Lab-5

Discrete time FT and LTI systems

Name:Manogna

Roll No:2022102021

Name:Srivarshitha

Roll No:2022102030

## 1.Discrete time Fourier transform(DTFT):

For finding the DTFT of any discrete time signal x[n] is given as

$$X(e^{j\omega}) = \sum_{-\infty}^{\infty} x[n]e^{-j\omega n}$$

Here the function

X = DT_Fourier(x,N0,w)

CODE:

```
function X = DT_Fourier(x, N0, w)
    N = length(x);
    X = zeros(size(w));
    for i = 1:length(w)
        X(i) = sum(x .* exp(-1j * w(i) * (0:N-1 - N0)));
    end
end
```

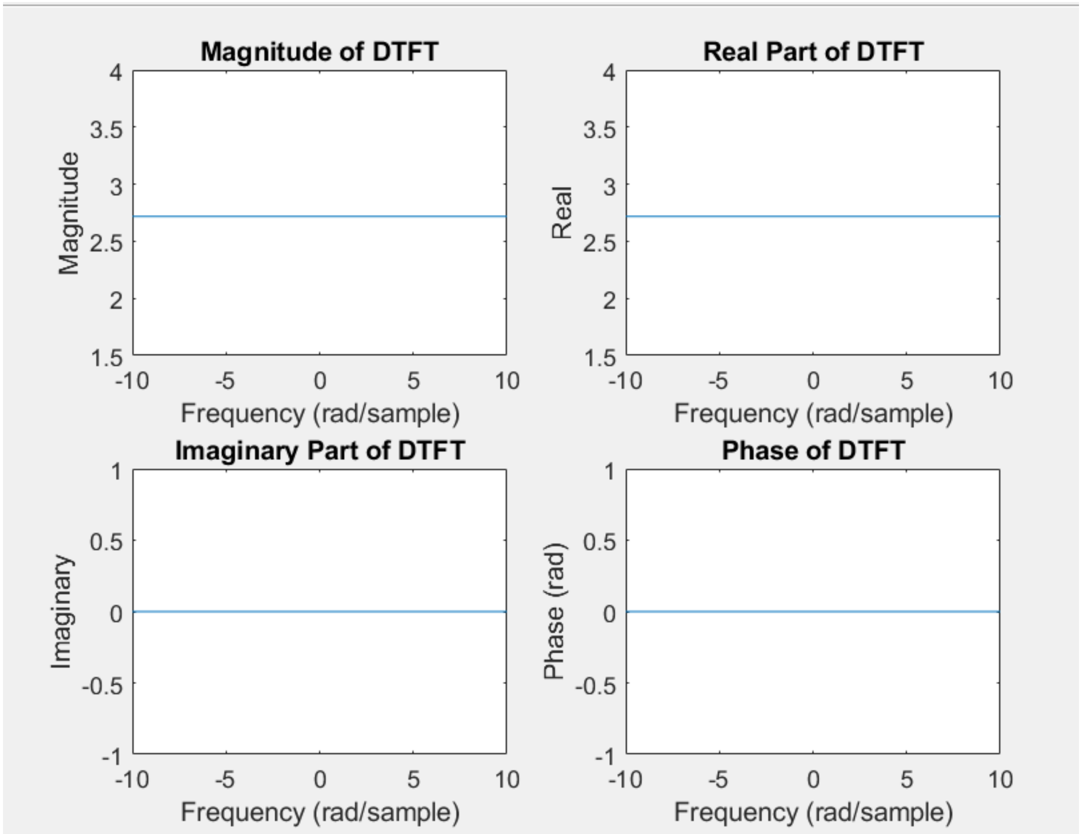
Here

• x, a discrete-time signal of finite duration (assume that the signal is zero elsewhere) • $N0$, location of the sample x[0] in the given input signal x, note that $N0$ is a positive integer between 1 and length(x) • $\omega$, a vector of frequencies at which to compute the DTFT (though frequency is a continuous variable in DTFT we can evaluate it at only finite set of points)

b) Set $\omega$ = −10: 0.01: 10

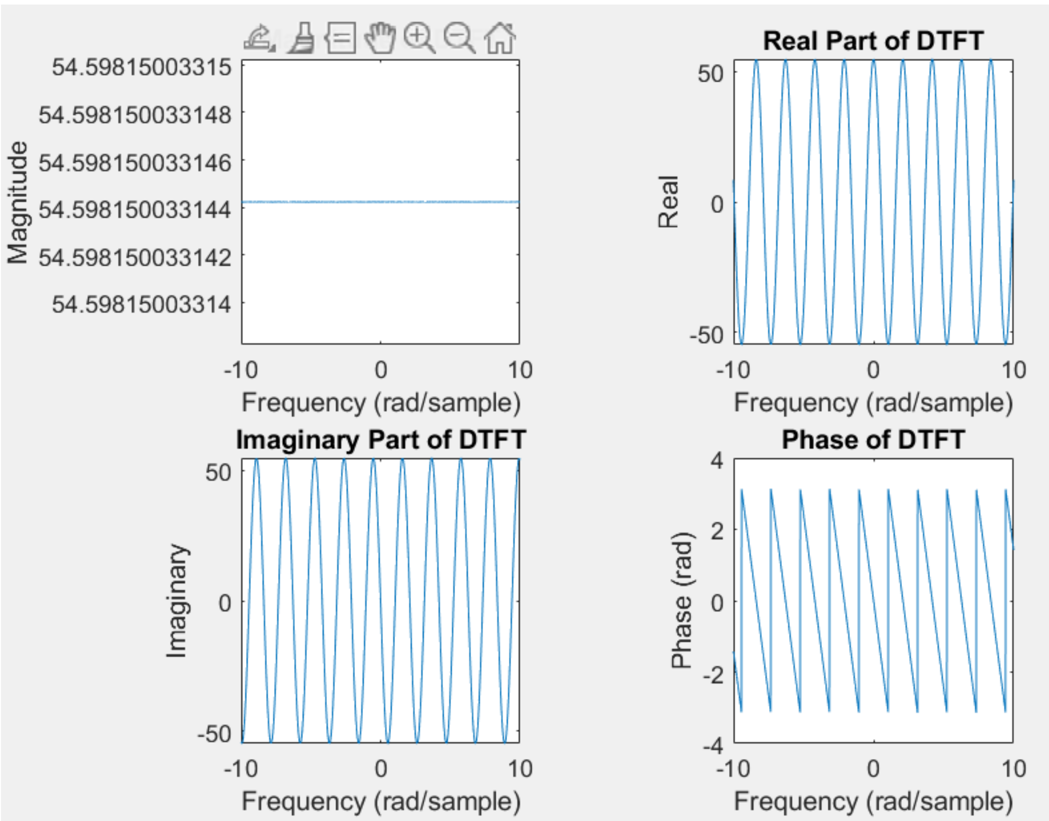
1) unit impulse $\delta[n]$

Plot:


Magnitude of DTFT
Magnitude
4
3.5
3
2.5
2
1.5
-10
-5
0
5
10
Frequency (rad/sample)
Real Part of DTFT
Real
Imaginary Part of DTFT
Imaginary
1
0.5
0
-0.5
-1
Phase of DTFT
Phase (rad)

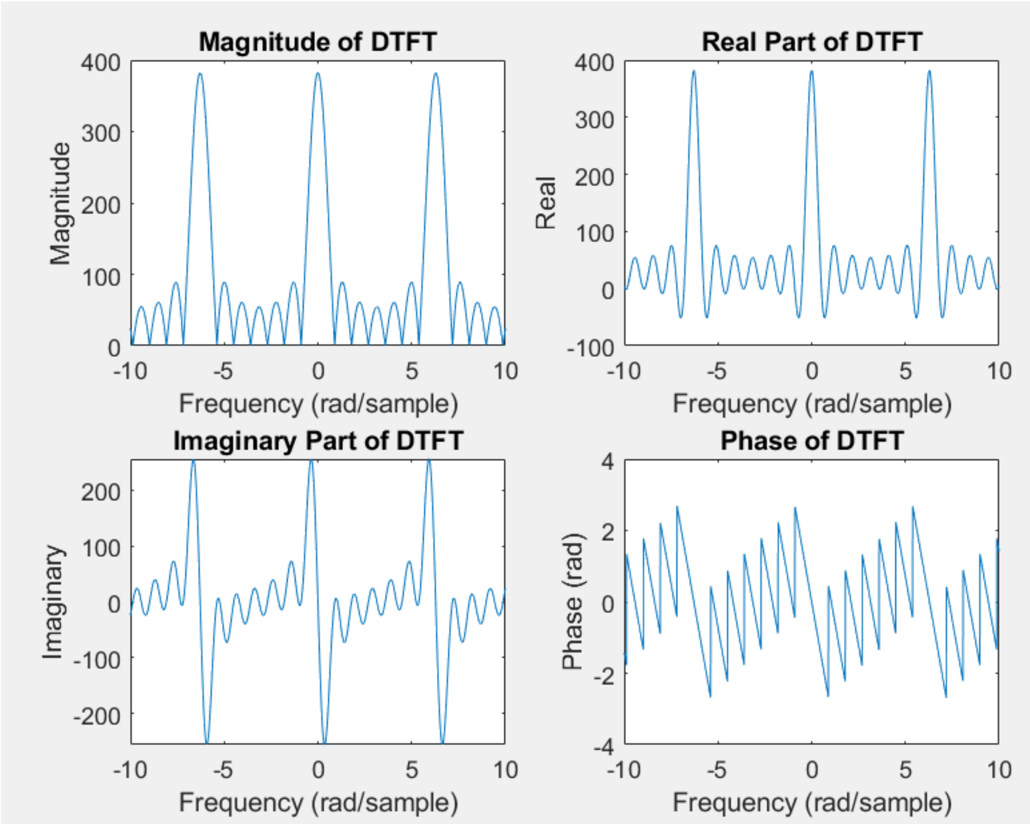
b2)

Plot:


54.59815003315
54.598150033148
54.598150033146
54.598150033144
54.598150033142
54.59815003314
Magnitude
-10
0
10
Frequency (rad/sample)
Real Part of DTFT
50
0
-50
Real
Imaginary Part of DTFT
Imaginary
Phase of DTFT
4
2
0
-2
-4
Phase (rad)

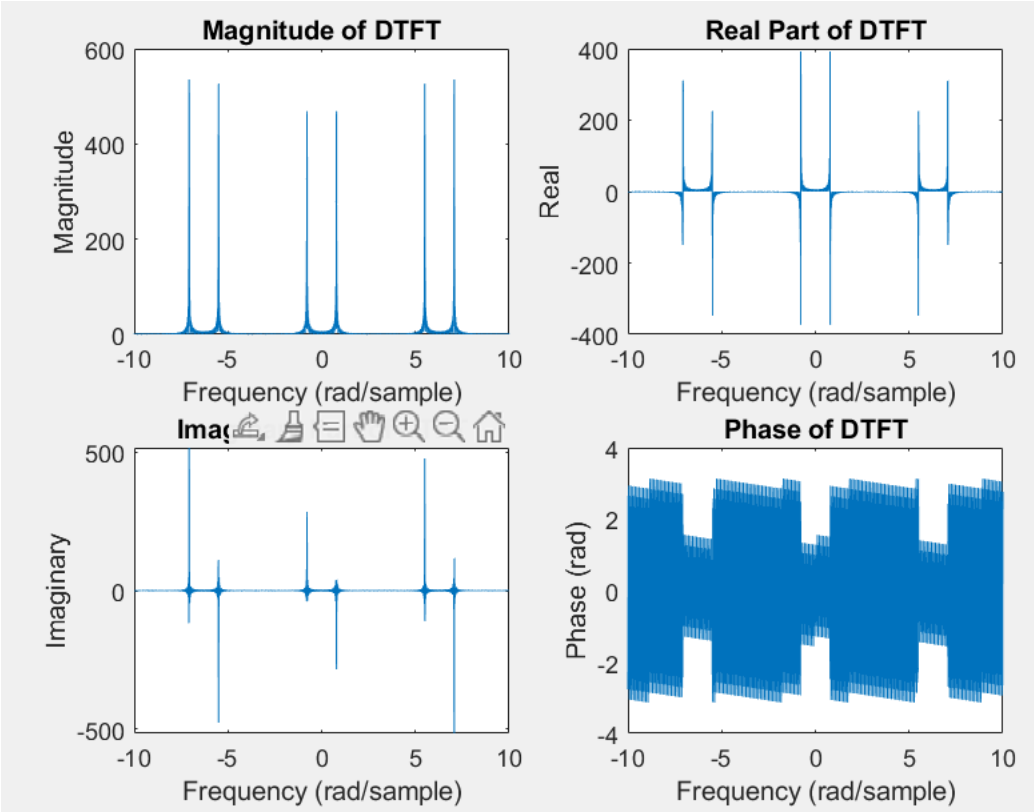
b3)

plot:

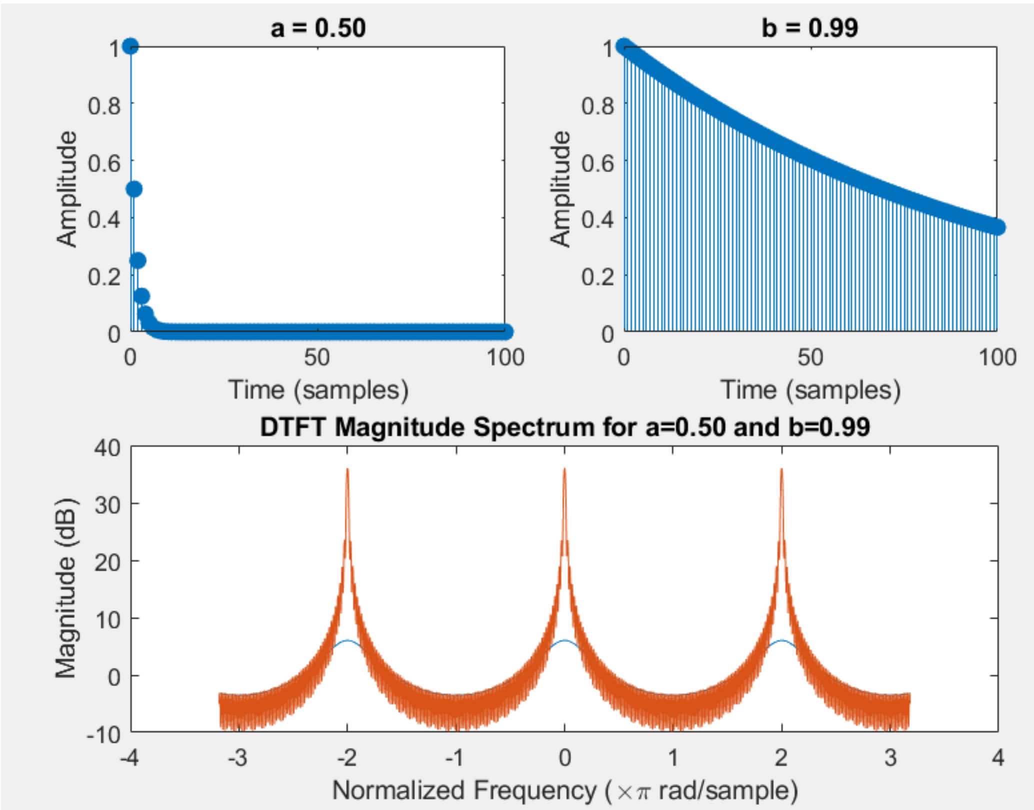


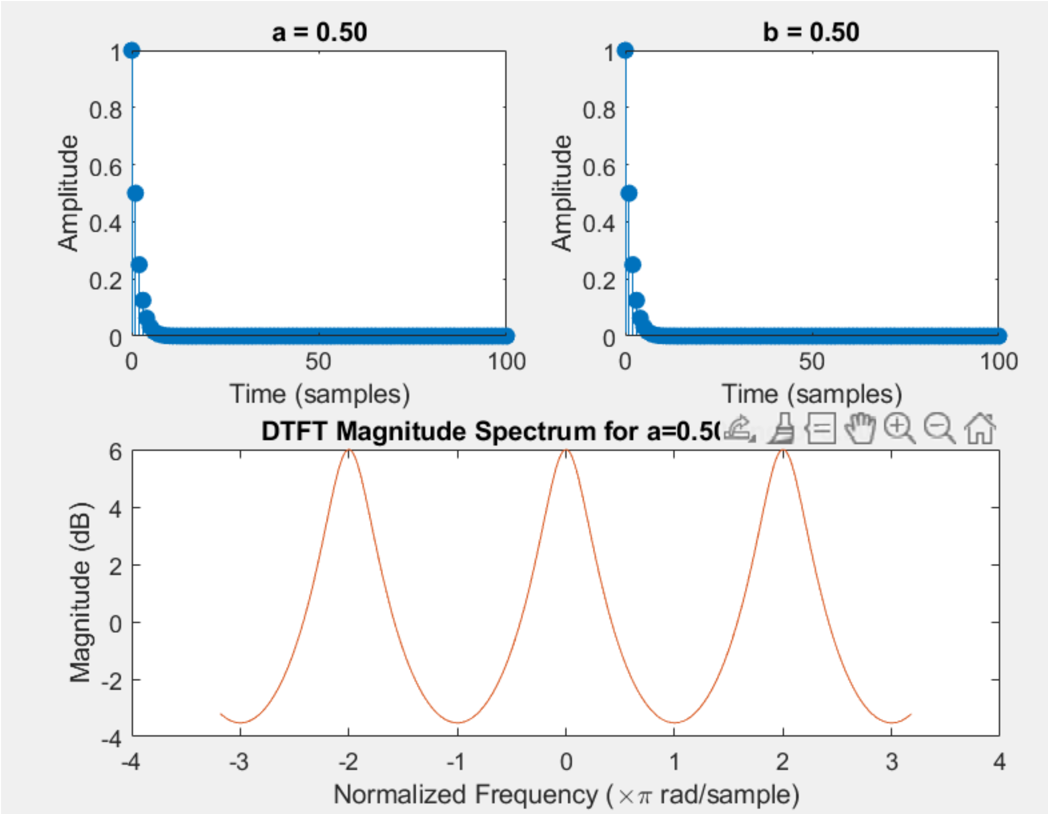

b4)

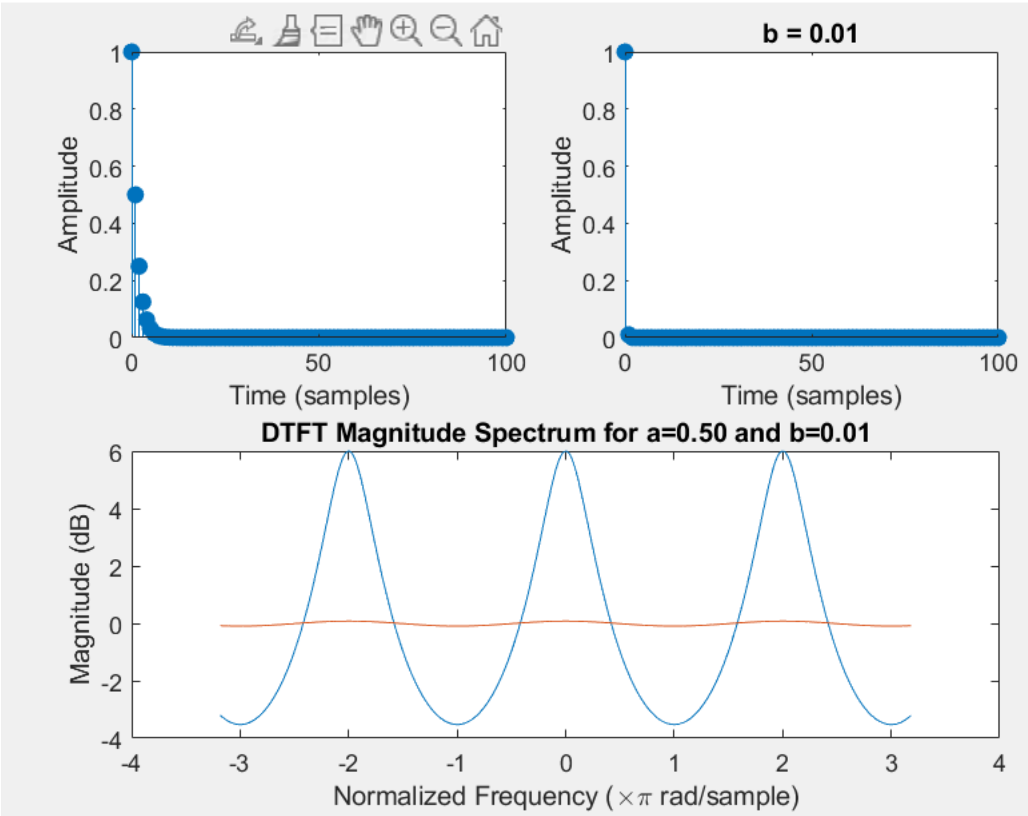
plot:




c) $\omega = -10: 0.01: 10$






2.Discrete -time filters:

Here we are considering the x[n] as input and y[n] as output.

And the relation between the input and the output is

$$y[n] = \frac{1}{M}\sum_{m=0}^{M-1} x[n-m]$$

a)here the impulse response of this system is h[n] they the input x[n] should be the $\boldsymbol{\delta}$[n] i.e

x[n]= $\boldsymbol{\delta}$[n]then the h[n]= $\frac{1}{M}\sum_{m=0}^{M-1} x[n-m]$ and the x[n]= $\boldsymbol{\delta}$[n] and the x[n-m]=$\boldsymbol{\delta}$[n-m]

The impulse response of a moving average filter with order M is a sequence that has a value of 1/M for the first M samples and is zero for all other samples. It can be represented as follows:

**h[n] = (1/M) for n = 0, 1, 2, ..., M-1

h[n] = 0 otherwise

b) y[n]=conv(x[n],v[n])

c)here the input signal becomes noisy when we add the s[n] and v[n] here the s[n] is given as

$s[n]=5\sin(w_o n)$

and the v[n] is random signal.

The input signal x[n]=s[n]+v[n] and we are considering the $w_o=\pi/200$

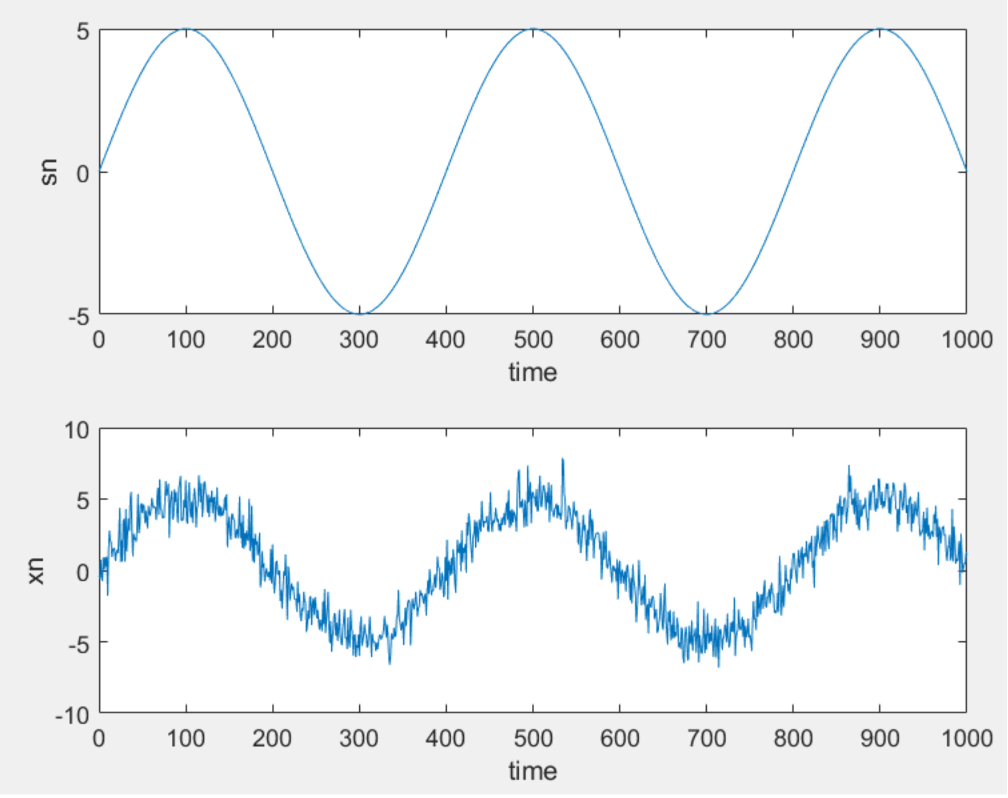
And taking the n values from 0 to 1000

CODE:

```
wo=pi/200;
n=0:1000;
sn=5*sin(wo*n);
vn=randn(1,1001);
xn=sn+vn;
figure
subplot(2,1,1)
plot(n,sn);
xlabel('time')
ylabel('sn')

subplot(2,1,2)
plot(n,xn)
xlabel('time')
ylabel('xn')
```

The plots of x[n] and the s[n] verses n are:

Plot:




d) Filter the noisy signal **`x[n]`** using the moving average filter with different values of M and plot the results. Here's how you can do it:

here we are changing the M values

M=5,21,51

then

$$y[n] = \frac{1}{M}\sum_{m=0}^{M-1} x[n-m]$$

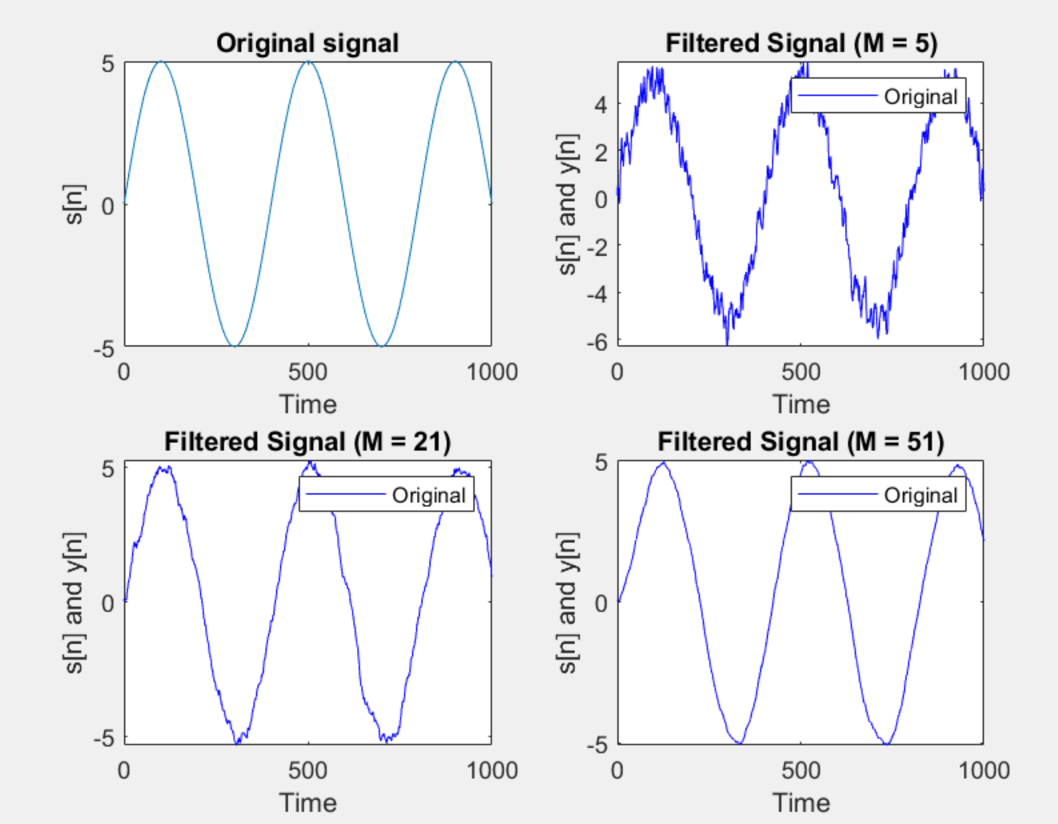
For finding y[n] we convolute the x[n] and the h[n]

ie y[n]=conv(x[n],v[n])

CODE:

```
M = [5, 21, 51];
n = 0:1000;
wn = pi / 200;
sn = 5 * sin(wn * n);
vn = randn(1, 1001);
xn = sn + vn;

figure;

for i = 1:length(M)
    hn = ones(1, M(i)) / M(i);
    yn = conv(xn, hn, 'full');

    subplot(2,2,1);
    plot(n, sn);
    xlabel('Time');
    ylabel('s[n]');
    title('Original signal');

    subplot(2,2,i+1);
    plot(n, yn(1:length(n)), 'b');
    xlabel('Time');
    ylabel('s[n] and y[n]');
    title(['Filtered Signal (M = ' num2str(M(i)) ')']);
    legend('Original', 'Filtered');
end

```

Plot of y[n] for all the values of M:




e)*Observation:

Here as you change the value of M:

*smaller M values result in less smoothing and better presentation of high frequency components in the signal.

*Larger M values result in more smoothing and better noise reduction but can blur or distort the signal's sharp features.

*There's a trade-off between noise reduction and signal fidelity.

Higher M values reduce noise but may introduce distortion.

g)here the relation between the input and the output signal is given by

y[n]=x[n]-x[n-1]

For the digital differentiator filter , the impulse response h[n] is given by:

h[n]=1 for n=0

h[n]=-1 for n=1

h[n]=0 otherwise

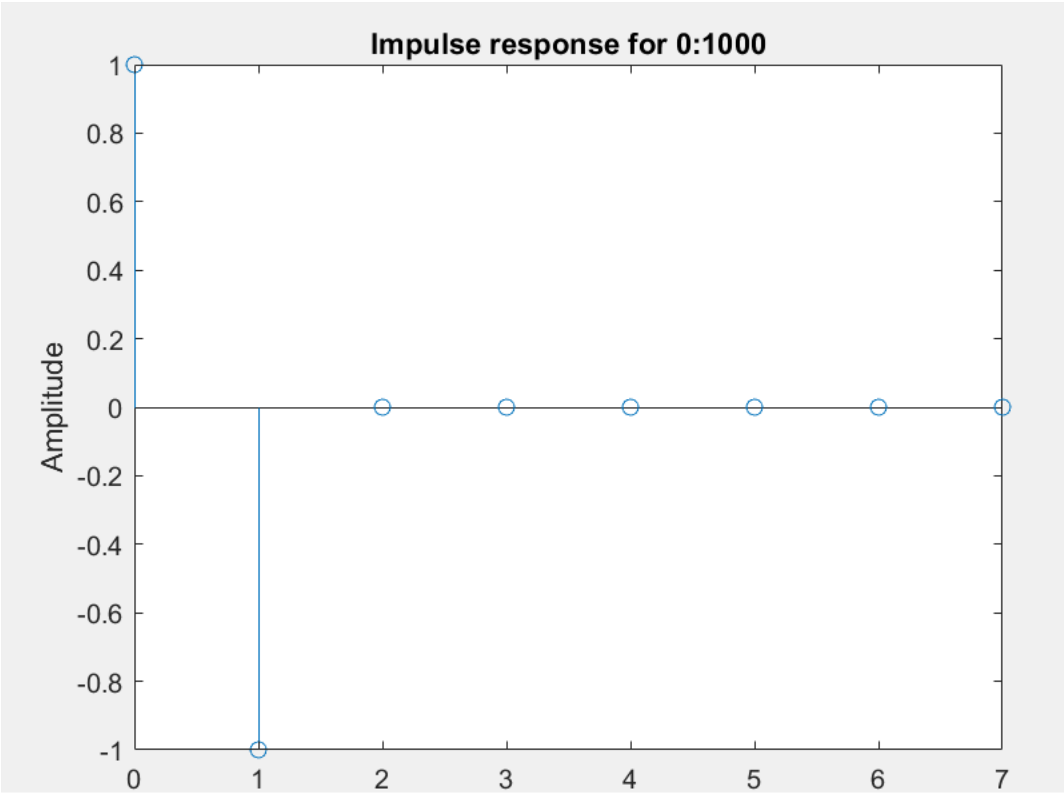
part.a)

Code:

Here n=0:1000;

```
sum = zeros(n);
hn = [1, -1, zeros(1,6)];

figure;
stem(0:7,hn);
title(sprintf("Impulse response for %d:%d",n(1),n(end)));
xlabel("Time");
ylabel("Amplitude");
```

Plot:

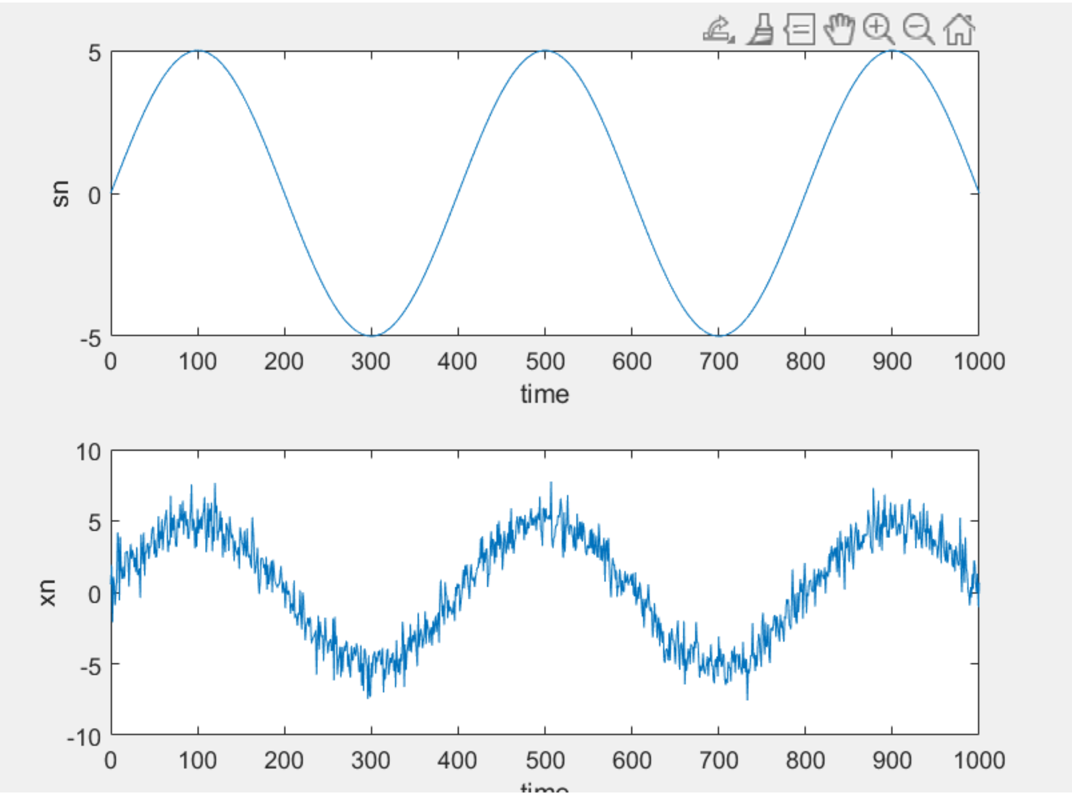



Part.c:

Code:

```
wo=pi/200;
n=0:1000;
sn=5*sin(wo*n);
vn=randn(1,1001);
xn=sn+vn;
figure
subplot(2,1,1)
plot(n,sn);
xlabel('time')
ylabel('sn')

subplot(2,1,2)
plot(n,xn)
xlabel('time')
ylabel('xn')
```

Plot:

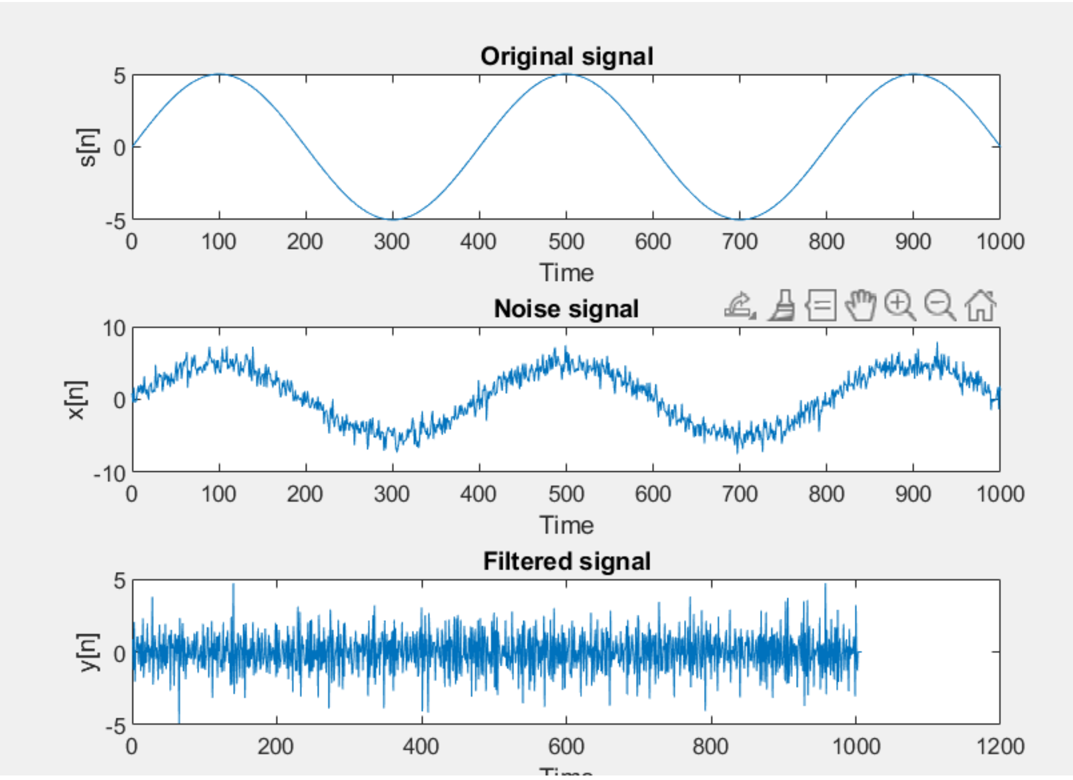



Part.d)

CODE:

```
n = 0:1000;
wn = pi / 200;
sn = 5 * sin(wn * n);
vn = randn(1, 1001);
xn = sn + vn;

figure;
hn = [1, -1, zeros(1,6)];

    subplot(3,1,1);
    plot(n, sn);
    xlabel('Time');
    ylabel('s[n]');
    title('Original signal');


    subplot(3,1,2);
    plot(n, xn);
    xlabel('Time');
    ylabel('x[n]');
    title('Noise signal');


    subplot(3,1,3);
    plot(yn);
    xlabel('Time');
    ylabel('y[n]');
    title('Filtered signal');


```

Plot:




Part.f

CODE:

```
N = 1000;
omega = pi / 200;
n = 0:N;
s = 5 * sin(omega * n);
 w = -10: 0.01: 10;
v = randn(size(n));
x = s + v;


t = 10;
filter = [1, -1, zeros(1, t-2)];

y = conv(x, filter, 'full');

X_noisy = DT_Fourier(x, 1, w);
Y_filtered = DT_Fourier(y, 1, w);

figure(1)
subplot(2, 2, 1);
plot(w, abs(X_noisy));
title('Magnitude Spectrum of Noisy Signal');
subplot(2, 2, 2);
plot(w, angle(X_noisy));
title('Phase Spectrum');
subplot(2, 2, 3);
plot(w, real(X_noisy));
title('Real Part Spectrum');
subplot(2, 2, 4);
plot(w, imag(X_noisy));
title('Imaginary Part Spectrum');

figure(2)
subplot(2, 2, 1);
plot(w, abs(Y_filtered));
title('Magnitude Spectrum of Filtered Signal');
subplot(2, 2, 2);
plot(w, angle(Y_filtered));
title('Phase Spectrum');
subplot(2, 2, 3);
plot(w, real(Y_filtered));
title('Real Part Spectrum');
subplot(2, 2, 4);
plot(w, imag(Y_filtered));
```

h)

Frequency Selectivity:

*The moving average filter is a low-pass filter that attenuates high-frequency components while preserving low-frequency components . Its frequency selectivity is determined by the filter order M.

Smaller values of M allow higher frequencies to pass through , while larger values of M result in a more significant attenuation of high-frequency components.

EFFECT:

It is effective for smoothing signals and reducing noise but may blur or distort signals with rapid changes.

*Digital Differentiator Filter:

The digital differentiator filter enhances high-frequency components and attenuates low- frequency components . It acts as a high-pass filter.It approximates the first derivative of the signal.

EFFECT:

It can emphasize edges and rapid changes in the signal, making it useful for detecting abrupt transitions or fine details.

** The nature of the two filters is as follows:

The moving average filter is low-pass and attenuates high frequencies , while the digital differentiator filter is high-pass and enhances high -frequency information. The choice of filter depends on the specific signal processing requirements and the desired trade-off between noise reduction and feature preservation.

3)

Here the inverse DTFT is calculated by using the expression

$$x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(e^{j\omega})\, e^{j\omega n}\, d\omega.$$

Here the $X(e^{jw})$ is given as

$$X(e^{j\omega}) = \begin{cases} 1, & |\omega| \le \omega_c \\ 0, & \omega_c < |\omega| < \pi \end{cases}$$

a)

here

wc=pi/16

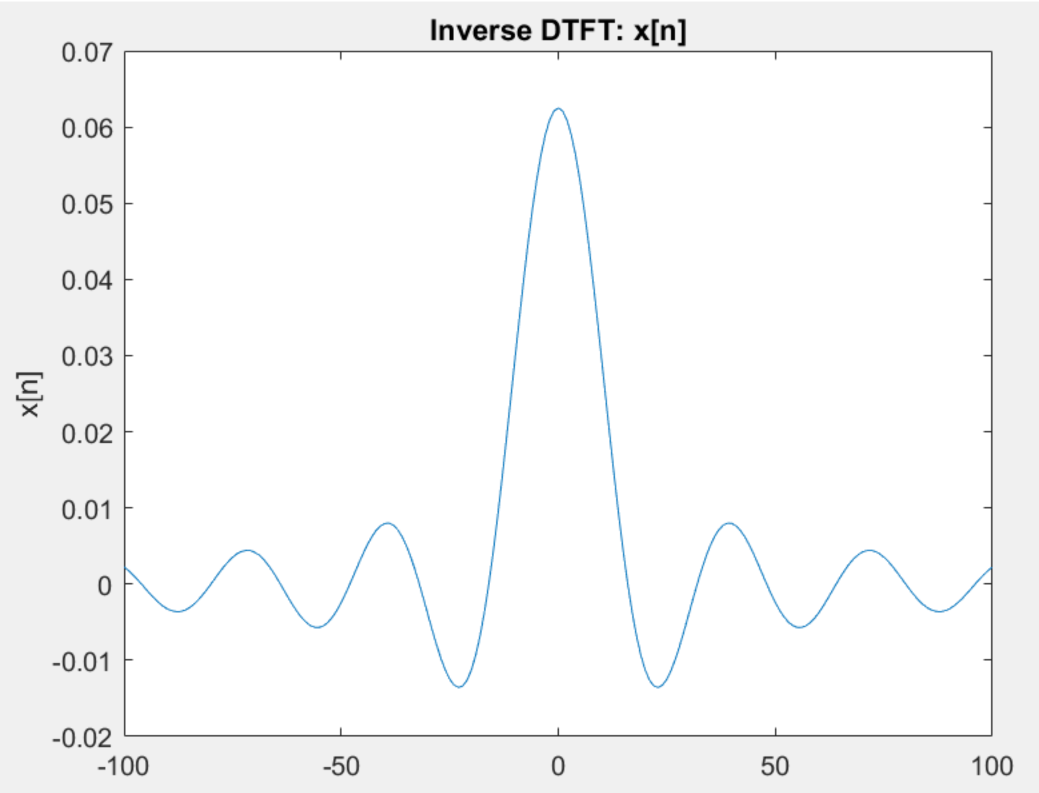
and n=-100:100

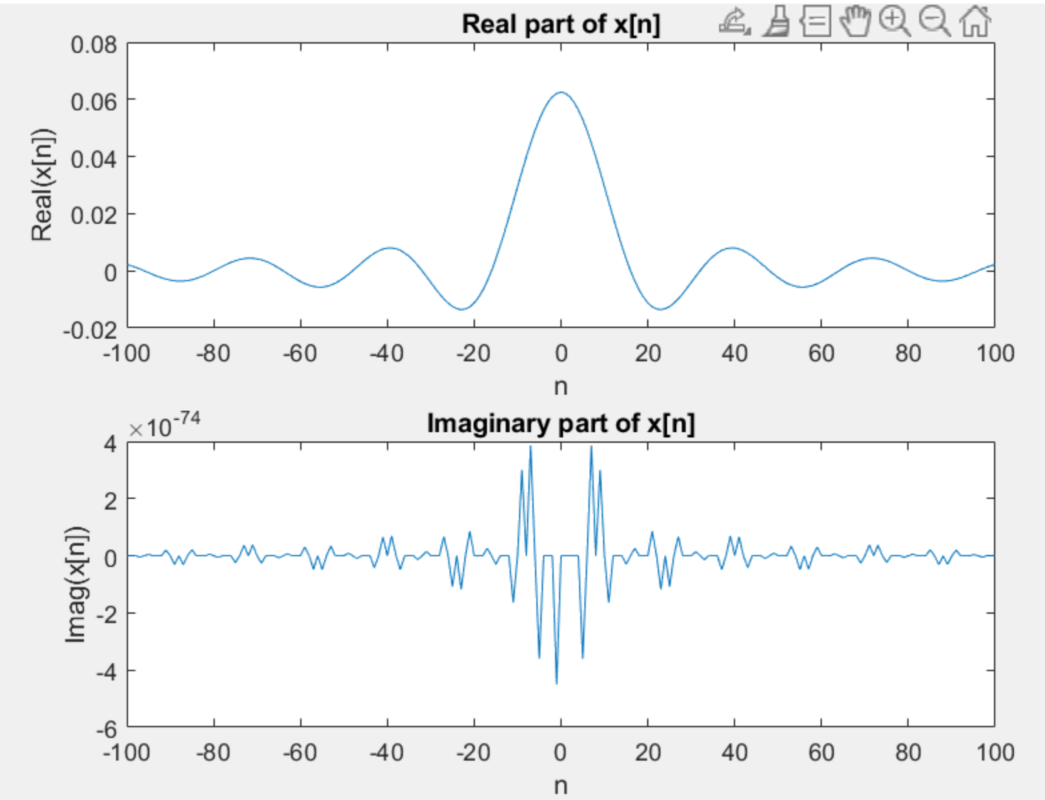
we are plotting the real and complex valued of x[n] in the interval [-pi,pi]

CODE:

```
wc = pi/16;
n = -100:100;
x = zeros(1, length(n));
syms w;
X = piecewise(abs(w) <= wc, 1, wc < abs(w) & abs(w) < pi, 0);

for k = 1:length(n)
    t = exp(1j * w * n(k));
    x(k) = (1/(2*pi)) * int(X * t , w, -pi, pi);
end

figure;
subplot(2,1,1);
plot(n, real(x));
title('Real part of x[n]');
xlabel('n');
ylabel('Real(x[n])');

subplot(2,1,2);
plot(n, imag(x));
title('Imaginary part of x[n]');
xlabel('n');
ylabel('Imag(x[n])');

figure;
plot(n, real(x));
title('Inverse DTFT: x[n]');
xlabel('n');
ylabel('x[n]');

```

Plot:






Here the x[n] is both real and imaginary those can be seen in the plots above clearly.

b)By changing the wc , you are altering the bandwidth of the rectangular frequency domain signal .

*As wc increases the rectangular frequency domain signal becomes narrower in frequency ,resulting in a boarder time domain signal and vice-versa

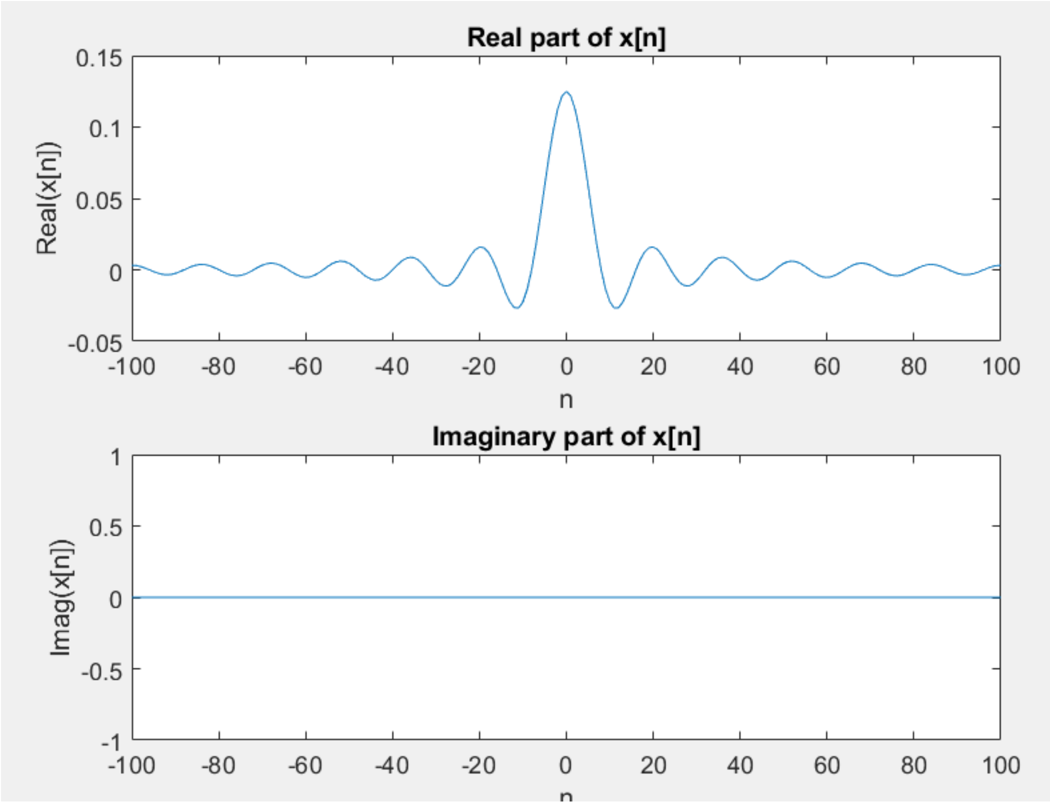
*when wc=pi the entire frequency domain signal becomes zero(no signal) , which means x[n] will also be zero for all values of n. This is because the signal bandwidth has become zero, resulting in no signal in the time domain.

CODE:

```matlab
wc = pi;
n = -100:100;
x = zeros(1, length(n));
syms w;
X = piecewise(abs(w) <= wc, 1, wc < abs(w) & abs(w) < pi, 0);

for k = 1:length(n)
    t = exp(1j * w * n(k));
    x(k) = (1/(2*pi)) * int(X * t , w, -pi, pi);
end

figure;
subplot(2,1,1);
plot(n, real(x));
title('Real part of x[n]');
xlabel('n');
ylabel('Real(x[n])');

subplot(2,1,2);
plot(n, imag(x));
title('Imaginary part of x[n]');
xlabel('n');
ylabel('Imag(x[n])');

figure;
plot(n, real(x));
title('Inverse DTFT: x[n]');
xlabel('n');
ylabel('x[n]');
```

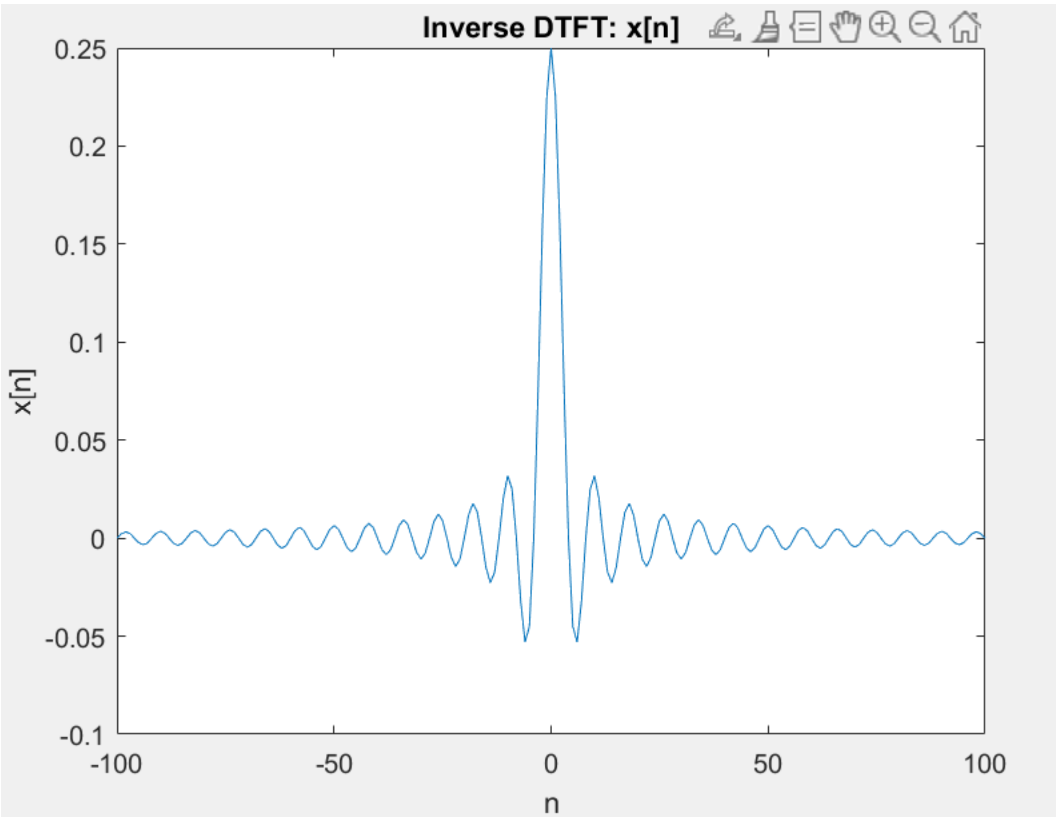
Plot:

a.for wc=pi/8

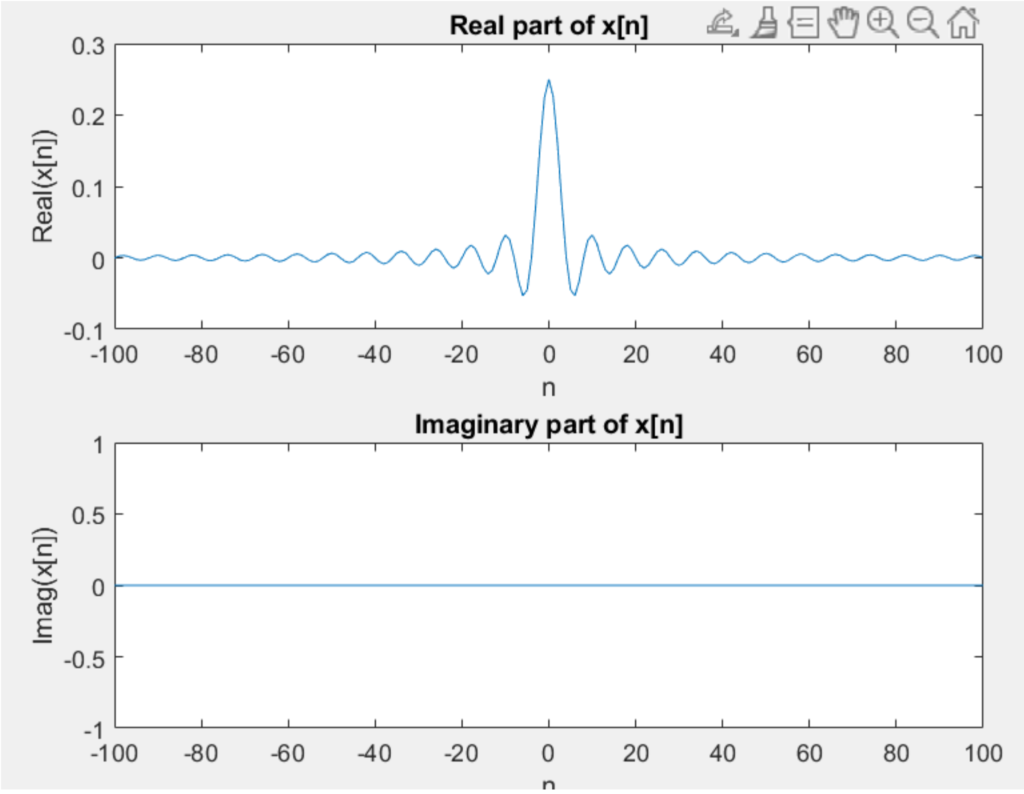


Real part of x[n]
Real(x[n])
0.15
0.1
0.05
0
-0.05
-100
-80
-60
-40
-20
0
20
40
60
80
100
n
Imaginary part of x[n]
Imag(x[n])
1
0.5
0
-0.5
-1
n

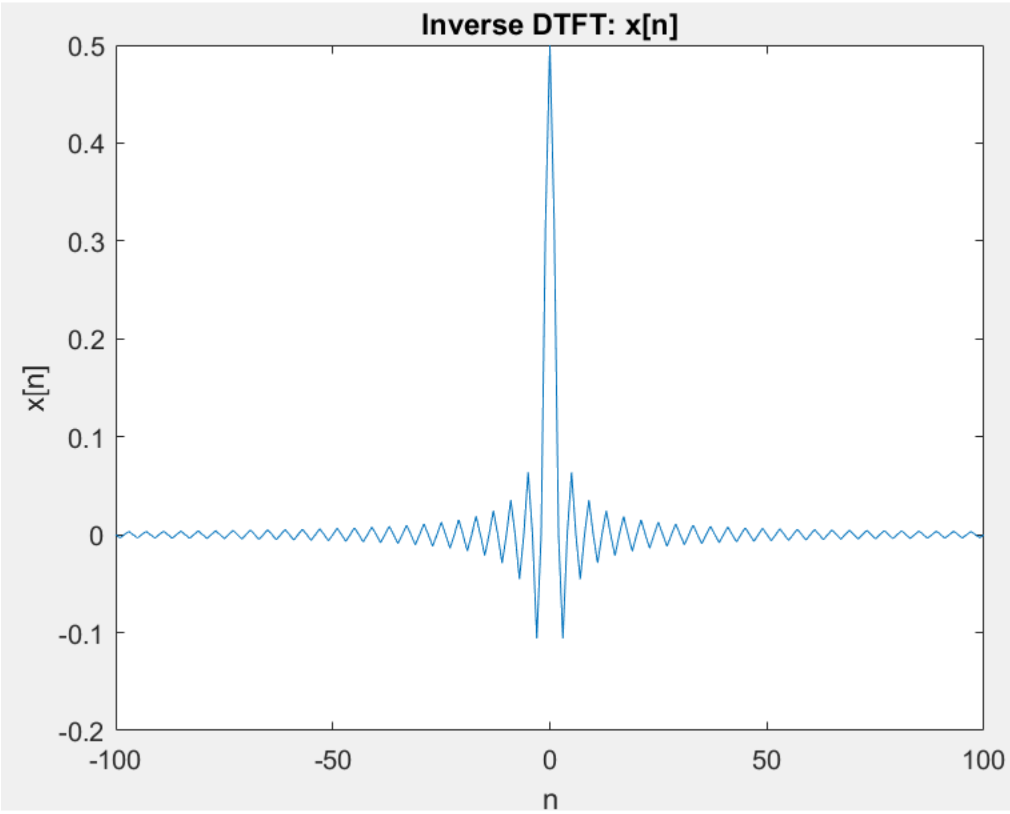

For wc=pi/4

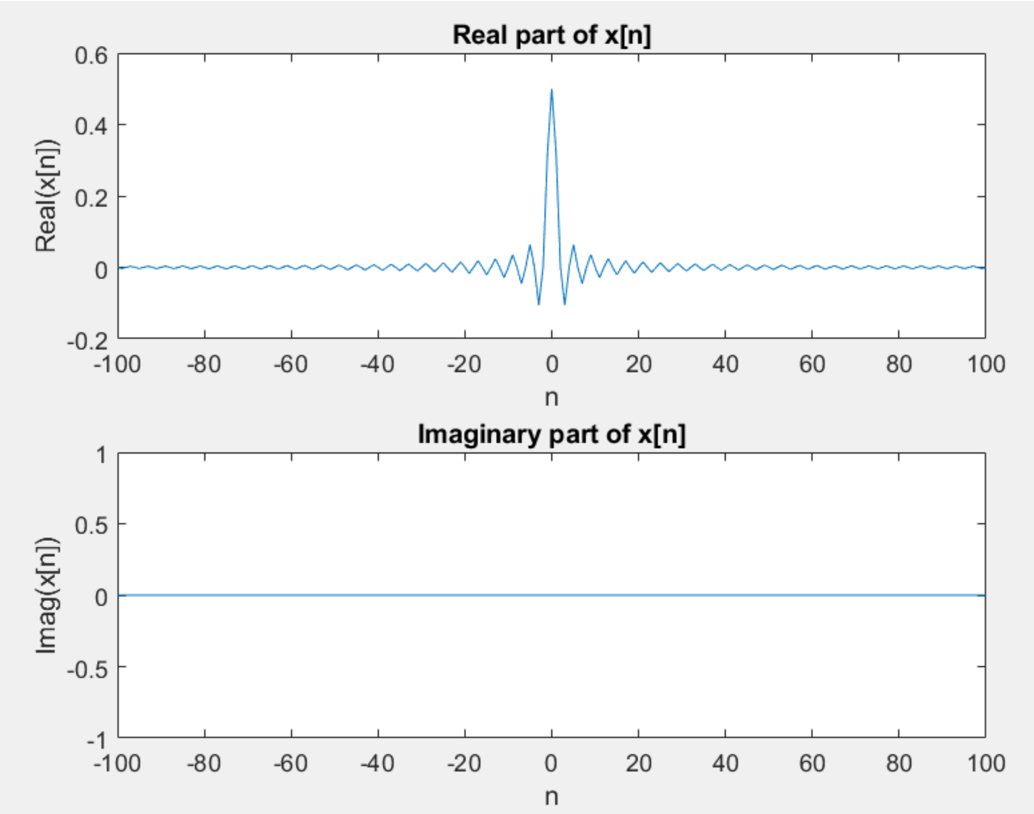


Inverse DTFT: x[n]
x[n]
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-100
-50
0
50
100
n

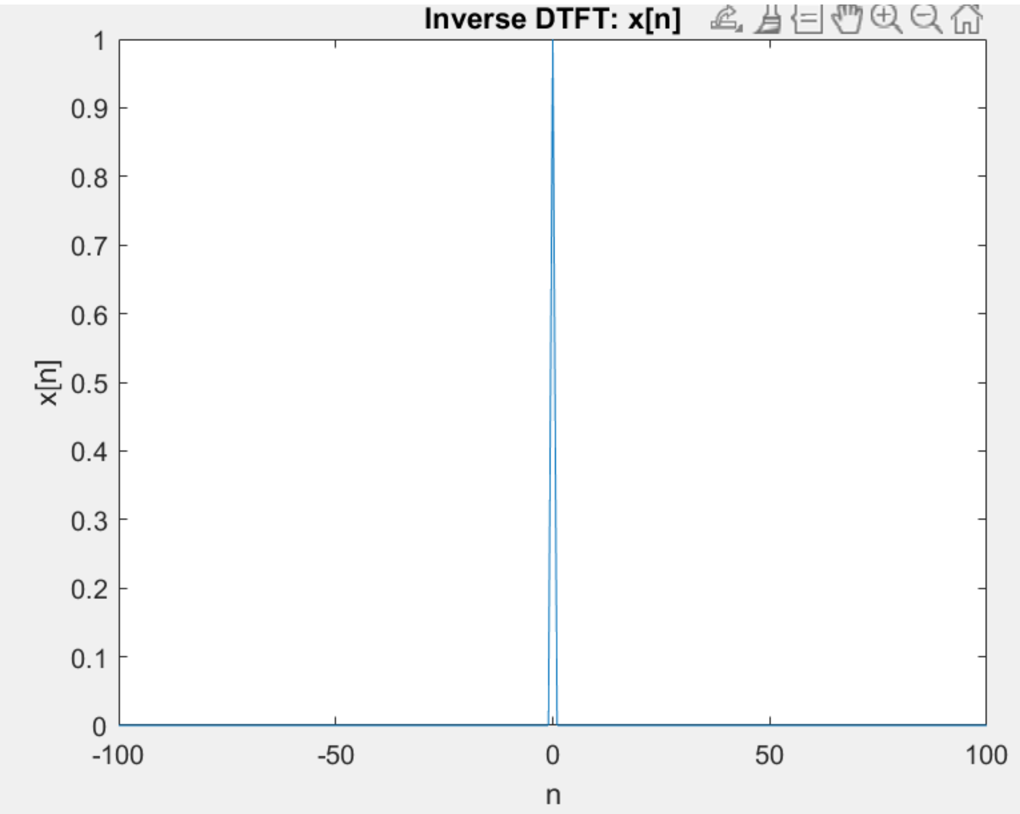

Real part of x[n]
Real(x[n])
n
Imaginary part of x[n]
Imag(x[n])
n

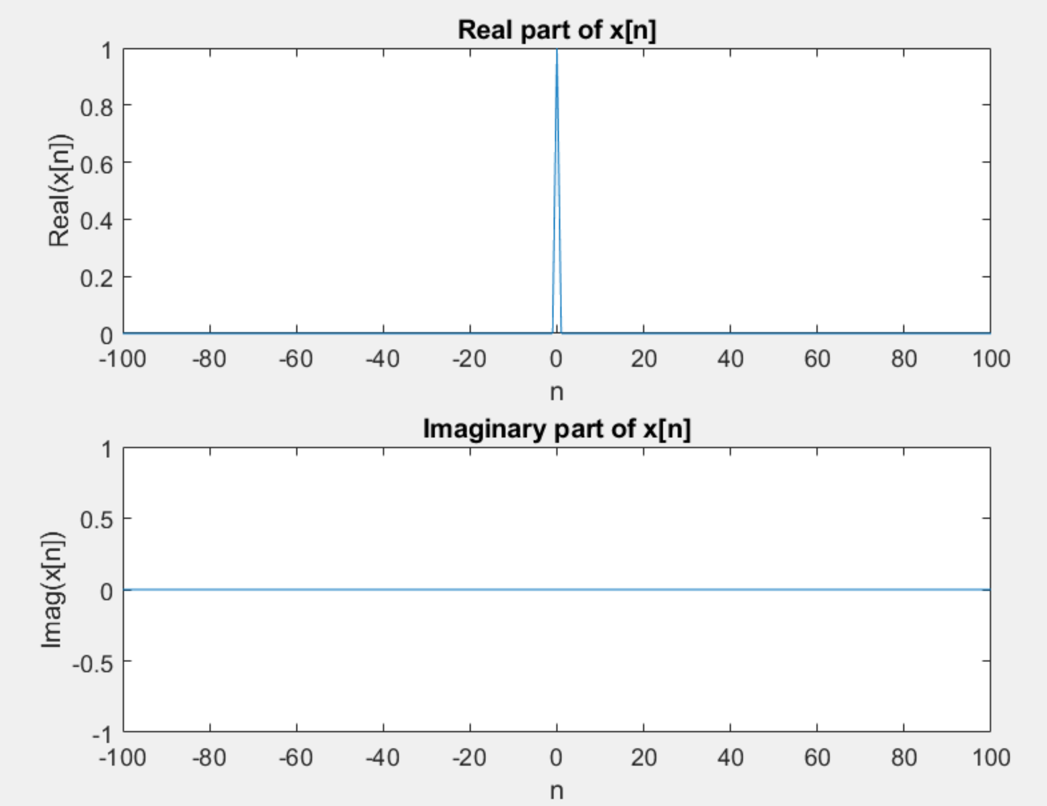
For wc=pi/2


Inverse DTFT: x[n]
x[n]
n


Real part of x[n]
Real(x[n])
0.6
0.4
0.2
0
-0.2
-100 -80 -60 -40 -20 0 20 40 60 80 100
n
Imaginary part of x[n]
Imag(x[n])
1
0.5
0
-0.5
-1
-100 -80 -60 -40 -20 0 20 40 60 80 100
n

For wc=pi


Inverse DTFT: x[n]
x[n]
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-100 -50 0 50 100
n




c)

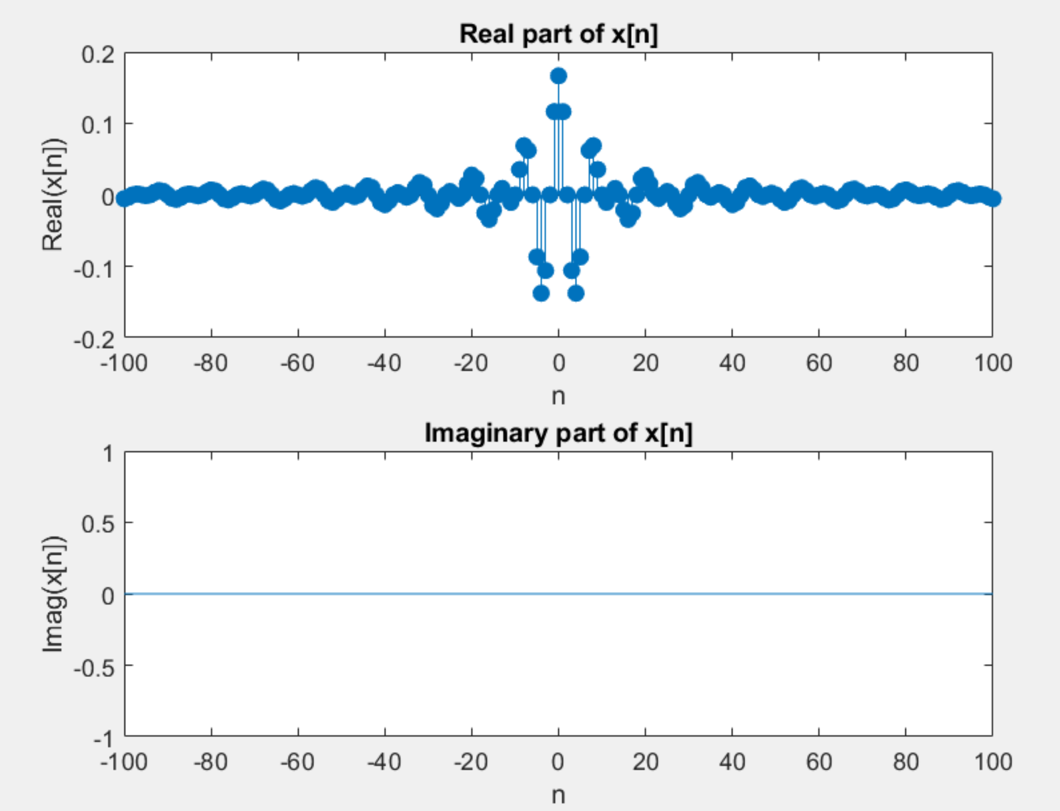
here the w1=pi/8

w2=pi/4

CODE:

```
n = -100:100;
x = zeros(1, length(n));
w1=pi/8;
w2=pi/4;

syms w;
assume(w,'real');
X(w)= piecewise(w1<=abs(w) & abs(w)<=w2,1,0);

for k = 1:length(n)
    t = exp(1j * w * n(k));
    x(k) = (1/(2*pi)) * int(X(w) * t , w, -pi, pi);
end

figure;
subplot(2,1,1);
plot(n, real(x));
title('Real part of x[n]');
xlabel('n');
ylabel('Real(x[n])');

subplot(2,1,2);
plot(n, imag(x));
title('Imaginary part of x[n]');
xlabel('n');
ylabel('Imag(x[n])');

figure;
plot(n, real(x));
title('Inverse DTFT: x[n]');
xlabel('n');
ylabel('x[n]');
```

Plot:

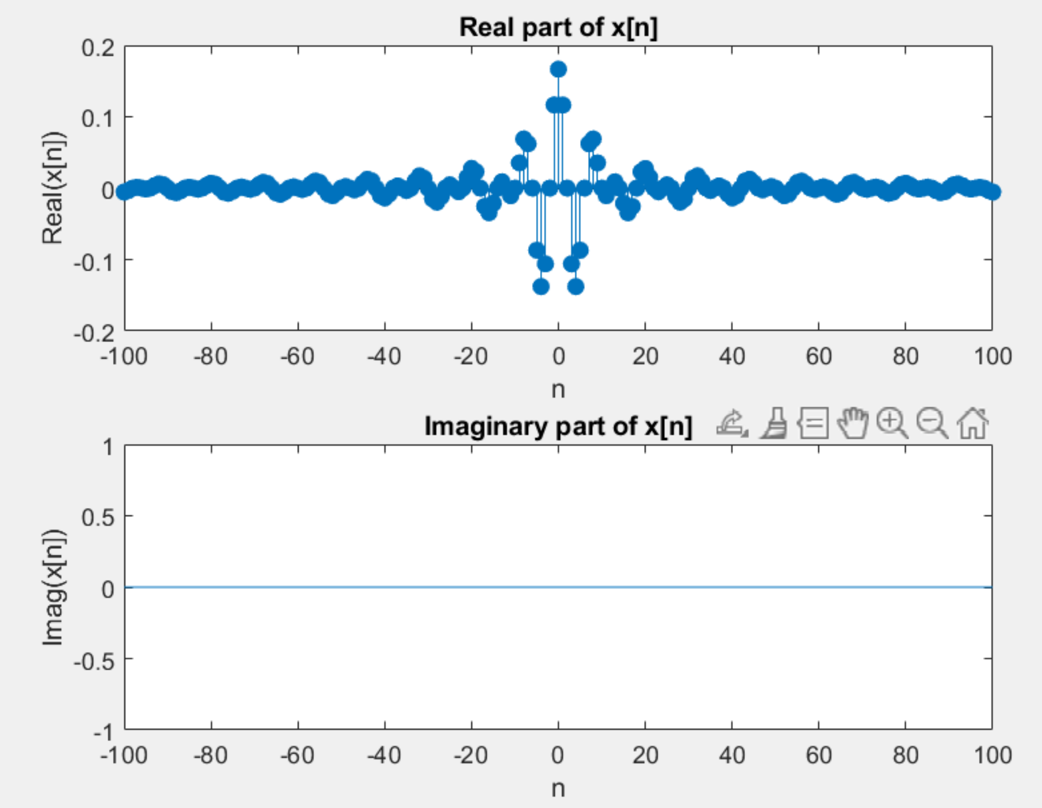



When the w1=pi/6 w2=pi/3

Code:

```
n = -100:100;
x = zeros(1, length(n));
w1=pi/6;
w2=pi/3;

syms w;
assume(w,'real');
X= piecewise(w1<=abs(w) & abs(w)<=w2,1,0);

for k = 1:length(n)
    t = exp(1j * w * n(k));
    x(k) = (1/(2*pi)) * int(X * t , w, -pi, pi);
end

figure;
subplot(2,1,1);
stem(n, real(x),"filled");
title('Real part of x[n]');
xlabel('n');
ylabel('Real(x[n])');

subplot(2,1,2);
plot(n, imag(x));
title('Imaginary part of x[n]');
xlabel('n');
ylabel('Imag(x[n])');

figure;
plot(n, real(x));
title('Inverse DTFT: x[n]');
xlabel('n');
ylabel('x[n]');
```

Plots:




A band -pass signal with different values of w1 and w2 are considered .

*this part explores band-pass signals , where w1 and w2 define the frequency range of the signal.

*For the given values of w1 and w2, the script computes x[n] and plots it for each case.

By changing w1, w2 you can observe how the bandwidth of the signal affets the time-domain signal x[n] . Narrower bands result in longer duration signals , and wider bands result in shorter duration signals.

This explains the frequency content of a signal in the frequency domain relates to its time domain representation through the inverse DTFT . different values of parameters like w1,wc,w2 lead to different time domain signals , explains relationship between frequency and time domains.