TLT-5200/5206 Communication Theory, Matlab Exercise #1

MATLAB (www.mathworks.com)

Interactive commercial program (Windows, Linux, Unix)

Room TC221 has currently Matlab version 2009a (7.8.0.)

Has many functions for different applications, organized as toolboxes. (DSP, Communication, Neural network, Statistics ….).

Easy to program with m-code (somewhat similar to C and Fortran)

Started by the command “matlab” (unix) or double clicking the Matlab icon (windows)

When started you should have visible some of the following: Command window, Current Directory, Workspace, Command History or Launch Pad, Depending on View settings.
(View-> Desktop Layout-> Default, is good basic setup)

(Help -> Using the Desktop)

Very good Help function for all commands:
```
>> help function_name
```

Also, further help can be got with:
```
>> doc function_name
```

Environment:

All Variables are m * n -matrixes (a special case is a scalar)

<u>Constants:</u> pi, complex numbers (imaginary unit i or j), realmax, realmin, inf, eps etc.

<u>Functions:</u> sin, cos, exp, log10, exp, abs etc.

Matrices:

Try out commands (preceded with >>) in your Matlab command window (you can skip this part if you already feel comfortable using Matlab.)

Creating scalar
```
>> s=2
```

Creating matrix
```
>> a=[1 3;6 9]
```

Creating vector
```
>> v=[1 5 9]
```

Summation
```
>> a+5
```

Multiplication
```
>> b=s*v
```

Element-by-element multiplication
```
>> v.*b
```

Checking length of a vector
```
>> length(v)
```

Checking size of a matrix
```
>> size(a)
```

Accessing certain element of a matrix
```
>> a(1,2)
```

Accessing certain elements of a variable
```
>> v(1:2)
```

Creating vector with elements from 0 to 0.5 with 0.1 step-size (begin:stepsize:end)
```
>> t=0:0.1:0.5
```
OR
```
>> t=linspace(0,0.5,6)
```

M-file

Ascii-file with the extension .m, includes Matlab commands

To create M-file:

- File->New->M-file (windows)

OR

- ‘edit mfilename’, where mfilename is the desired file name for the m-file. The actual file created with the command is mfilename.m

The file is created to “current directory” displayed above the command window

Run the M-file in command line by using the file name without the .m

M-files can also act as functions. More information for this with command:

```
>> help function
```

Notice! DO NOT use file names starting with number, or filenames having any special characters in it. Matlab might recognize them as commands, not as m-files, when you try to run them

Simulation:

Sampling:

The modeled signals and systems in this Course are mostly analog (continuous-time). The simulation environment is digital (discrete-time).

In Matlab, we have certain constraints from the DSP world, such as

- A continuous time signal $x_c(t)$ is represented by a vector, $[x_c(t_0)\ x_c(t_1)\ ...\ x_c(t_{N-1})]$, whose length is N, i.e. the signal is truncated and sampled.

- Time interval t_{N-1}-t_0 and sampling interval T_s=t_{i+1}-t_i depend on the signal.

- Time interval is chosen large enough to obtain enough information. Sampling frequency typically chosen much higher than the Nyquist frequency, 2*f_max.

- Sampling frequency F_s=1/T_s is chosen so large that the sampled signal "looks like" analog signal in Matlab. The rule of thumb is 10 times the maximum frequency component.

<u>Example. Creating 50 Hz sinusoidal signal</u>

```
>> Fs=500;                  % sampling frequency = 10*50Hz
>> Ts=1/Fs;                 % sampling interval
>> t=0:Ts:0.1;              % sampling time instants [s]
>> x=sin(2*pi*50*t);        % signal vector
>> plot(t,x)                % plot in time domain
```

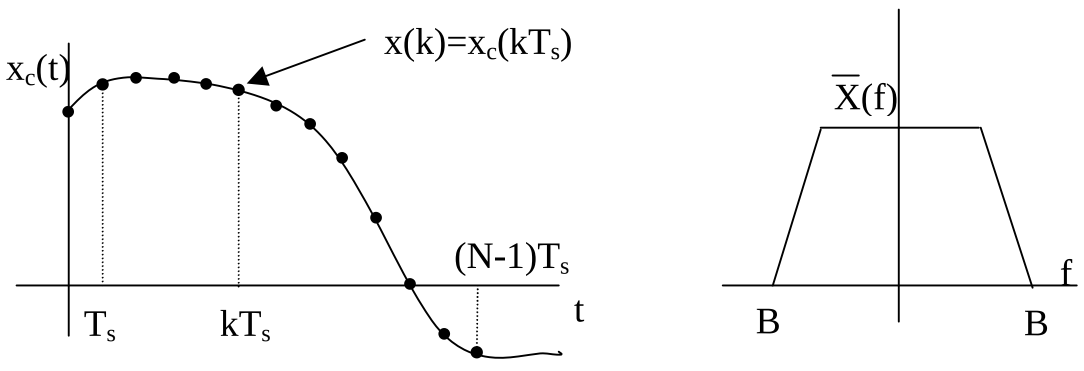
Fourier Transform (FT):

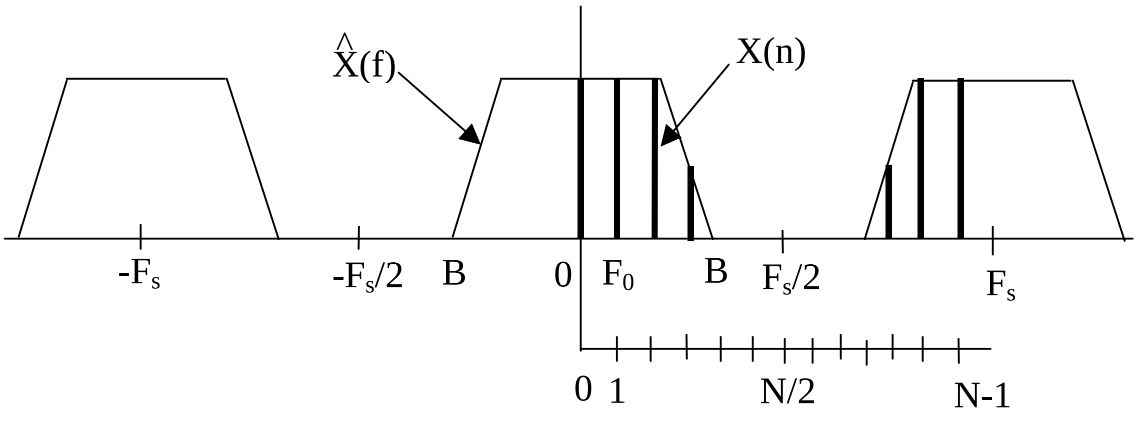
$$\overline{X}(j\omega) = \int_{-\infty}^{\infty} x_c(t)e^{-j\omega t}dt$$ Continuous time FT, CTFT

$$\hat{X}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} x(k)e^{-j\omega k}$$ Discrete time FT, DTFT

$$X(n) = \sum_{k=0}^{N-1} x(k)e^{-j2\pi kn/N}, \quad n = 0,1,\ldots,N-1$$ Discrete FT, DFT

x(k) is the sampled signal from the $x_c(t)$



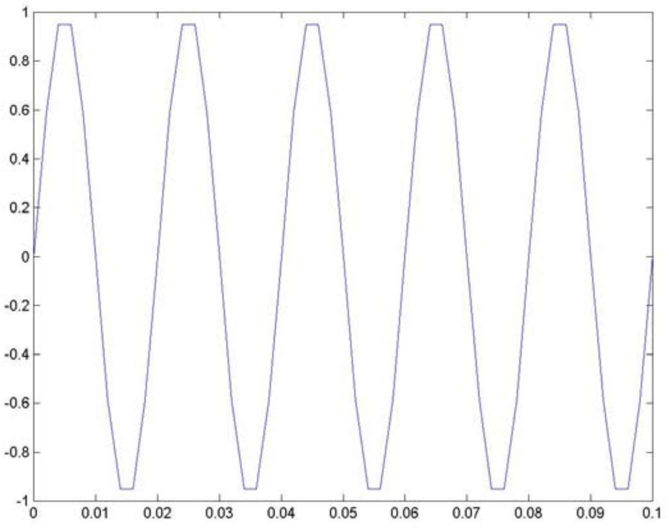


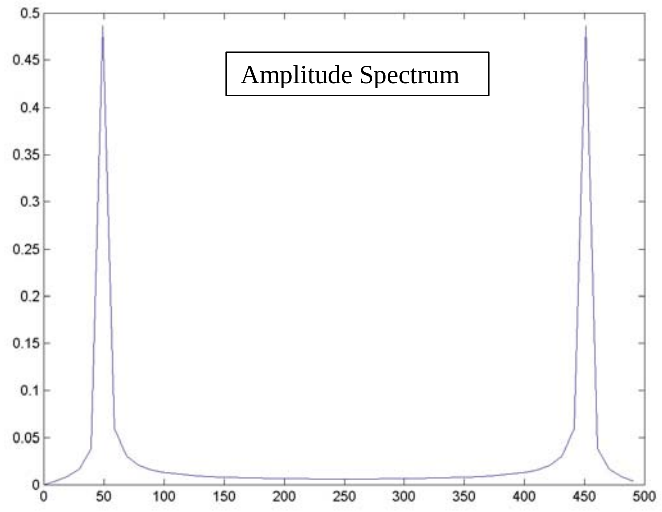
In Matlab, DFT can be calculated with the command FFT.

FFT calculation is not related to sampling frequency Fs, it is from 0 … ((N-1)/N)*2pi where pi refers to half the sample rate. For plotting you have to generate the vector of frequency points. Frequency resolution Fo depends on sampling frequency and the length of the signal vector. With real signals all the information is between 0 – Fs/2.

Example. Plotting DFT of 50 Hz sinusoidal signal

```
>> Fx=fft(x); % DFT of X, saved to Fx
>> Nx=length(x);
>> Fo=1/(Ts*Nx); % frequency resolution.
>> freq=0:Fo:(Nx-1)*Fo;  % Frequency Axis
>> plot(freq,abs(Fx)/Nx)  % Amplitude Spectrum
```





Matlab-Exercise 1.

1. Signal generation

Generate 35 Hz sinusoidal-signal with sampling frequency Fs=650.

- Determine Ts
- Enter the length N= …
- Generate the signal x=sin(..)
- Plot in time domain plot(t,x)
- Compute the length N and the frequency resolution Fo of the signal
- Compute DFT F=fft(x);
- Generate Frequency axis f=…
- Plot the amplitude spectrum plot(f,…

2. Adding two signals

An analog model sn(t)=x(t)+n(t)
Let's add some noise into the information signal.

```
>>n=randn(size(x)); % Gaussian noise,  see help randn
>>sn=x+n;
```

Plot the signal sn in time and frequency domain.

3. Multiplication of signals

An analog model m(t)=x(t)*y(t)

```
>> y=sin(2*pi*60*t); % 60 Hz signal
>> m=x.*y; % multiplication (Note element-by-element multiplication .*)
```

Plot the signal m in time and frequency domain. What are the resulting frequencies in m.
(use title, xlabel, ylabel)

4. Random sine wave

Generate a sine wave with random amplitude and frequency using the code below.

```
xx= abs(randn(1))*sin(2*pi*abs(randn(1))*Fs/2*t);
```

Determine the frequency and amplitude of xx.
Hint: in Matlab you have to scale the result of fft with the length of the signal to get the correct amplitude in the frequency domain plots.

Generally in frequency domain analysis the TWO SIDED SPECTRUM should be used. See part 1.d in the example code.

Example for Exercise 1

Below there is a commented Matlab script that you can copy to an m-file and run.

1. Start Matlab
2. Open new file with M-file editor
3. Copy and paste the script below to the file
4. Save the file to the work directory (My Documents\MATLAB), use e.g. filenames my_file.m, etc. (do not begin file names or variable names with numbers!)
5. Now you can execute the file by typing the name of the file in the Command window without .m extension:

```
>> my_file
```

6. copy and paste the code below to your m-file. Save and run it once. Then start with parts 2, 3 and 4. There are pause command in the code. In these parts, the execution pauses, and you can continue by pressing any key.

```
% Part 1.a Generating the signal
close all   % closes all figures
clear all   % clears all variables from the Workspace memory
clc         % clears the command window

Fs=650; %
Ts=1/Fs; %
N=2000;            % number of samples wanted, vector length
%( better N=Fs; or N=Fs*2 or N=Fs/2...) why ????
Fo=1/(Ts*N)        % frequency resolution
t=0:Ts:(N-1)*Ts;                              % time instances for calculating the sine function
x=sin(2*pi*35*t);  % x= sin (2 pi f t)


% Part 1.b Plotting the signal in time domain
figure(1)   % Opens or switches to figure window 1.
plot(t,x)
title('Time domain Plot of x')
xlabel('t [ s ]')
ylabel('Amplitude')
pause
axis([0 0.2 -1.2 1.2])
pause

% Part 1.c. Since the signal is real we can plot only the positive frequencies
figure(2)   % Opens or switches to figure window 1.
Fx=fft(x);  % DFT of x
Nx=length(x);   %
Fo= 1/(Ts*Nx);
f=0:Fo:Fs-Fo;                            % frequency vector from 0 – ((N-1)/N) 2 pi = 0 – Fs-Fo
plot(f(1:Nx/2),abs(Fx(1:N/2)))       % we plot only the first halves of f and Fx. 0-Fs/2

title('Frequency domain Plot of x')
```

```
xlabel('f [ Hz ]')
ylabel('Amplitude')
pause

% Part 1.d Plotting the two sided spectrum using fftshift.
% fftshift moves negative frequencies to the beginning.
% see help fftshift

figure(3)                          % Opens a new figure window
f=-Fs/2:Fo:Fs/2-Fo;                % frequency vector from -Fs/2 - Fs/2
plot(f,fftshift(abs(Fx))/Nx)     % two sided amplitude spectrum with fftshift

title('Two sided Frequency domain Plot of x')
xlabel('f [ Hz ]')
ylabel('Amplitude')
pause



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 2. Adding noise to the signal
disp('Part 2:')
Fs  % Variable name without ; prints the value to Command window
N,Fo

% fill in the code for generating noise vector and adding that to the sine vawe

%  in addition Signals have to be same size.

% plot the result in time and frequency domain, use fftshift.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 3. Multiplying two signals
disp('Part 3:')

% fill in the code for generating a sine signal and multiplication

% Signals have to be the same size. and  '.*'  operator
% has to be used

% plot the result in time and frequency domain, use fftshift.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Part 4. random sine signal.
disp('Part 4:')
% generate the random sine wave.
% determine the frequency and amplitude.
```