Digital Signal Processing

Lab3: sampling, quantization and signal operations

Instructor: Eng\ Samar Shaaban
E-mail: ssa10@fayoum.edu.eg
Github Repo: https://github.com/SamarShabanCS/DSP
Slack workspace: https://fayoum-university-fci.slack.com

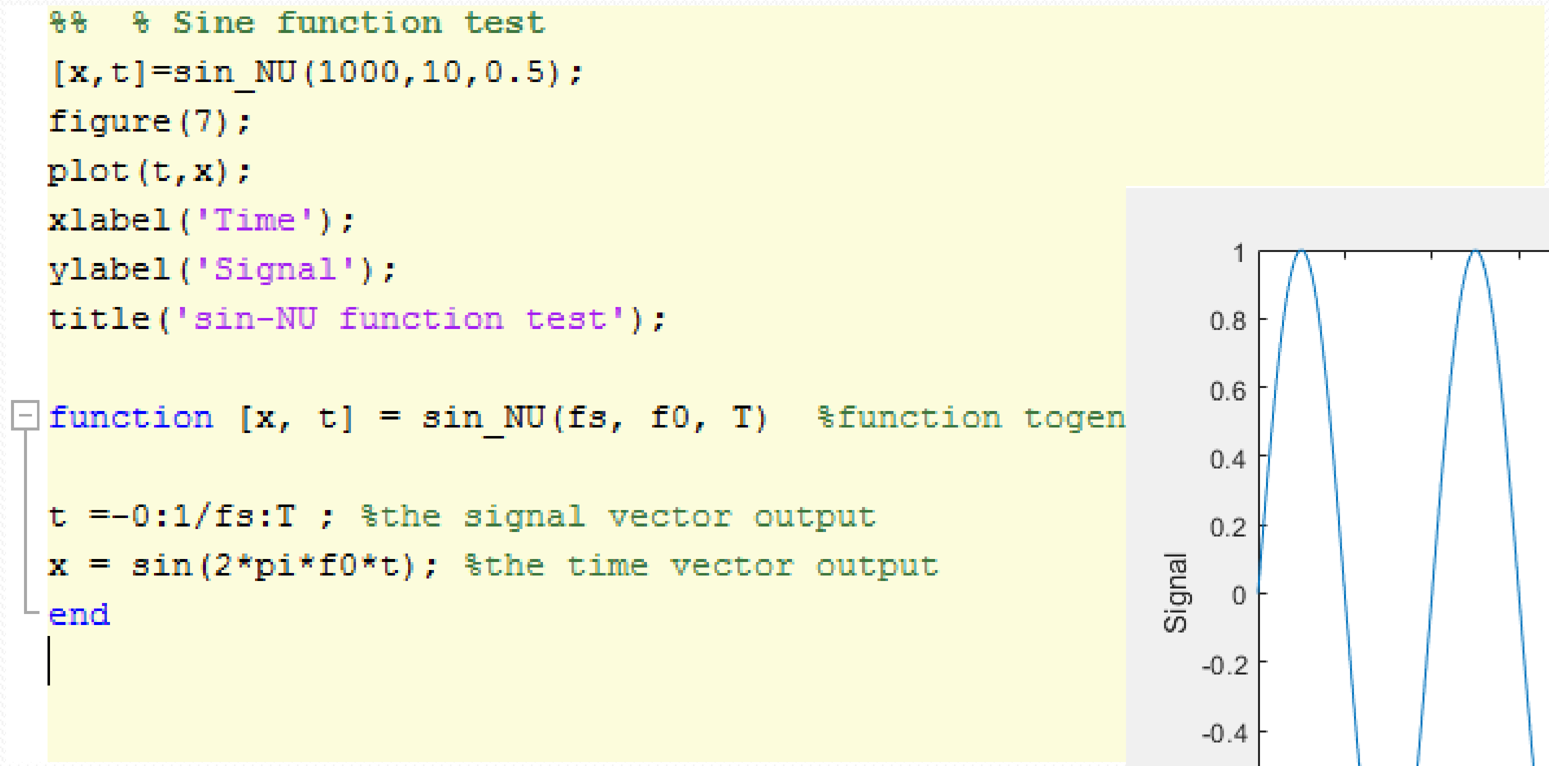
Sheet1 Discussion

Exercise

- 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.).

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

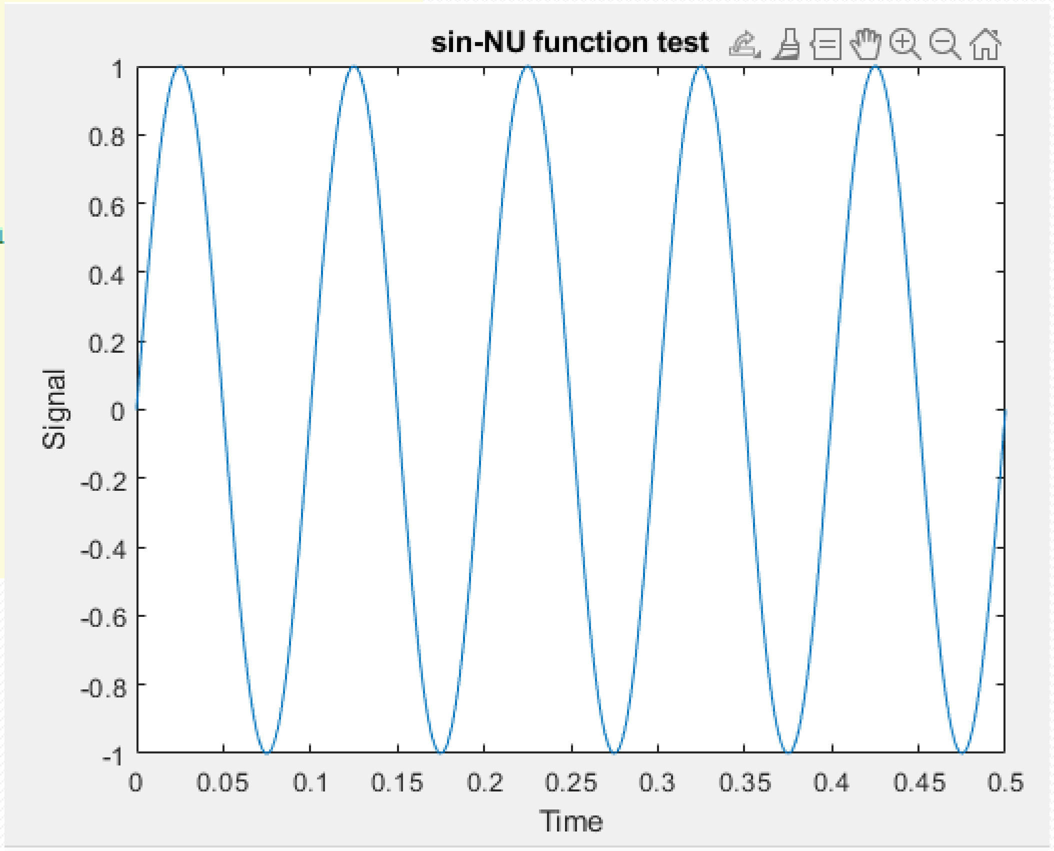
Sol



```
%%  % 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');

function [x, t] = sin_NU(fs, f0, T)  %function togen

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

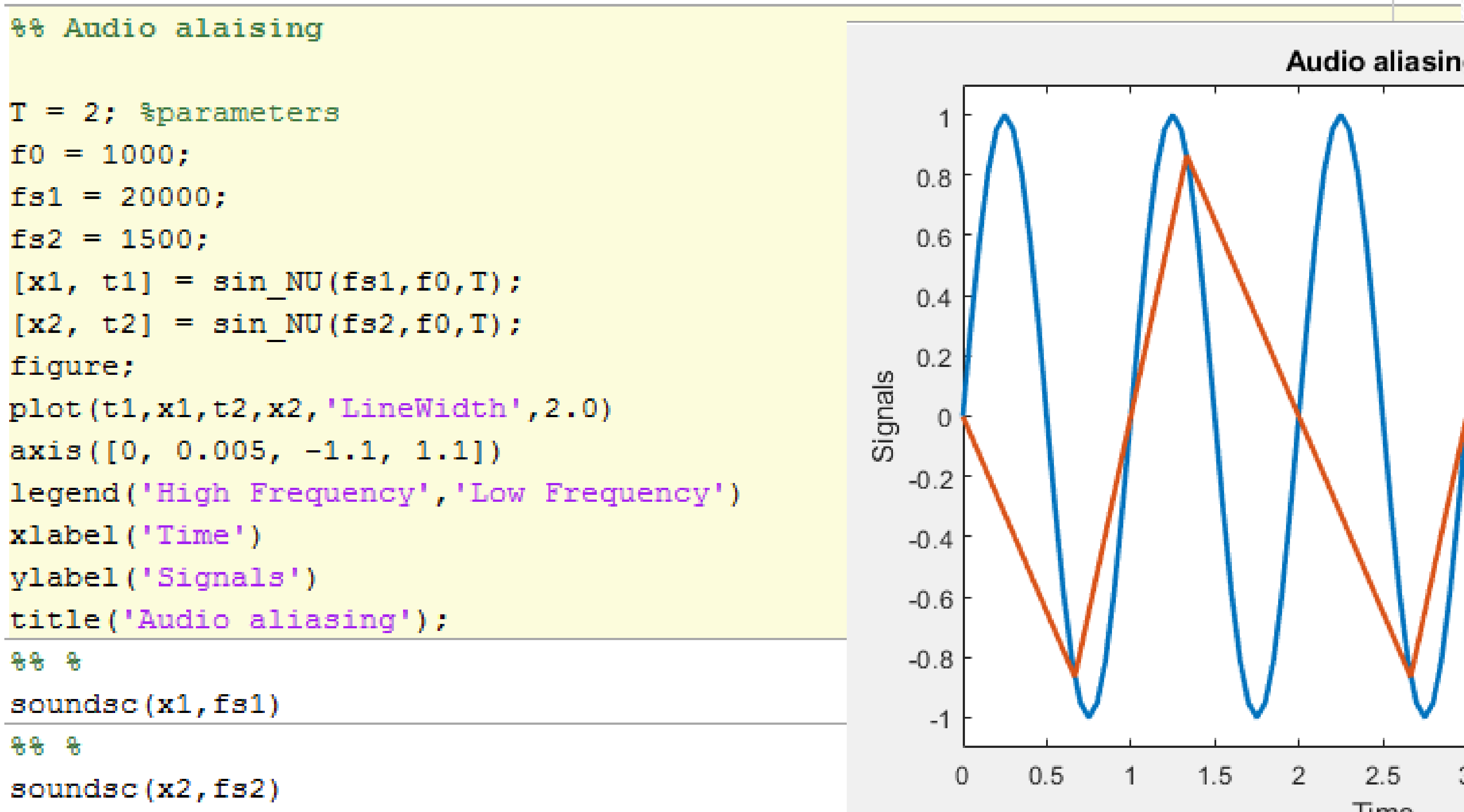


Exercise : Audio aliasing

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

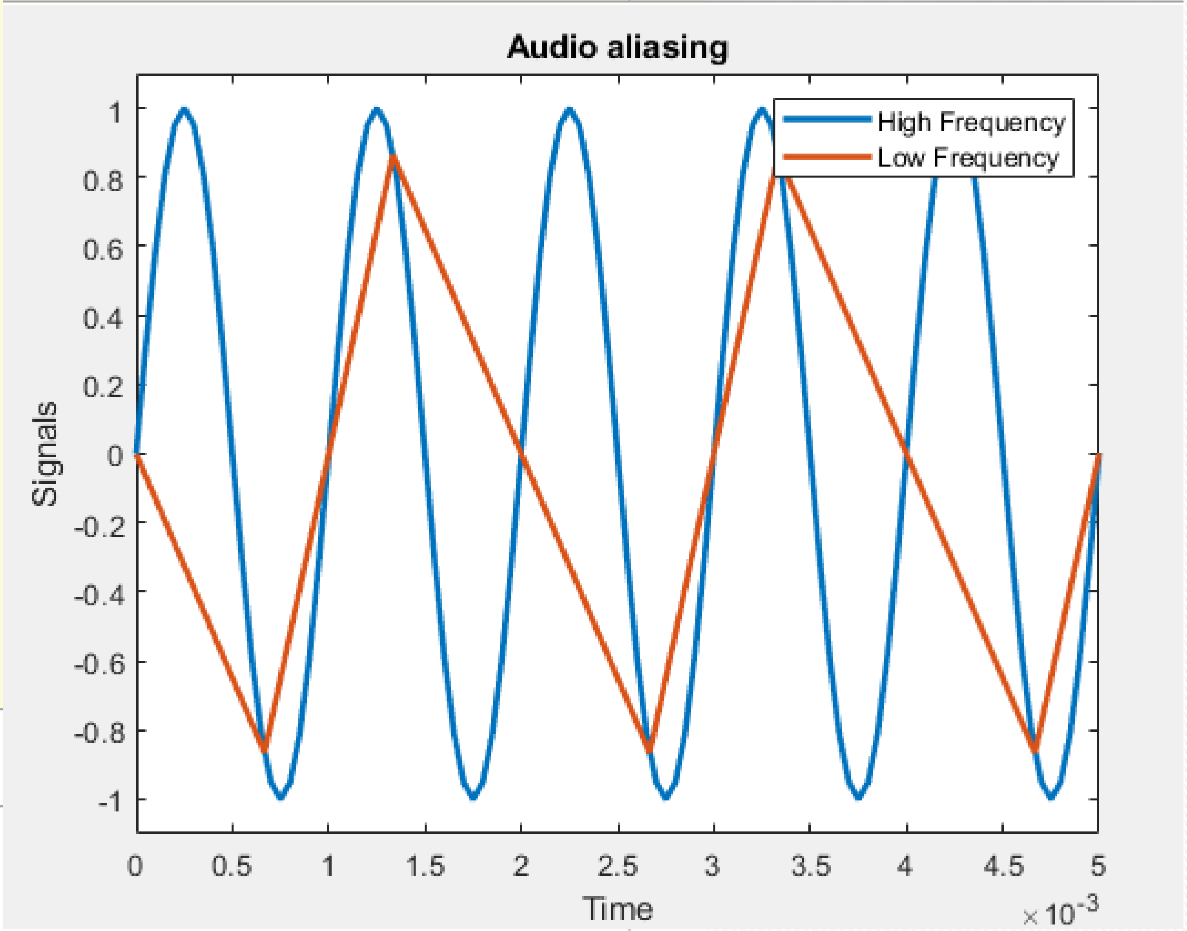
1. Generate two 1 kHz sine signals (2 seconds duration), first signal at 20 kHz sample frequency and second signal at 1.5 kHz sample frequency;
2. On the same graph, use the **plot** function to display the two signals versus *t in the range* $0 \leq t \leq 5$ msec.;
3. Listen to the two signals one after another using the function **soundsc(x, fs);** and Give your interpretation of this listening.

Sol



```
%% Audio alaising

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',2.0)
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.

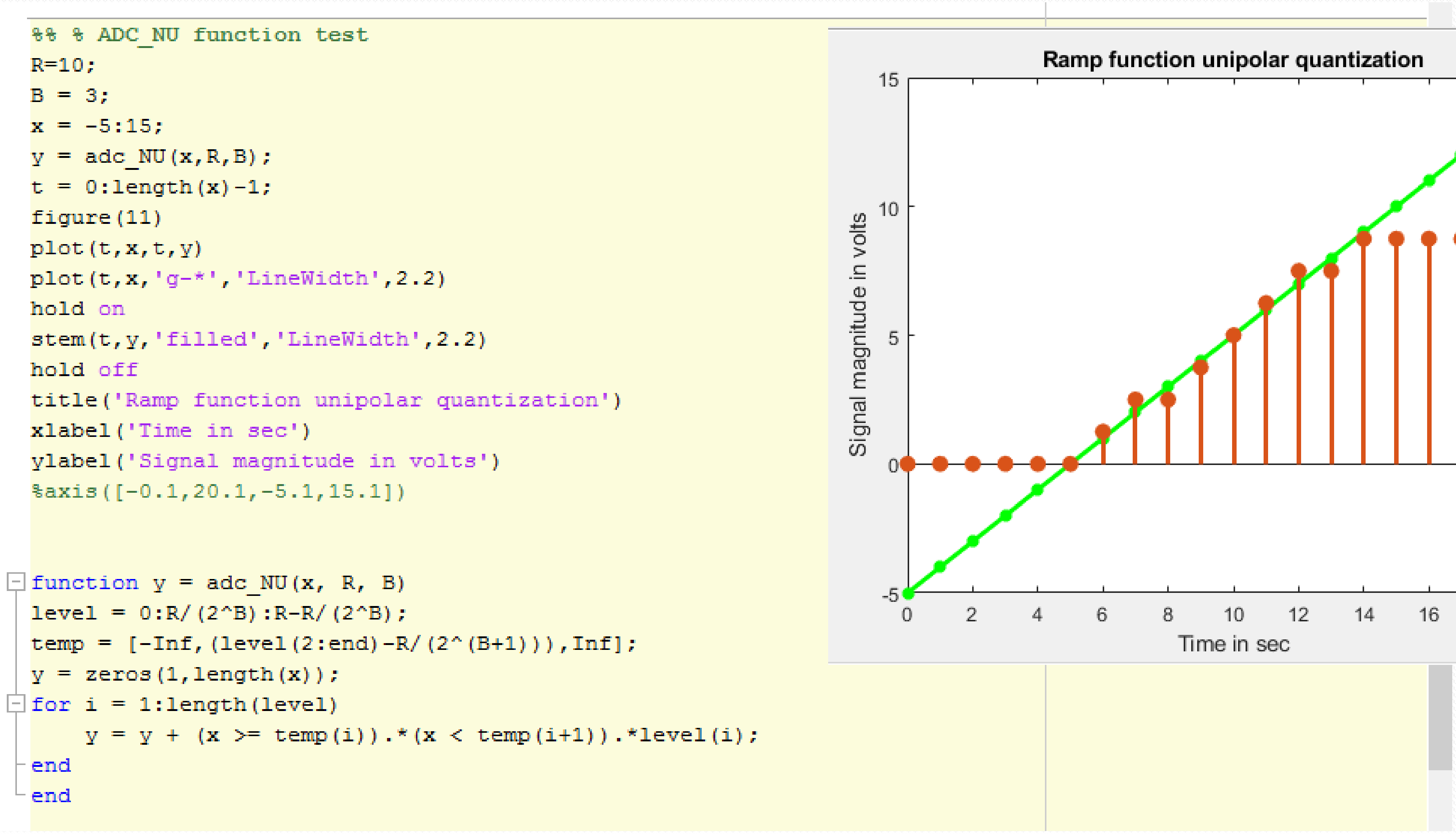
From the data above, we have two sampling frequencies. For 1500 Hz sampling frequency, the maximum spectrum that can be heard is 750 Hz signal. As for 20000 Hz, the maximum frequency is 10000 Hz. Since the sound's initial frequency is 1000 Hz, it cannot be heard with original frequency for 1500 Hz sampling 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: 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).*

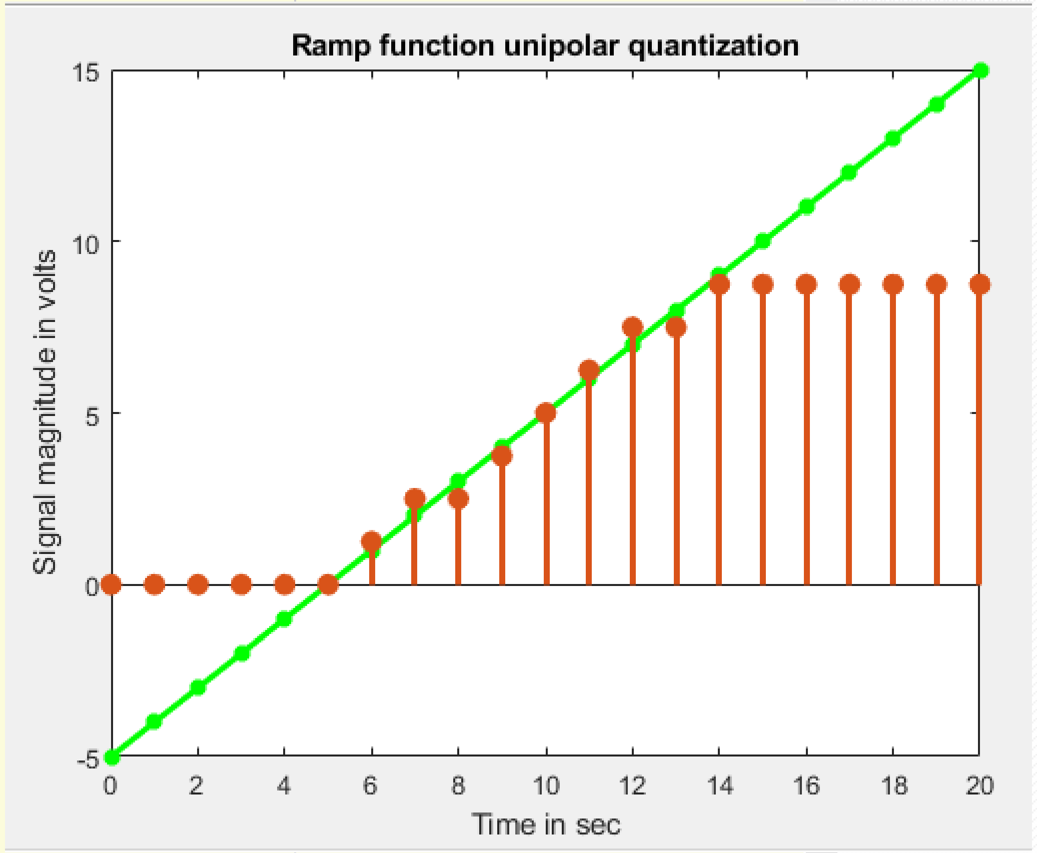
1. Write a MATLAB function **y = adc_NU(x, R, B)** where x and y are vectors containing the input signal and the quantized signal, respectively;
2. Test your function with an input ramp signal ranging from -5 to 15 Volts (1 volt per step);
3. On the same graph, use the **plot and stem** functions to display the input signal and quantized signal, respectively.

Sol



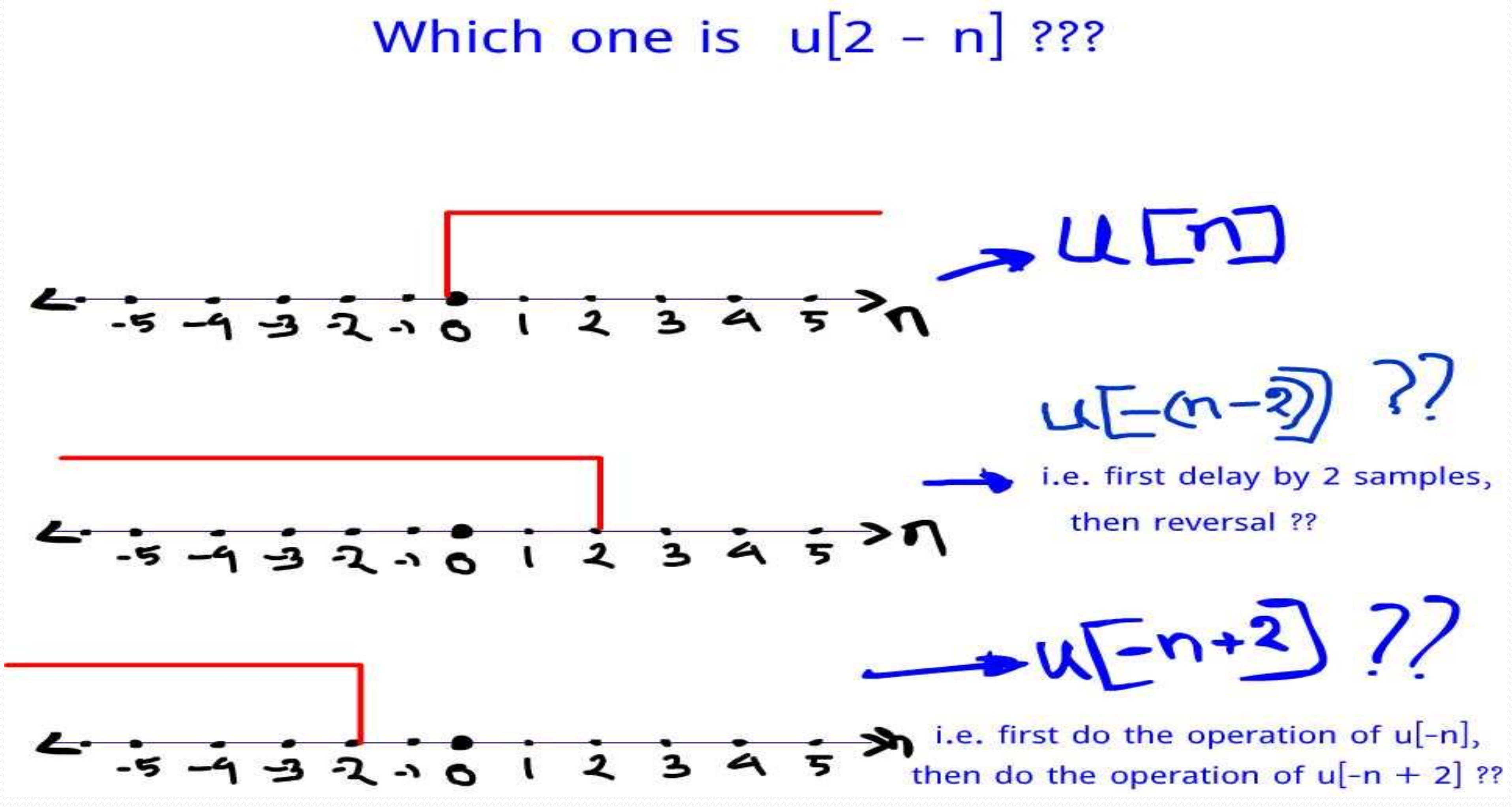
```
%% % ADC_NU function test
R=10;
B = 3;
x = -5:15;
y = adc_NU(x,R,B);
t = 0:length(x)-1;
figure(11)
plot(t,x,t,y)
plot(t,x,'g-*','LineWidth',2.2)
hold on
stem(t,y,'filled','LineWidth',2.2)
hold off
title('Ramp function unipolar quantization')
xlabel('Time in sec')
ylabel('Signal magnitude in volts')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


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
end
```



STEPS To Remember

- If you have Time shifting and Reversal (Reflection) together • Do 1st Shifting • Then Reflection


Which one is u[2 − n] ???
u[n]
n
-5 -4 -3 -2 -1 0 1 2 3 4 5
u[−(n−2)] ??
i.e. first delay by 2 samples,
then reversal ??
u[−n+2] ??
i.e. first do the operation of u[−n],
then do the operation of u[−n + 2] ??

- Write a MATLAB program to generate and display (using the **stem function) the signals defined in Table 1.**

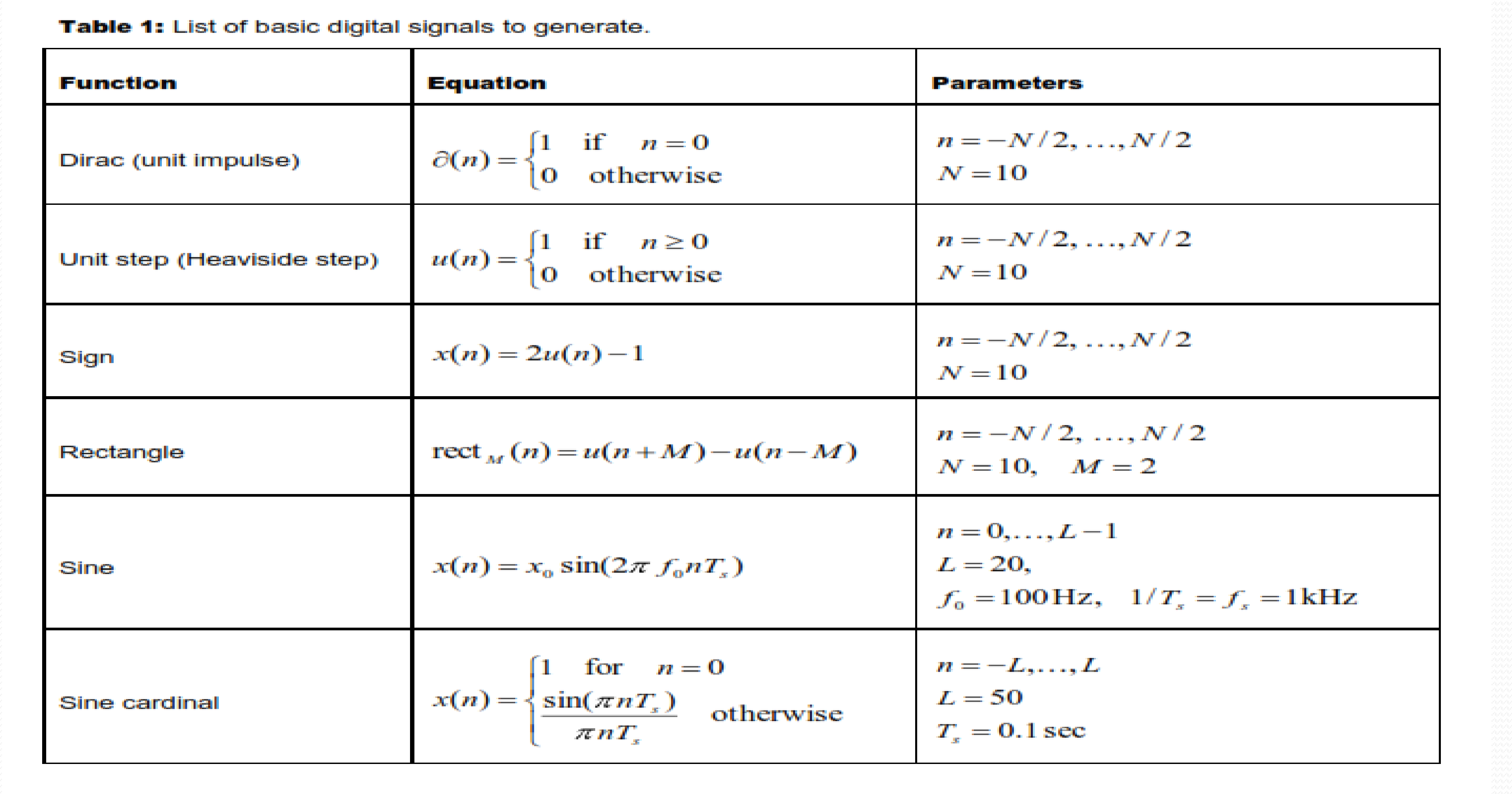
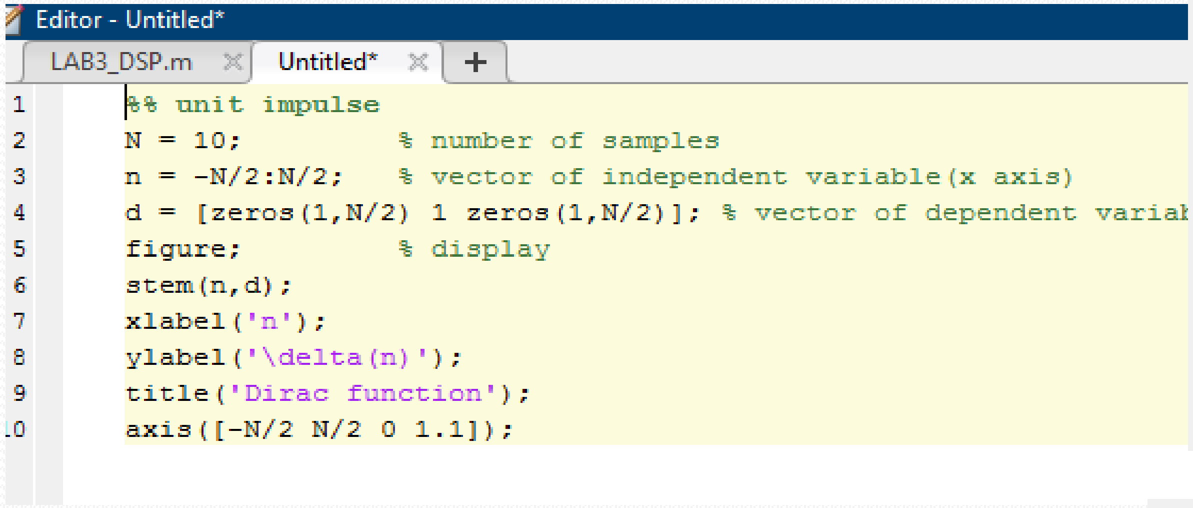
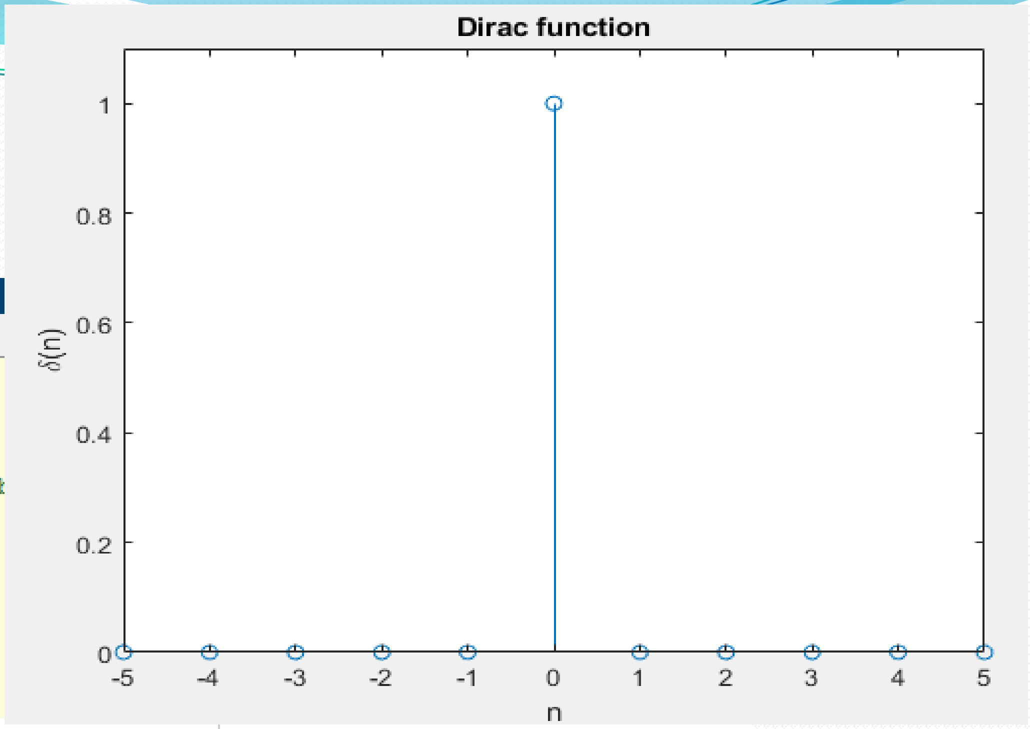


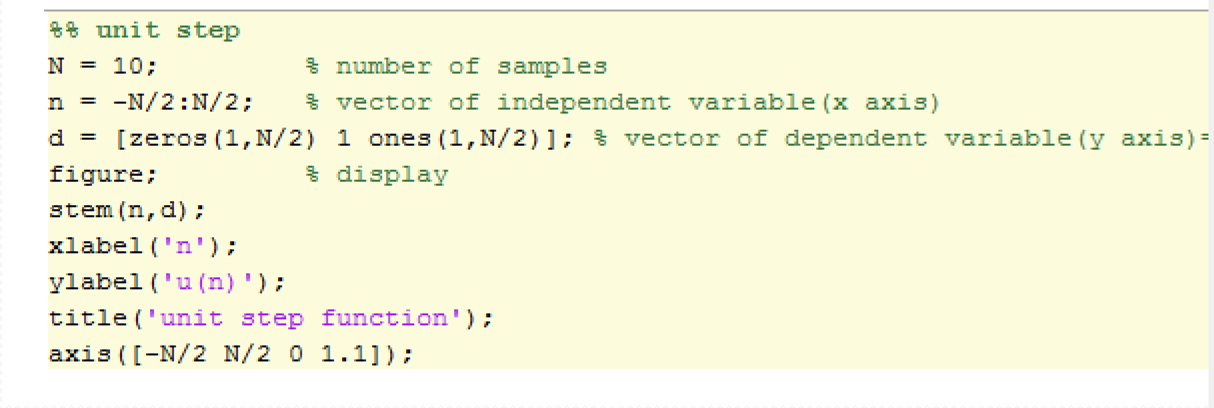
Table 1: List of basic digital signals to generate.

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	$x(n) = \begin{cases} 1 & \text{for} \quad n = 0 \\ \dfrac{\sin(\pi n T_s)}{\pi n T_s} & \text{otherwise} \end{cases}$	$n = -L, \ldots, L$ $L = 50$ $T_s = 0.1\,\text{sec}$

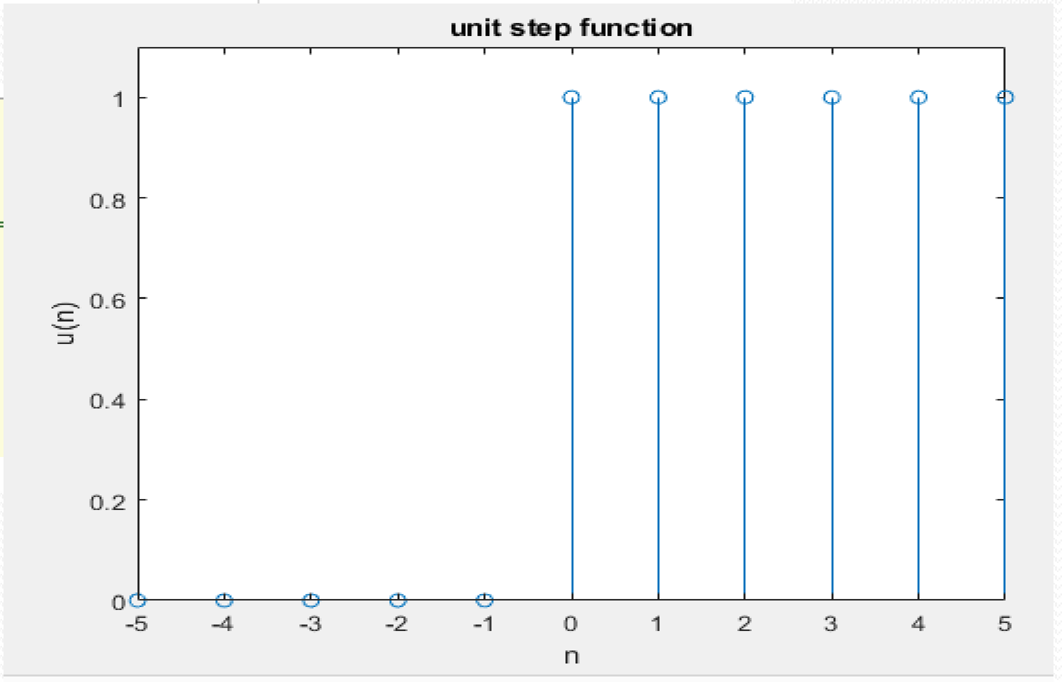


```
%% unit impulse
N = 10;           % number of samples
n = -N/2:N/2;    % vector of independent variable(x axis)
d = [zeros(1,N/2) 1 zeros(1,N/2)]; % vector of dependent varia
figure;           % display
stem(n,d);
xlabel('n');
ylabel('\delta(n)');
title('Dirac function');
axis([-N/2 N/2 0 1.1]);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

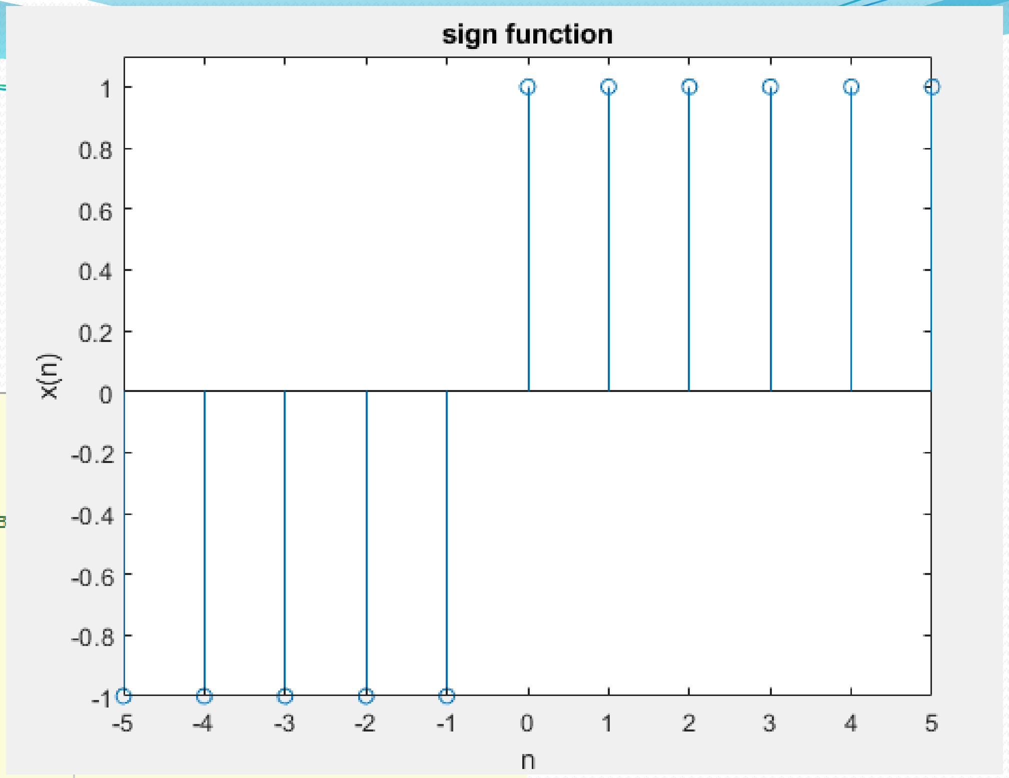




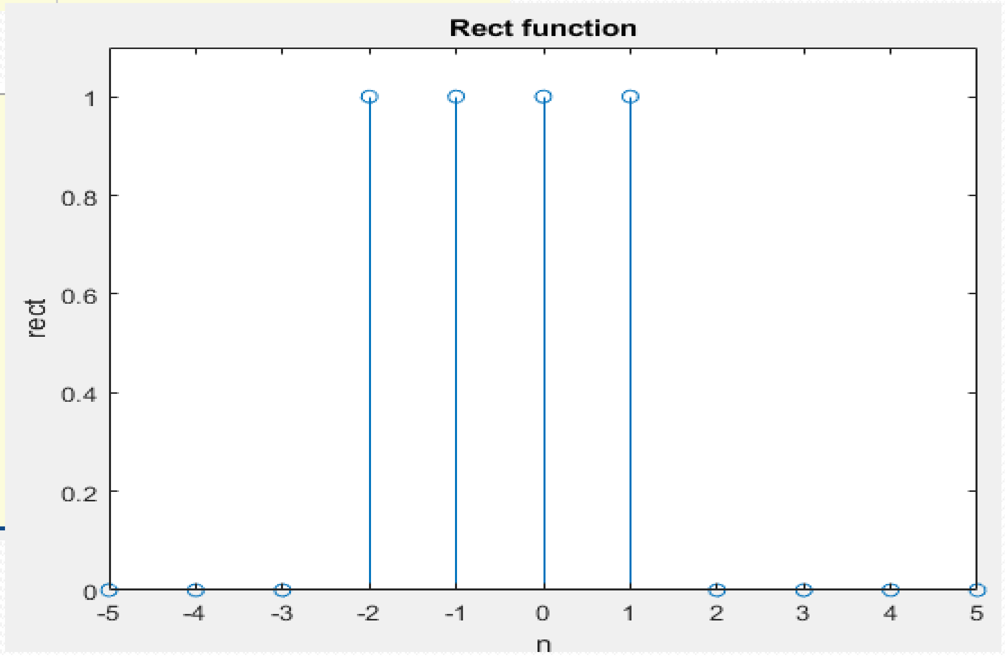
```
%% unit step
N = 10;           % number of samples
n = -N/2:N/2;    % vector of independent variable(x axis)
d = [zeros(1,N/2) 1 ones(1,N/2)]; % vector of dependent variable(y axis)=
figure;           % display
stem(n,d);
xlabel('n');
ylabel('u(n)');
title('unit step function');
axis([-N/2 N/2 0 1.1]);
```

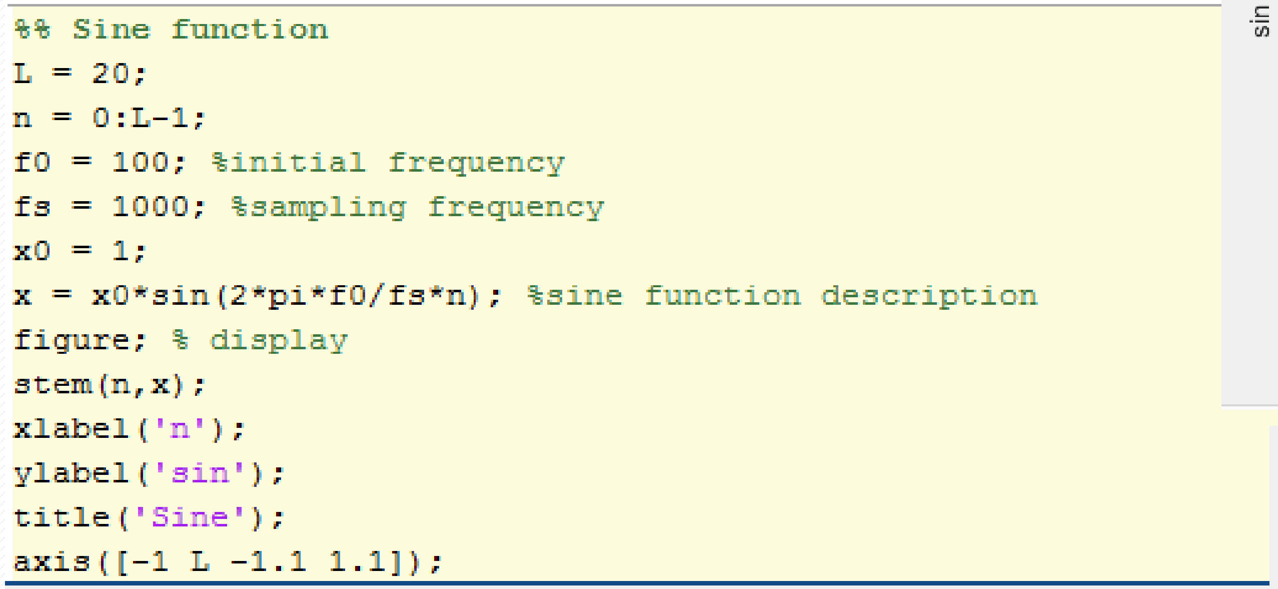


```
%% sign function
N = 10;          % number of samples
n = -N/2:N/2;    % vector of independent variable(x axis)
d = [zeros(1,N/2) 1 ones(1,N/2)]; % vector of dependent variable(y axis
x = 2.*d-1; % 2u(2)-1
figure;          % display
stem(n,x);
xlabel('n');
ylabel('x(n)');
title('sign function');
axis([-N/2 N/2 -1 1.1]);
```

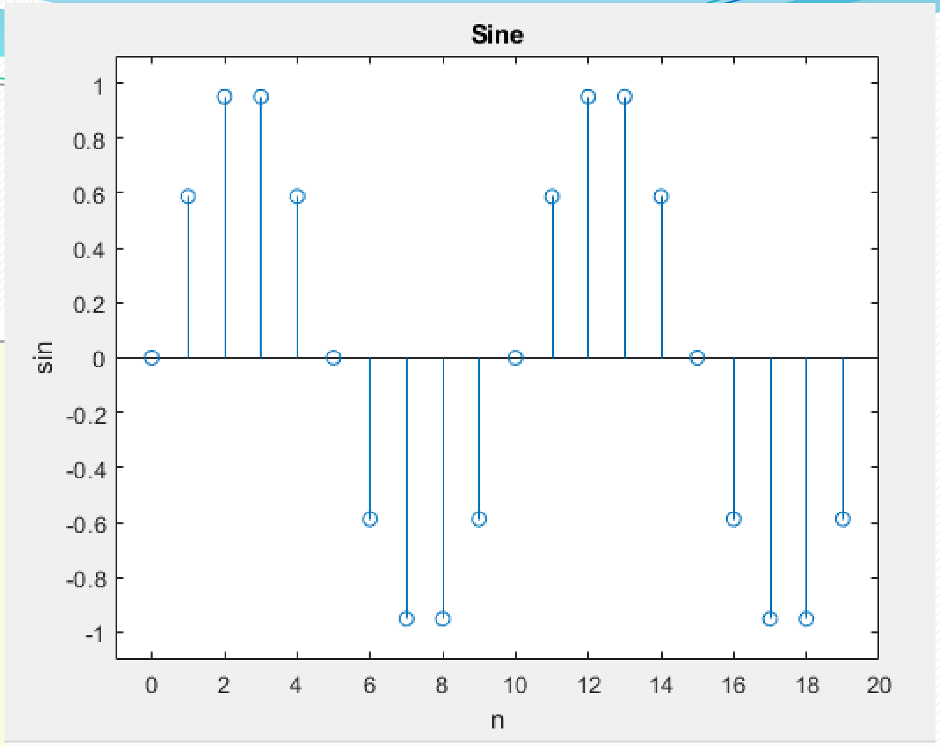


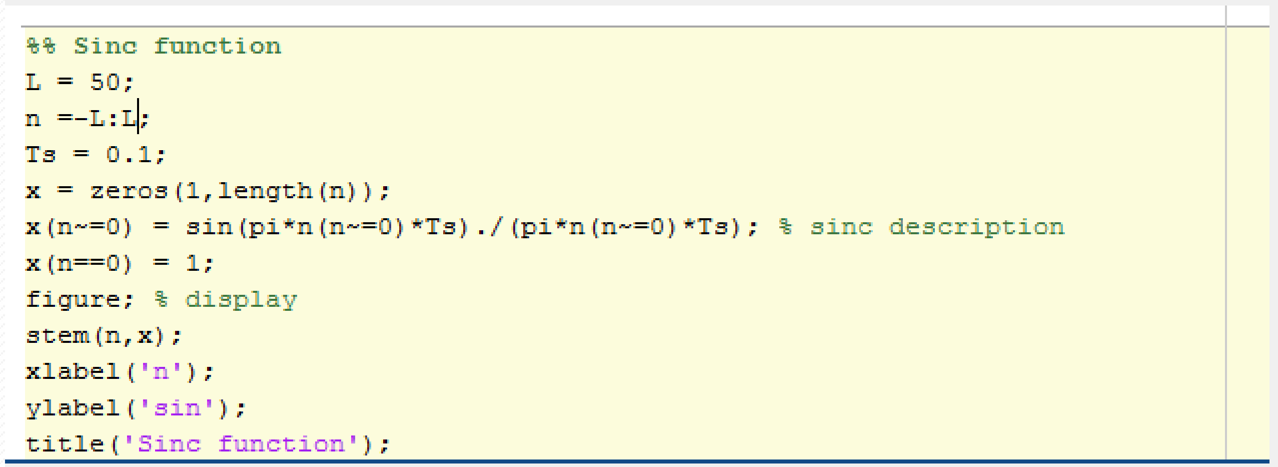
```
%% Rect function
M = 2;
N = 10;          % number of samples
n = -N/2:N/2;    % vector
Rect = [zeros(1,N/2-M) ones(1,2*M) zeros(1,N/2-M+1)];
%logic for rect
figure;          % display
stem(n,Rect);
xlabel('n');
ylabel('rect');
title('Rect function');
axis([-N/2 N/2 0 1.1]);
```



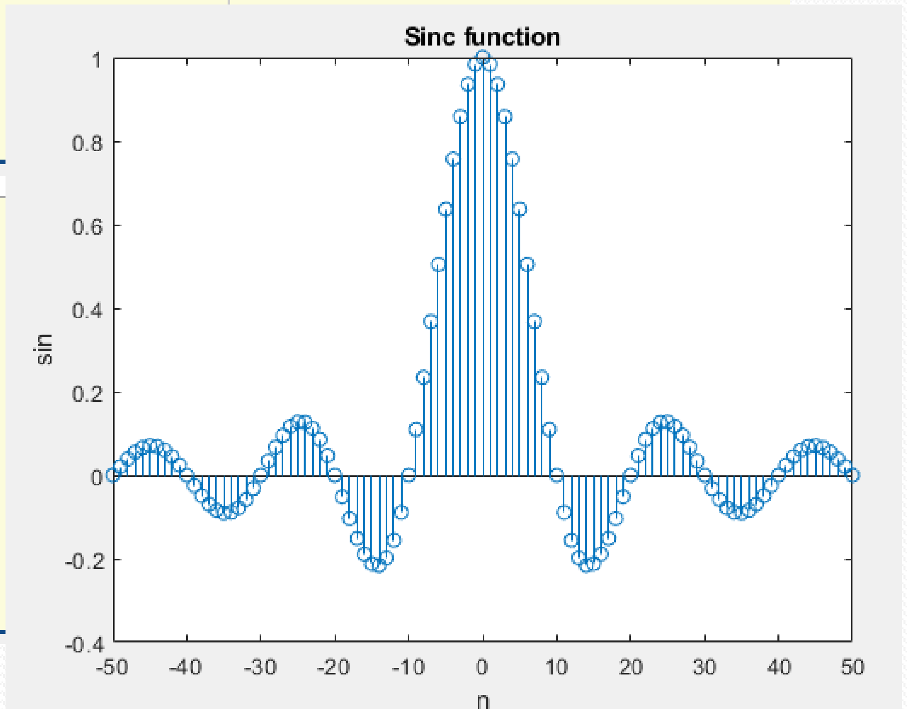


```
%% Sine function
L = 20;
n = 0:L-1;
f0 = 100; %initial frequency
fs = 1000; %sampling frequency
x0 = 1;
x = x0*sin(2*pi*f0/fs*n); %sine function description
figure; % display
stem(n,x);
xlabel('n');
ylabel('sin');
title('Sine');
axis([-1 L -1.1 1.1]);
```





```
%% 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); % sinc description
x(n==0) = 1;
figure; % display
stem(n,x);
xlabel('n');
ylabel('sin');
title('Sinc function');
```


Sinc function
sin
n