Home works:

1. x(n) = sin(n∏/2) & h(n)= cos(n∏)

Determine that system’s output/response for -5<=n<=5

Code :

clc;

clear all;

disp('linear convolution program');

fs=input('Enter sampling frequency : ');

n= -5:(1/fs):5;

x=sin(n\*pi/2);

m1=length(x);

h=cos(n\*pi);

m2=length(h);

x=[x,zeros(1,m2)];

subplot(2,2,1), stem(x);

title('i/p sequence x(n) is :');

xlabel('----->n');

ylabel('-----> x(n)'); grid;

h=[h,zeros(1,m1)];

subplot(2,2,2), stem(h);

title('i/p sequence h(n) is :');

xlabel('----->n');

ylabel('-----> h(n)'); grid;

disp('Linear convolution of x(n) & h(n) is y(n) :');

y=zeros(1,m1+m2-1);

for i=1:m1+m2-1

y(i)=0;

for j=1:m1+m2-1

if(j<i+1)

y(i)=y(i)+x(j)\*h(i-j+1);

end

end

end

y

subplot(2,2,[3,4]), stem(y);

title('Linear convolution of x(n) & h(n) is :');

xlabel('----->n');

ylabel('-----> y(n)');

grid;

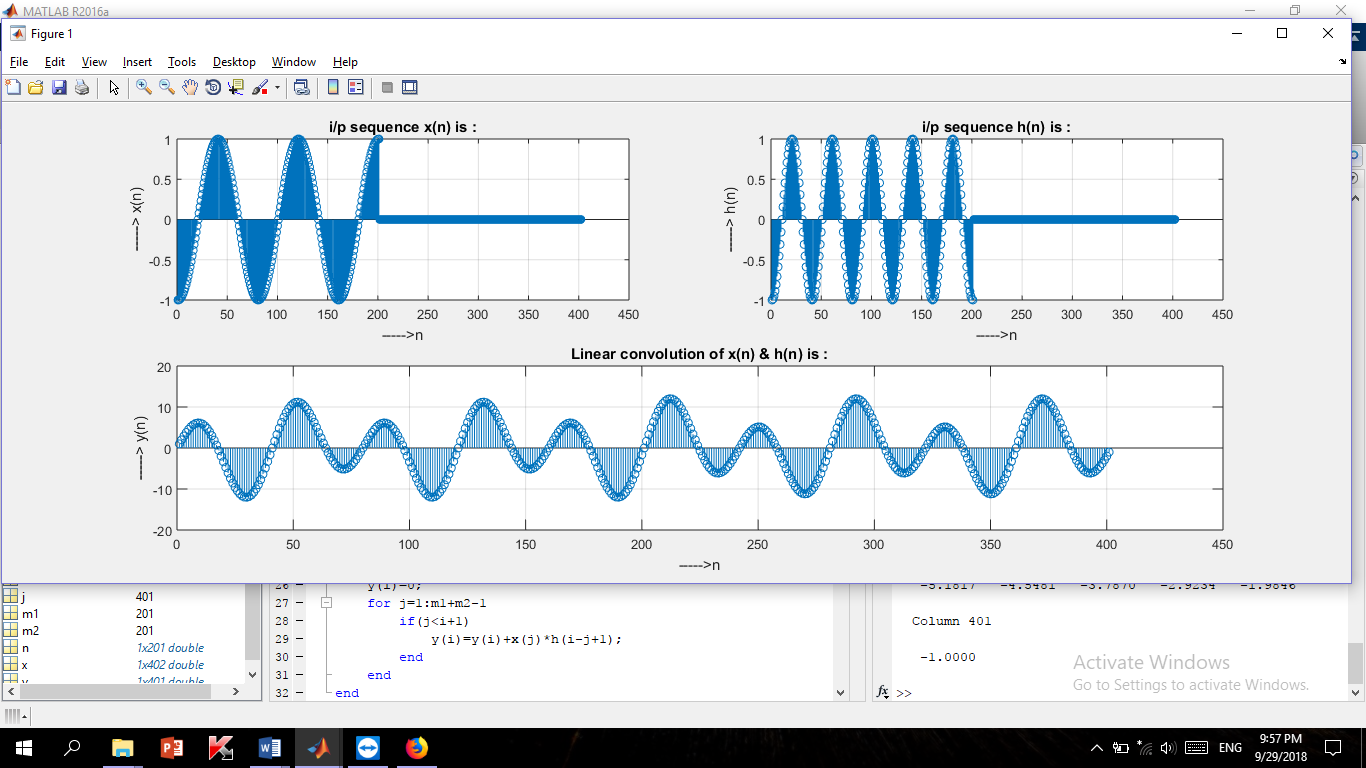


Figure 2.7: convolution between x(n) = sin(n∏/2) & h(n)= cos(n∏)

Class works:

Task: To generate random signals using MATLAB.

N=1024;

R1=randn(1,N); %generate normal random numbers

R2=rand(1,N); %generate uniformly random numbers

figure(1);

subplot('221');

plot(R1);

grid;

title('Normal [Gaussian] Distributed Random Signal');

xlabel('Sample Number');

ylabel('Amplitude');

subplot('222');

hist(R1);

grid;

title('Histogram[Pdf] of a normal Random Signal');

xlabel('Sample Number');

ylabel('Total');

subplot('223');

plot(R2);

grid;

title('Uniformly Distributed Random Signal');

xlabel('Sample Number');

ylabel('Amplitude');

subplot('224');

hist(R2);

grid;

title('Histogram[Pdf] of a uniformly Random Signal');

xlabel('Sample Number');

ylabel('Total');

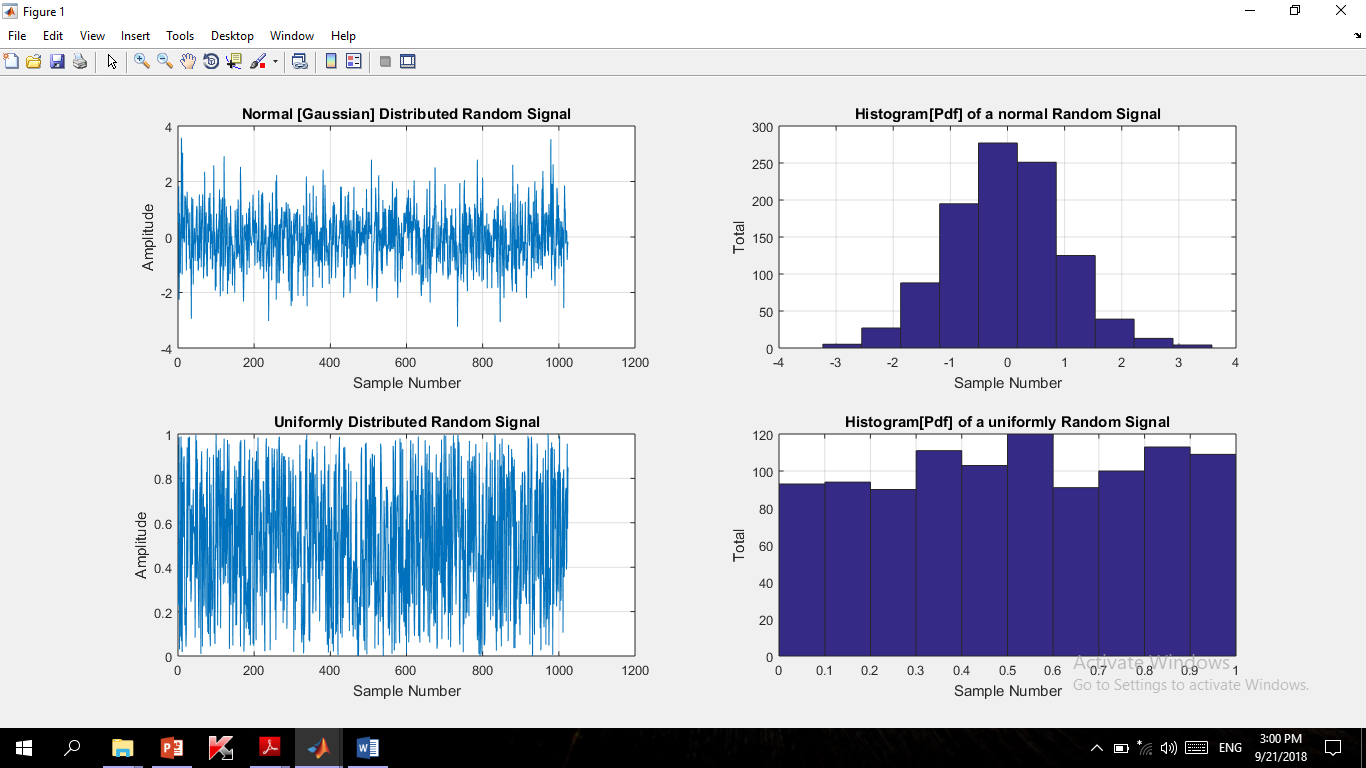


Figure 2.5: generating random signals using MATLAB.

Task: To perform convolution between two vectors using MATLAB.

clc;

clear all;

disp('linear convolution program');

x=input('enter i/p x(n):');

m=length(x);

h=input('Enter i/p h(n): ');

n=length(h);

x=[x,zeros(1,n)];

subplot(2,2,1), stem(x);

title('i/p sequence x(n) is :');

xlabel('----->n');

ylabel('-----> x(n)'); grid;

h=[h,zeros(1,m)];

subplot(2,2,2), stem(h);

title('i/p sequence h(n) is :');

xlabel('----->n');

ylabel('-----> h(n)'); grid;

disp('Linear convolution of x(n) & h(n) is y(n) :');

y=zeros(1,m+n-1);

for i=1:m+n-1

y(i)=0;

for j=1:m+n-1

if(j<i+1)

y(i)=y(i)+x(j)\*h(i-j+1);

end

end

end

y

subplot(2,2,[3,4]), stem(y);

title('Linear convolution of x(n) & h(n) is :');

xlabel('----->n');

ylabel('-----> y(n)');

grid;

Output:

linear convolution program

enter i/p x(n):[1 2 3 4 1]

Enter i/p h(n): [2 1 0 -2]

Linear convolution of x(n) & h(n) is y(n) :

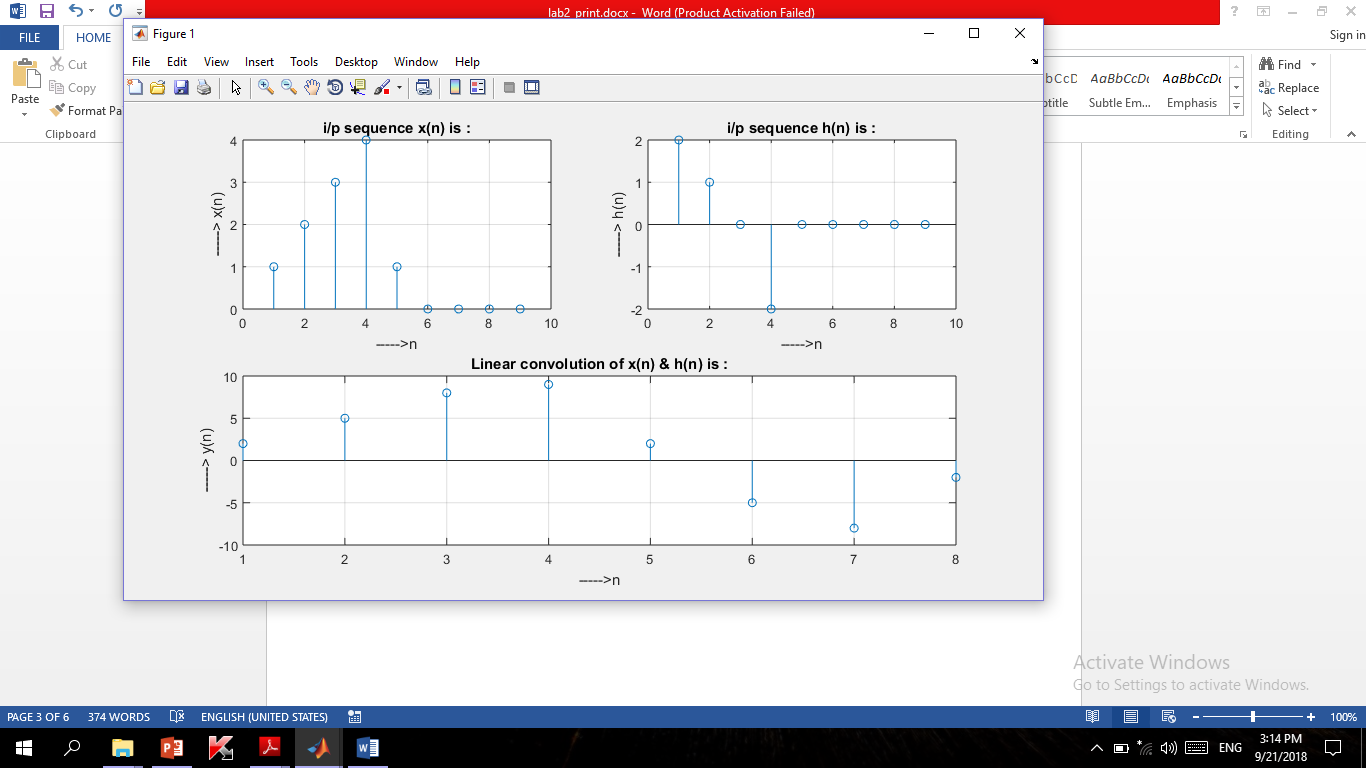
y = 2 5 8 9 2 -5 -8 -2

Figure 2.6: convolution between two vectors

Class works:

Task: To represent complex exponential as a function of real & imaginary part.

clc;

clear all;

close all;

N=100;

dw=pi/N;

w=0:dw:2\*pi;

x=exp(-j\*w); %%complex exponential

subplot(2,2,1)

stem(w,real(x));

title('Real part')

xlabel('Index(n)')

subplot(2,2,2)

stem(w,imag(x));

title('Imag. part')

xlabel('Index(n)')

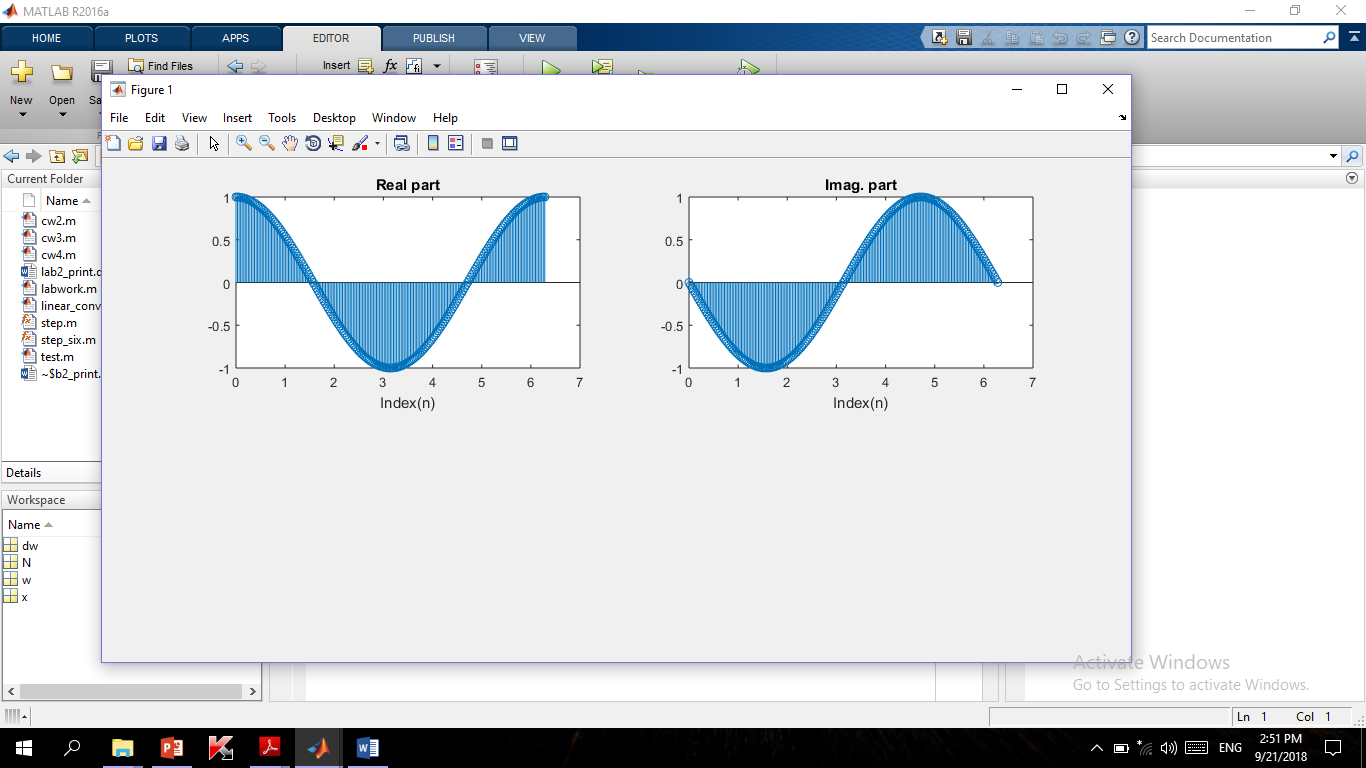


Figure 2.4: To represent complex exponential as a function of real & imaginary part.

Class works:

Task: To represent complex exponential as a function of real & imaginary part.

clc;

clear all;

close all;

N=1024;

a=3;

b=2;

c=1;

dw=2\*pi/N;

w=-pi:dw:pi-dw;

s=exp(j\*w);

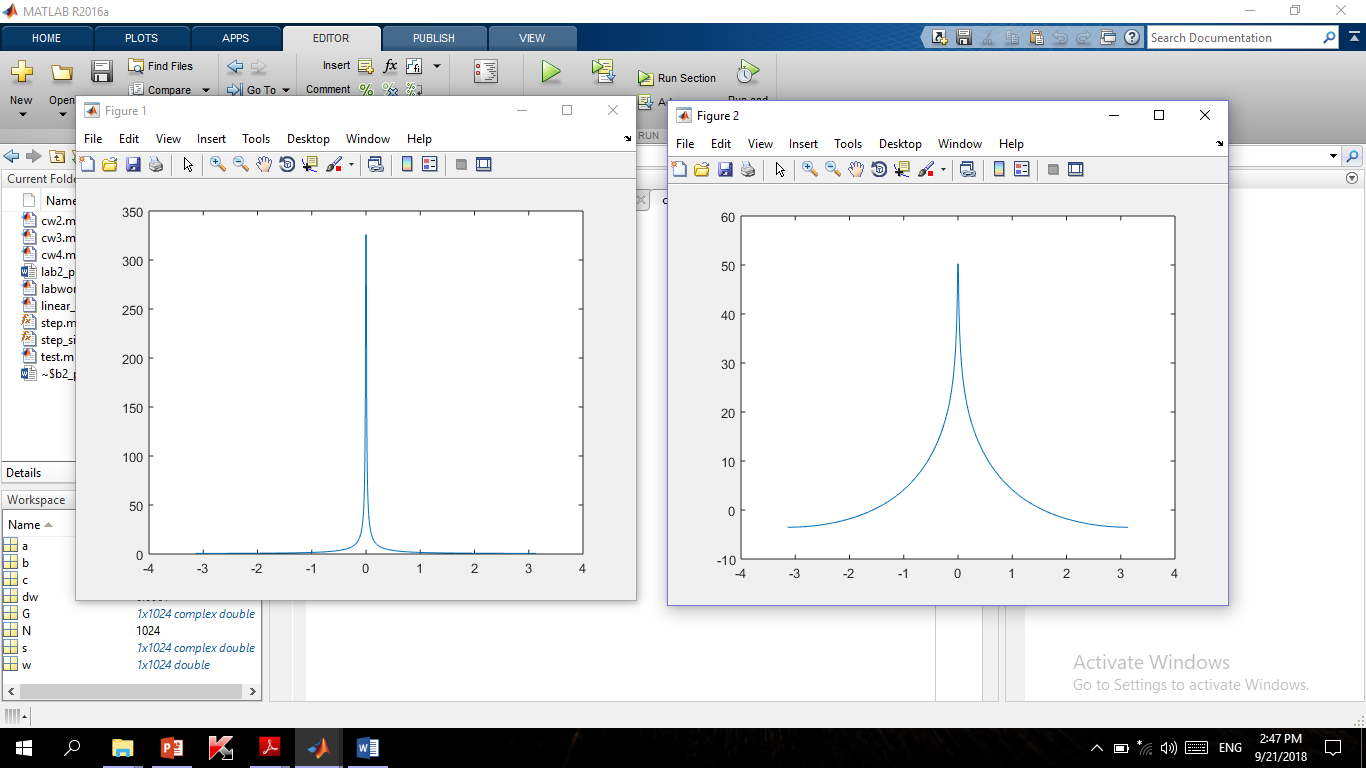
G=(s-a)./((s-b).\*(s-c));

figure;

plot(w,abs(G));

figure;

plot(w,20\*log10(abs(G)));



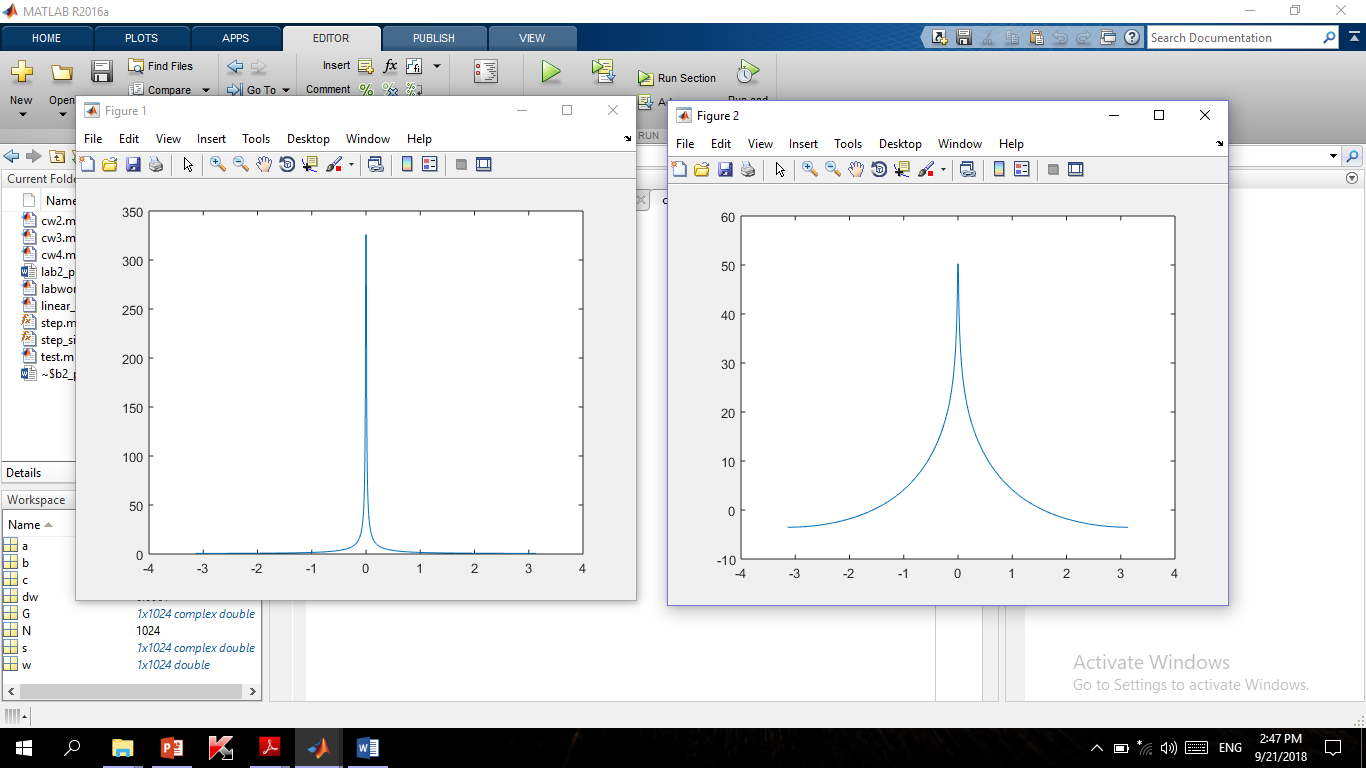


Figure 2.3: representing complex exponential function in logarithmic scale.

Figure 2.2: representing complex exponential function .

Class works:

Task : To generate discrete sine & cosine signals with given sampling frequency.

fs=input('Enter sampling frequency : ');

f=input('Enter signal frequency : ');

a=input('Enter amplitude : ');

%%generation of sine signal

t=0:(1/fs):1;

y=a\*sin(2\*pi\*f\*t);

subplot(2,1,1);

stem(t,y);

xlabel('Time---------');

ylabel('Amplitude-----------');

title('Sine Wave');

%%generation of cosine signal

t=0:(1/fs):1;

y=a\*cos(2\*pi\*f\*t);

subplot(2,1,2);

stem(t,y);

xlabel('Time---------');

ylabel('Amplitude-----------');

title('Cosine Wave');

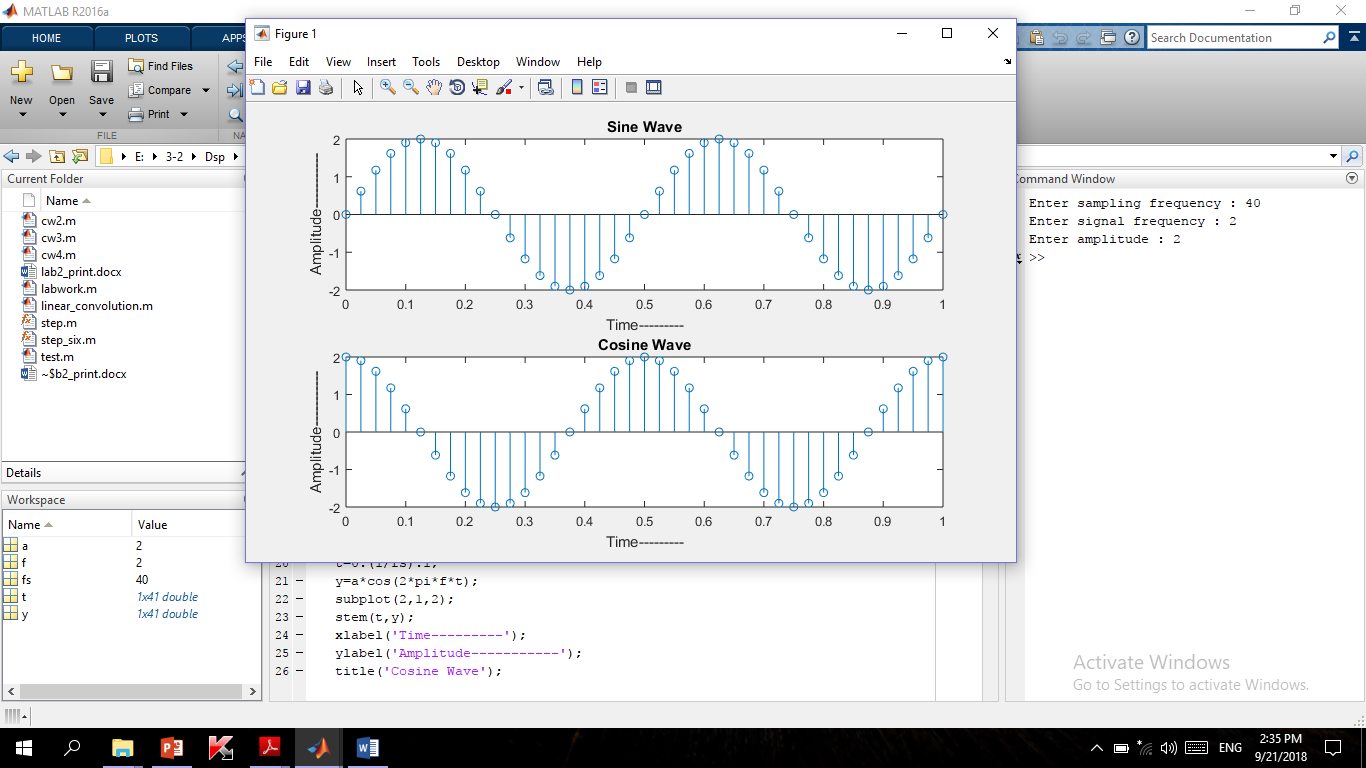


Figure 2.1: To generate discrete sine & cosine signals with given sampling frequency

Fs= 40, f=2 , a= 2 .