Part 1: MATLAB environment and programming language

Duration: 1 h 00 at home (mandatory)

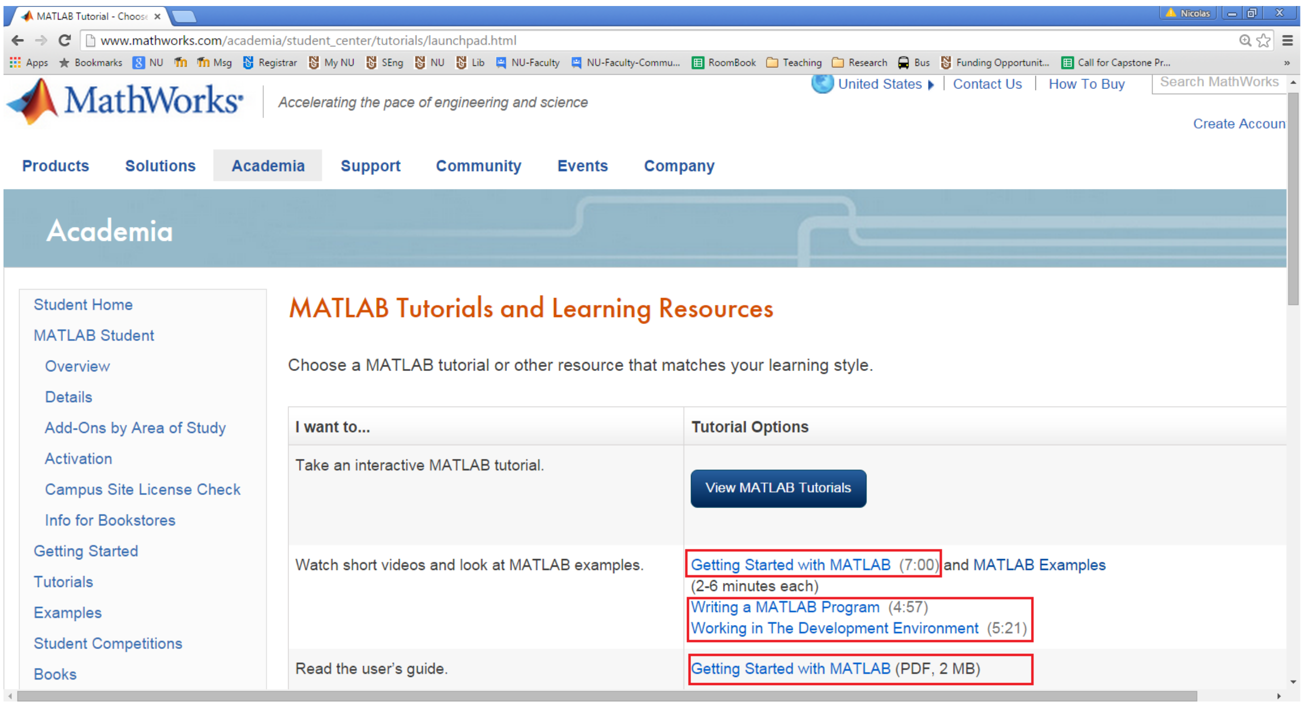
The lab works play a crucial role in assimilating course theory on digital signal and image processing. During the laboratory sessions, we will use a high performance numerical computing environment named MATLAB (abbreviation for matrix laboratory). The MATLAB software is widely used in scientific research and industrial R&D but it is a commercial package. To experiment on your own computer, we recommend you to install a free MATLAB clone called OCTAVE: ftp://ftp.gnu.org/gnu/octave/windows/octave-4.0.0_0-installer.exe. The OCTAVE programming language is almost identical to MATLAB so that the source code is easily portable.

The objective of this first part of the lab is to get you familiar with the MATLAB environment and its basic functionalities. For that purpose, please open the following web page

http://www.mathworks.com/academia/student_center/tutorials/launchpad.html

and study the items squared in red (**Figure 1**). You must read the following sections of the user's guide:

- Quick Start (from 1-1 to 1-31) and
- Language Fundamentals (from 2-1 to 2-15).



MATLAB Tutorials and Learning Resources

Choose a MATLAB tutorial or other resource that matches your learning style.

I want to...	Tutorial Options
Take an interactive MATLAB tutorial.	View MATLAB Tutorials
Watch short videos and look at MATLAB examples.	Getting Started with MATLAB (7:00) and MATLAB Examples (2-6 minutes each) Writing a MATLAB Program (4:57) Working in The Development Environment (5:21)
Read the user's guide.	Getting Started with MATLAB (PDF, 2 MB)

***Figure 1:** Screenshot of MATLAB tutorials and learning resources web site.*

Important note: to display documentation of a MATLAB function, use **help** or **doc** command (e.g. **doc cos**).

Report your results in the file:EEE238_Lab 1_DSP - Sampling and Quantization – Report template.doc

Part 2: Signal generation, sampling and quantization

Duration: 2 h 00

Exercise 2.1: Basic digital signals

(a) Write a MATLAB program to generate and display (using the **stem** function) the signals defined in **Table 1**. The MATLAB code of the first signal (dirac) is given in the report template as an example.

(b) Write a MATLAB function **[x, t] = sin_NU(f0, fs, T)** to generate a sine signal. The output parameters **x** and **t** are the signal and time vectors, respectively. The input parameters are **f0** (signal frequency in Hz), **fs** (sampling frequency in Hz), **T** (signal duration in sec.).

(c) Test your **sin_NU** function with the input parameter values $\{\ f_0 = 10, \quad f_s = 1000, \quad T = 0.5\ \}$ and display the result using the **plot** function.

Table 1: List of basic digital signals to generate.

Function	**Equation**	**Parameters**
Dirac (unit impulse)	$\partial(n) = \begin{cases} 1 & \text{if} \quad n = 0 \\ 0 & \text{otherwise} \end{cases}$	$n = -N/2, \ldots, N/2$ $N = 10$
Unit step (Heaviside step)	$u(n) = \begin{cases} 1 & \text{if} \quad n \geq 0 \\ 0 & \text{otherwise} \end{cases}$	$n = -N/2, \ldots, N/2$ $N = 10$
Sign	$x(n) = 2u(n) - 1$	$n = -N/2, \ldots, N/2$ $N = 10$
Rectangle	$\text{rect}_M(n) = u(n+M) - u(n-M)$	$n = -N/2, \ldots, N/2$ $N = 10, \quad M = 2$
Sine	$x(n) = x_0 \sin(2\pi\, f_0 n T_s)$	$n = 0, \ldots, L-1$ $L = 20,$ $f_0 = 100\,\text{Hz}, \quad 1/T_s = f_s = 1\text{kHz}$
Sine cardinal	$x(n) = \begin{cases} 1 & \text{for} \quad n = 0 \\ \dfrac{\sin(\pi n T_s)}{\pi n T_s} & \text{otherwise} \end{cases}$	$n = -L, \ldots, L$ $L = 50$ $T_s = 0.1\,\text{sec}$

Exercise 2.2: Audio aliasing

To illustrate the aliasing phenomenon, let's perform two simple experiments allowing us to "hear" it. Using the **sin_NU** function of Exercise 1:

(a) Generate two 1 kHz sine signals (2 seconds duration), first signal at 20 kHz sample frequency and second signal at 1.5 kHz sample frequency;

(b) On the same graph, use the **plot** function to display the two signals versus t in the range $0 \leq t \leq 5$ msec.;

(c) Listen to the two signals one after another using the function **soundsc(x, fs)**; and

(d) Give your interpretation of this listening.

Exercise 2.3: Quantization

Quantization is done by replacing each value of an analog signal $x(t)$ by the value of the nearest quantization level. To exemplify this operation, let's simulate an unipolar ADC (Analog to Digital Converter) having the technical specifications: R = 10 Volts (full-scale range) and B = 3 (number of bits).

(a) Write a MATLAB function **y = adc_NU(x, R, B)** where **x** and **y** are vectors containing the input signal and the quantized signal, respectively;

(b) Test your function with an input ramp signal ranging from -5 to 15 Volts (1 volt per step); and

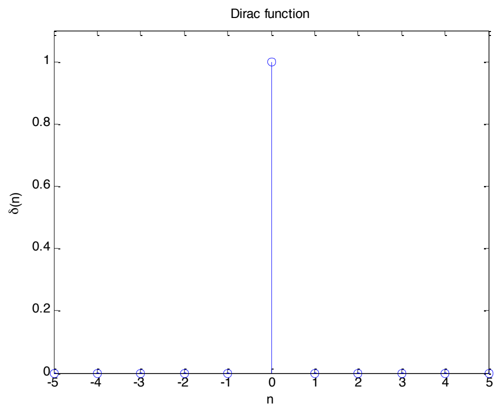
(c) On the same graph, use the **plot** and **stem** functions to display the input signal and quantized signal, respectively.

Lab group #: Session 3, Friday 2 p.m.
Student name: Sanzhar Askaruly
Student ID #: 2011XXXXX

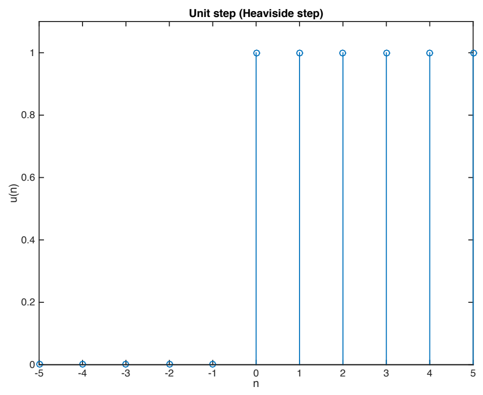
Part 2: Signal generation, sampling and quantization

Exercise 2.1: Basic digital signals

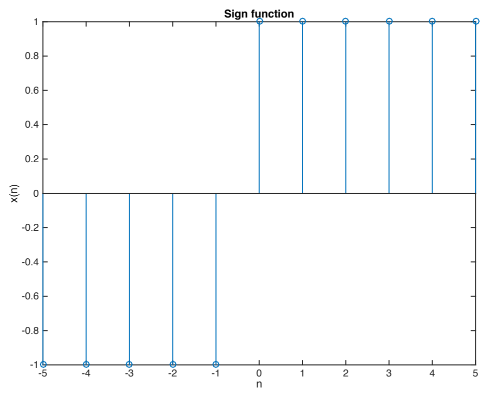
```
% Dirac
N = 10;          % number of samples
n = -N/2:N/2;    % vector
d = [zeros(1,N/2) 1 zeros(1,N/2)];
figure;          % display
stem(n,d);
xlabel('n');
ylabel('\delta(n)');
title('Dirac function');
axis([-N/2 N/2 0 1.1]);
```



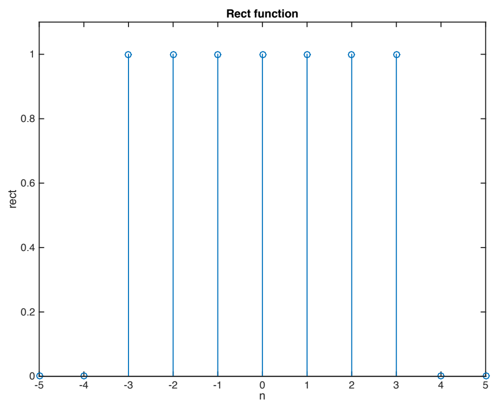
```
% Unit step function
N = 10;          % number of samples
n = -N/2:N/2;    % vector
u = [zeros(1,N/2) 1 ones(1,N/2)]; %logic to give unit
step
figure;          % display
stem(n,u);
xlabel('n');
ylabel('u(n)');
title('Unit step function'); %title
axis([-N/2 N/2 0 1.1]);
```



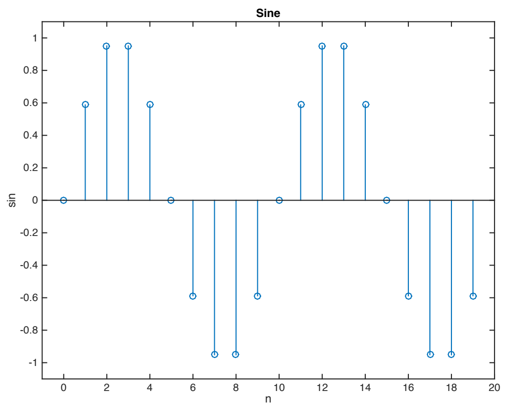
```
% Sign function
N = 10;          % number of samples
n = -N/2:N/2;    % vector
u = [zeros(1,N/2) 1 ones(1,N/2)]; %logic for sign
function
x = 2.*u-1;
figure;          % display
stem(n,x);
xlabel('n');
ylabel('x(n)');
title('Sign function');
axis([-N/2 N/2 -1 1]);
```



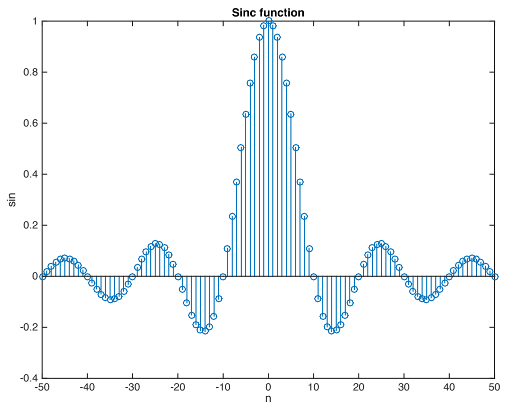
```
% Rect function
M = 3;
N = 10;          % number of samples
n = -N/2:N/2;    % vector
Rect = [zeros(1,N/2-M) ones(1,2*M+1) zeros(1,N/2-M)];
%logic for rect
figure;          % display
stem(n,Rect);
xlabel('n');
ylabel('rect');
title('Rect function');
axis([-N/2 N/2 0 1.1]);
```



```
% Sine function
L = 20;
n = 0:L-1;
f0 = 100; %initial frequency
fs = 1000; %sampling frequency
x0 = 1;
x = x0*sin(2*pi*f0/fs*n); %sine function description
figure; % display
stem(n,x);
xlabel('n');
ylabel('sin');
title('Sine');
axis([-1 L -1.1 1.1]);
```



```
% Sinc function
L = 50;
n = [-L:L];
Ts = 0.1;
x = zeros(1,length(n));
x(n~=0) = sin(pi*n(n~=0)*Ts)./(pi*n(n~=0)*Ts); %logic
for sinc
x(n==0) = 1;
figure; % display
stem(n,x);
xlabel('n');
ylabel('sin');
title('Sinc function');
```

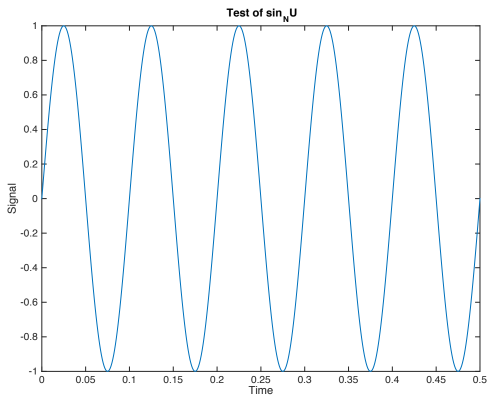


```
function [x, t] = sin_NU(fs, f0, T)  %function to
generate sine signal

t = 0:1/fs:T; %the signal vector output
x = sin(2*pi*f0*t); %the time vector output
end
```

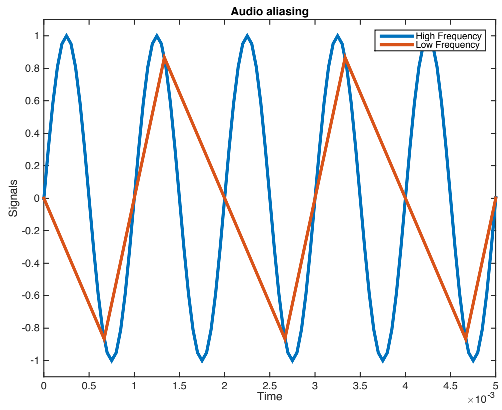
```
% Sine function test

[x,t]=sin_NU(1000,10,0.5);
figure(7);
plot(t,x);
xlabel('Time');
ylabel('Signal');
title('sin_NU function test');
```



Exercise 2.3: Audio aliasing

```
T = 2; %parameters
f0 = 1000;
fs1 = 20000;
fs2 = 1500;
[x1, t1] = sin_NU(fs1,f0,T);
[x2, t2] = sin_NU(fs2,f0,T);
figure;
plot(t1,x1,t2,x2,'LineWidth',3.0),
axis([0, 0.005, -1.1, 1.1])
legend('High Frequency','Low Frequency')
xlabel('Time')
ylabel('Signals')
title('Audio aliasing');
%%%
soundsc(x1,fs1)
%%%
soundsc(x2,fs2)
```



Interpretation of this listening:

From sampling theorem, sampling frequency should be twice more than the signal frequency. From the data above, we have two sampling frequencies. For 1500 Hz sampling frequency, the maximum spectrum that can be heard is 750 Hz signal. As for 20000 Hz, the maximum frequency is 10000 Hz. Since the sound's initial frequency is 1000 Hz, it cannot be heard with original frequency for 1500 Hz sampling rate, therefore it is heard as 750 Hz signal. As for 20000 Hz sampling frequency, there is no problem. It is heard as original sound, That's why, there is slightly difference in the sounds produced.

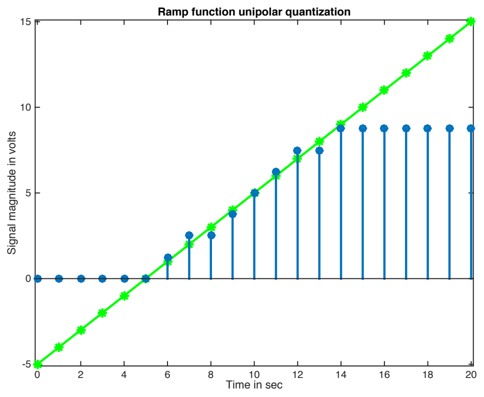
Exercise 2.3: Quantization

```
function y = adc_NU(x, R, B)
level = [0:R/(2^B):R-R/(2^B)];
temp = [-Inf,(level(2:end)-R/(2^(B+1))),Inf];
y = zeros(1,length(x));
for i = 1:length(level)
    y = y + (x >= temp(i)).*(x < temp(i+1)).*level(i);
end



% ADC_NU function test

R = 10;
B = 3;
x = -5:15;
y = adc_NU(x,R,B);
t = 0:length(x)-1;
figure(11)
plot(t,x,t,y)
plot(t,x,'g-*','LineWidth',2.2)
hold on
stem(t,y,'filled','LineWidth',2.2)
hold off
title('Ramp function unipolar quantization')
xlabel('Time in sec')
ylabel('Signal magnitude in volts')
axis([-0.1,20.1,-5.1,15.1])
```


Ramp function unipolar quantization
Signal magnitude in volts
Time in sec