1)),size(U1,1));

% 1 period is measured --> continuous frequency will be at line number 1
```

```
% 2 periods are measured --> continuous frequency will be at line number 2
% and so on so forth...
U1(Nper+1,1)     = 1/2*exp(-i*pi/2); % this is the scaled to 1 fft coeff
U1(end-Nper+1,1) = 1/2*exp(i*pi/2);

% the measured signal
ud1              = size(U1,1)*ifft(U1);

% frequency of the underlying signal is the frequency in f corresponding to
% the line number
fc               = f(1,Nper+1);
Tc               = 1/fc;
td               = 0:1/fs:Tc*Nper - 1/fs;

U2               = zeros(N*Nper,1);
U2(Nper+1,1)     = 1/2*exp(-i*pi/2);

ud2              = 2*real(size(U1,1)*ifft(U2));

figure()
plot(td,ud1,'o','Linewidth',4,'Markersize',15);
xlabel('time')
ylabel('measured signal')
grid on
hold on
plot(td,ud2,'rx','Linewidth',4,'Markersize',15);
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

% ------------------------------------------------------------------------ %
%% 2)

clear
clc
close all

% sampling frequency
fs   = 1000;

% amplitude is constant for all frequencies in the multisine
Ar   = 1;

% number of samples in one period of the multisine (the period of the
% slowest term in the multisine)
N    = 1000;

% the frequency of the multisine (the frequency of the slowest term in the
% multisine). Remember that we want 1000 samples in one period of the
% multisine. So we can define the slowest frequency in the multisine,
% namely the frequency of the first term (the slowest term). Moreover, in
% the discrete signal we want one period of the slowest term therefore we
% fix the frequency of the slowest term equal to the first (not zero-th)
% DFT line which corresponds to the resolution. The rest of the terms will
% have frequencies which will be integer number of this fundamental
% frequency, as suggested by the formula of the multisine.
```

```
fo   = fs/N;

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

% Frequency band 0 - 0.1*fs. So the highest frequency in the multisine
% which is equal to F*k (see the formula of the multisine) should be as
% close as possible to the largest frequency we want to excite
F    = 0.1*fs / fo; % in case this number is not an integer, use the
                    % commands ceil and floor to go to the closest integer

% ------------------------------------------------------------------------ %
% Now we will fill in the spectrum of the multisine
% keep in mind --> line number 2 will be excited because two periods of
% frequency 2*fo are measured and the same holds for all the frequencies
% till 100*fo (0.1*fs). If we would measure two periods of the multisine
% then there would be 4 periods of the term with frequency 2*fo and the
% same component would move from line number 2 to line number 4!

U    = zeros(N,1);

for k = 1:F

    U(k+1,1) = 1/2*exp(-i*2*pi*rand(1,1));

end

ud   = 2*real(size(U,1)*ifft(U));
ud   = ud/std(ud); % here we scale the discrete time signal to have an
                   % rms equal to 1

td   = 0:1/fs:(1/fo)-1/fs;
f    = 0:fs/N:fs - fs/N; % or f=linspace(0,fs - fs/N,N)

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

figure()
subplot(1,2,1)
plot(td,ud,'k','Linewidth',3);
grid on
xlabel('time')
ylabel('Multisine')
ylim([-5 15])
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

subplot(1,2,2)
plot(f,abs(U),'o','Linewidth',1.5,'Markersize',8);
xlabel('frequency (Hz)')
ylabel('Amplitude')
grid on
ylim([0 0.7])
xlim([0 2*F*fo])
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)
```

```
% ------------------------------------------------------------------------ %

% now we repeat the same signal in time and go back to frequency
ud_per = [ud;ud];
Ud_per = 1/length(ud_per)*fft(ud_per);
f_per  = 0:fs/length(ud_per):fs - fs/length(ud_per);

figure
plot(f_per,abs(Ud_per),'o','Linewidth',1.5,'Markersize',8);
xlabel('frequency (Hz)')
ylabel('Amplitude')
grid on
xlim([0 2*F*fo])
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

% We observe that there are now gaps between the excited DFT lines. The
% reason is that now we have measured more than one periods of the whole
% signal which corresponds to two periods of the slowest term in the
% multisine, namely the one with frequence fo. As a consequence, if we
% measure two periods of the whole signal then in the whole signal there
% will be two periods of the first term in the formula of the multisine
% (the term in the multisine with frequency 1*fo moved from DFT line 1 to
% DFT line 2), 4 periods of the second term of the multisine
% (the term in the multisine with frequency 2*fo moved from DFT line 2 to
% DFT line 4), 6 periods of the third term of the multisine
%(the term in the multisine with frequency 1*fo moved from DFT line 3 to
% DFT line 6), and so on so forth. In this way, there are now certain DFT
% lines which are not excited and appear in the amplitude of the DFT as
% gaps between the excited lines.
```

```
% Matlab exercise session for Bruface 10/11/2016 : SIMULINK
clear
clc
close all

% continuous transfer function
g       = tf([1 1],[1 0.9 1]);
[bc,ac] = tfdata(g,'v');

pc      = pole(g);
zc      = zero(g);

% sampling period in sec
Ts      = 1/1000;

% discretize the system g with sampling period Ts
gd      = c2d(g,Ts);
[bd,ad] = tfdata(gd,'v');

%%

% simulation time in seconds
T_sim   = 15;

% random input of N samples
N       = floor(T_sim / Ts);
u_rand  = randn(N,1);
t_rand  = (0:Ts:(N-1)*Ts)';

% simulate the Simulink model
sim('sim_model_2');

% excite the discrete time system also in Matlab and compare with the
% response from simulink
yd_matlab = filter(bd,ad,u_rand);

figure
subplot(1,2,1)
plot(t_rand,yd_matlab,'ko')
hold on
plot(t_rand,y_sim,'rx')
xlabel('time')
ylabel('Output')
legend('Matlab','Simulink')
grid on
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

subplot(1,2,2)
plot(t_rand,db(yd_matlab - y_sim),'mx')
grid on
xlabel('time')
```

```
ylabel('Error (dB)')
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)

%%

K = 1000;
% simulate the Simulink model
sim('sim_model_3');

figure
subplot(1,2,1)
plot(t_rand(1:end-1),u_rand(1:end-1))
hold on
plot(t_rand(1:end-1),y_sim(2:end),'r--')
grid on
legend('Reference','Output')
xlabel('time')
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)
subplot(1,2,2)
plot(t_rand(1:end-1),u_rand(1:end-1) - y_sim(2:end))
grid on
legend('Error')
set(gca,'FontSize',15)
set(findall(gcf,'type','text'),'FontSize',15)
```

% To ask for the sim_model_1.slx, sim_model_2.slx, sim_model_3.slx contact : Maral.Zyari@vub.ac.be or Georgios.Birpoutsoukis@vub.ac.be