Bangladesh University of Engineering & Technology

Department of Electrical and Electronics Engineering

Lab Report

Experiment Name:

4. Frequency Domain Analysis of DT Signals and Systems

Taught By

Shahed Ahmed
Lecturer, EEE, BUET

Barproda Halder
Lecturer, PT, EEE, BUET

Prepared By

Anindya Kishore Choudhury
STD. ID: 1906081, Group: B1
Course: EEE 312
Email: anindyakchoudhury@gmail.com

22 January 2023

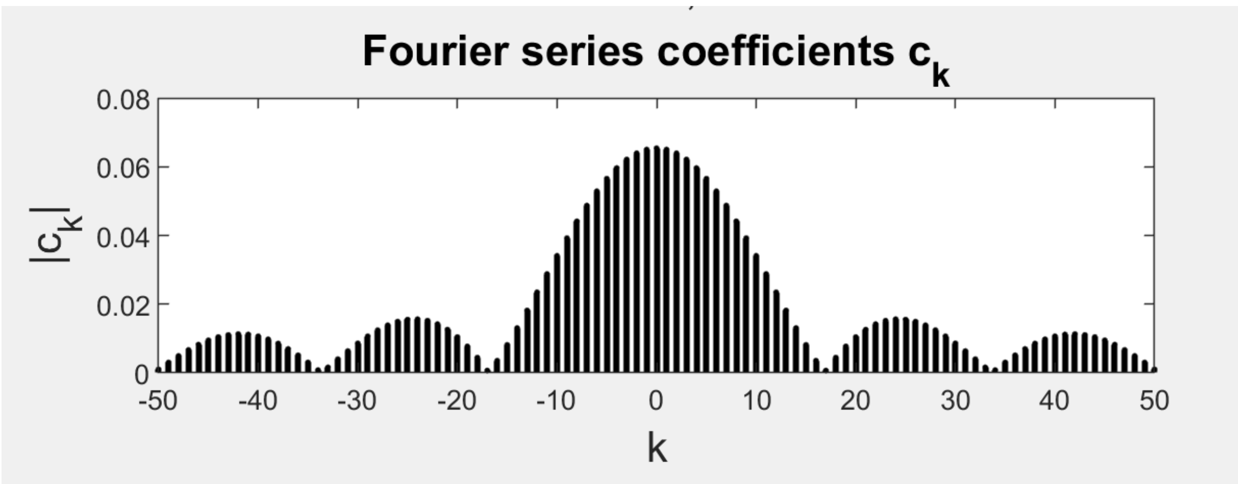
DTFS:

Problem No. 1

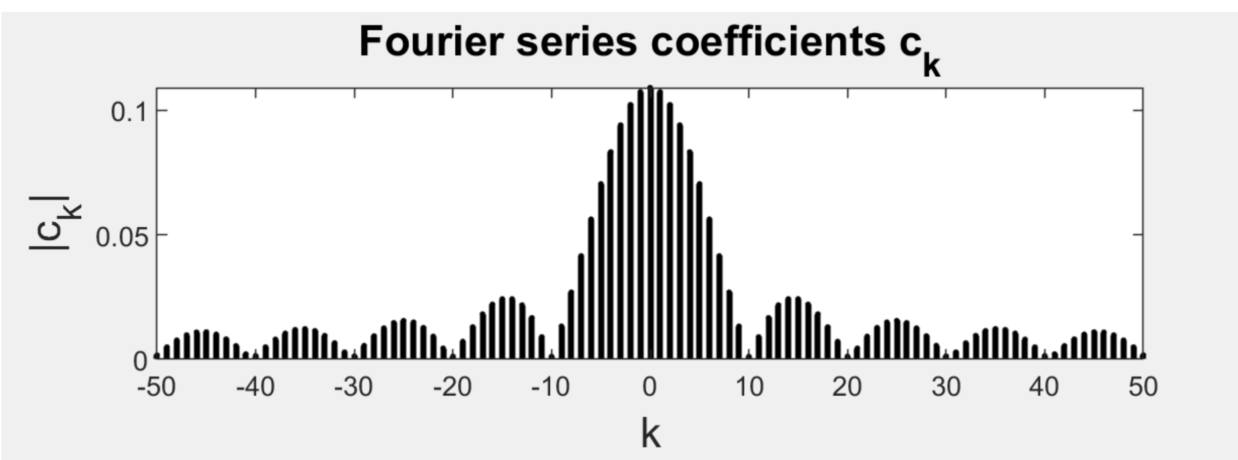
Code:

(As Given in the Lab sheet Example)

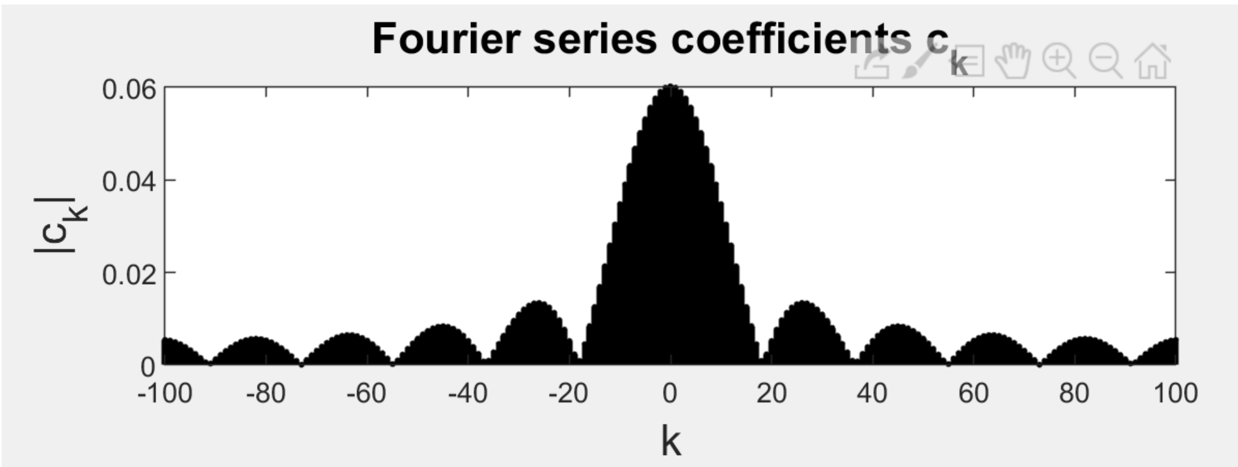
Time Interval: 0 to T



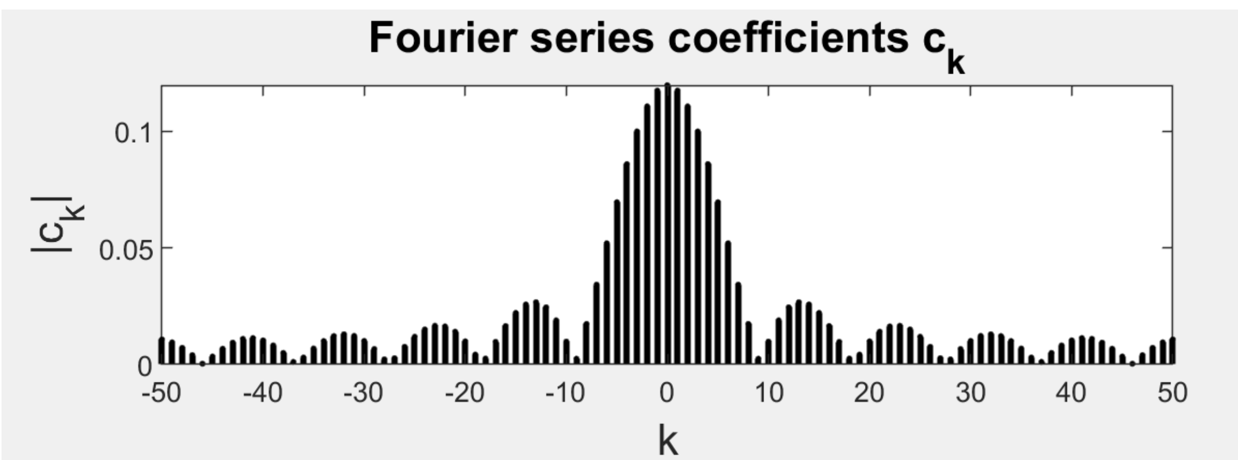
Time Interval: -T/4 to 3T/4



Time Interval: -T to T



Time Interval: -T/2 to T/2



Explanation:

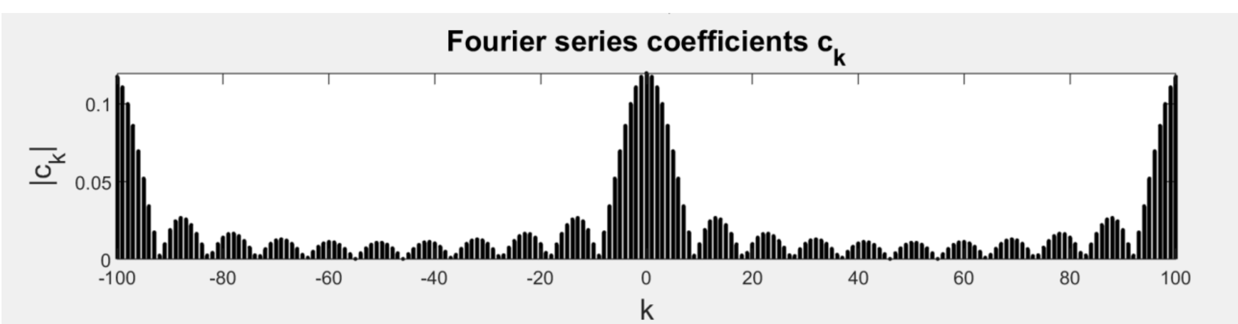
1. From changing the time domain -T/2 to T/2 to the 0 to T, we can evidently see that the magnitude spectrum has got wider resulting from the shrunken pulse width in the time domain.
2. From changing the time domain -T/2 to T/2 to the -T/4 to 3T/4. the magnitude spectrum is similar in both cases as the pulse width are same with a shift.
3. As the period got twice larger, the magnitude spectrum has got denser in -T to T from -T/2 to T/2.

Problem 2:

The modification of the code of Example 1 is given below:

```
N = length(x);
Nc = N; %TOTAL NO OF POINTS IN A PERIOD
if mod(Nc, 2) ==0
    k= -Nc:Nc;
else
    k= -(Nc-1):(Nc-1);
end
```

Output:



From the output, we can find the periodic repetition of c_k.

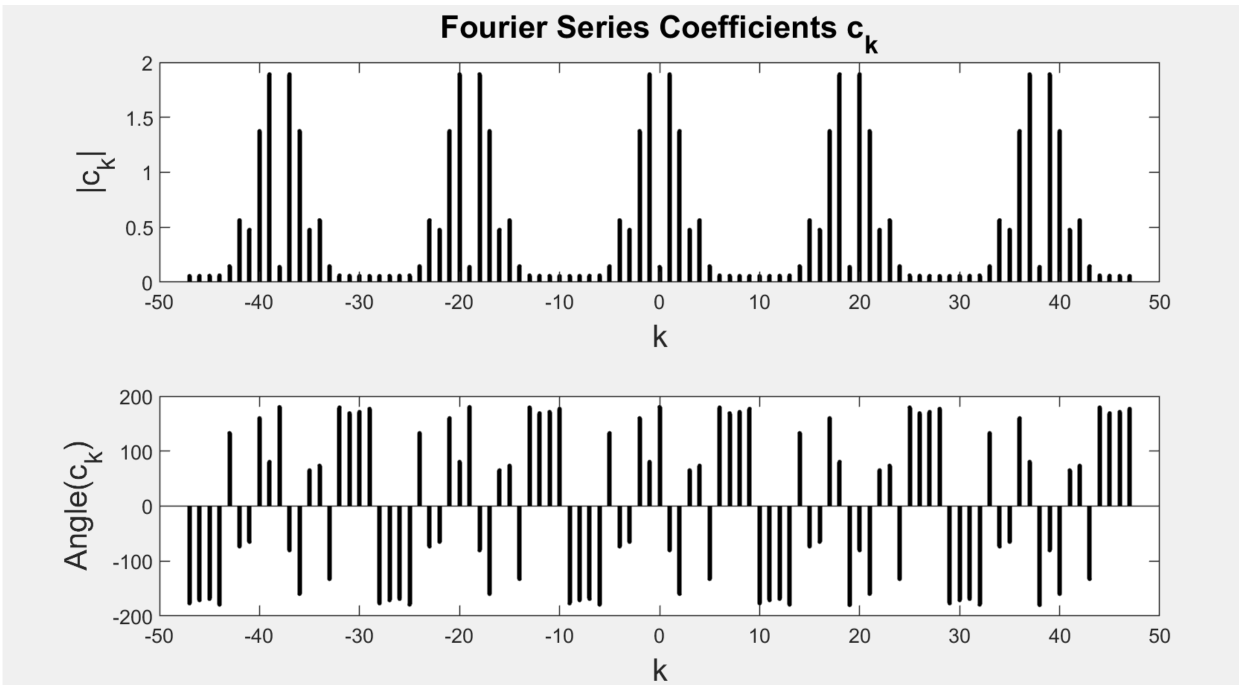
Problem 3:

Code:

```
clc;
clearvars;

Fs = 2600;
dt = 1/Fs;
%GENERATING THE SIGNAL
T = 0.007; %PERIOD OF THE PULSE TRAIN
D = .1; %DUTY CYCLE
PW = D*T; %PULSE WIDTH
f = 1/T; %ANALOG FREQUENCY
t = 0:dt:T; %TIME INTERVAL FOR A PERIOD
n = t/dt; %INDEX FOR DATA POINTS IN A PERIOD
L = PW/dt; %DATA POINTS IN THE THE HIGH TIME
x = 4*sin(300*pi*t) - 3*cos(600*pi*t) + cos(1200*pi*t);
N = length(x); %TOTAL NO DATA POINTS IN A PERIOD
Nc = 5*N;
if mod(Nc,2)==0
    k=-Nc/2:Nc/2-1;
else
    k=-(Nc-1)/2:(Nc-1)/2;
end
c = zeros(1,length(k));%INITIALIZING FOURIER
COEFFICIENTS
for i1=1:length(k)
    for i2=1:length(x)
        c(i1)=c(i1)+1/N*x(i2)*exp(-
1i*2*pi*k(i1)*n(i2)/N);
    end
end

subplot(2,1,1)
stem(k,abs(c), '.k', 'linewidth', 2); xlabel('k',
'fontsize', 14); ylabel('|c_k|','fontsize', 14);
title('Fourier Series Coefficients c_k', 'fontsize',
14);

subplot(2,1,2)
```

```
stem(k,angle(c)*180/pi, '.k', 'linewidth', 2);
xlabel('k', 'fontsize', 14);
ylabel('Angle(c_k)','fontsize', 14)
```

Output:



So this the magnitude and phase spectrum for the input signal 4*sin(300*pi*t) - 3*cos(600*pi*t) + cos(1200*pi*t) .

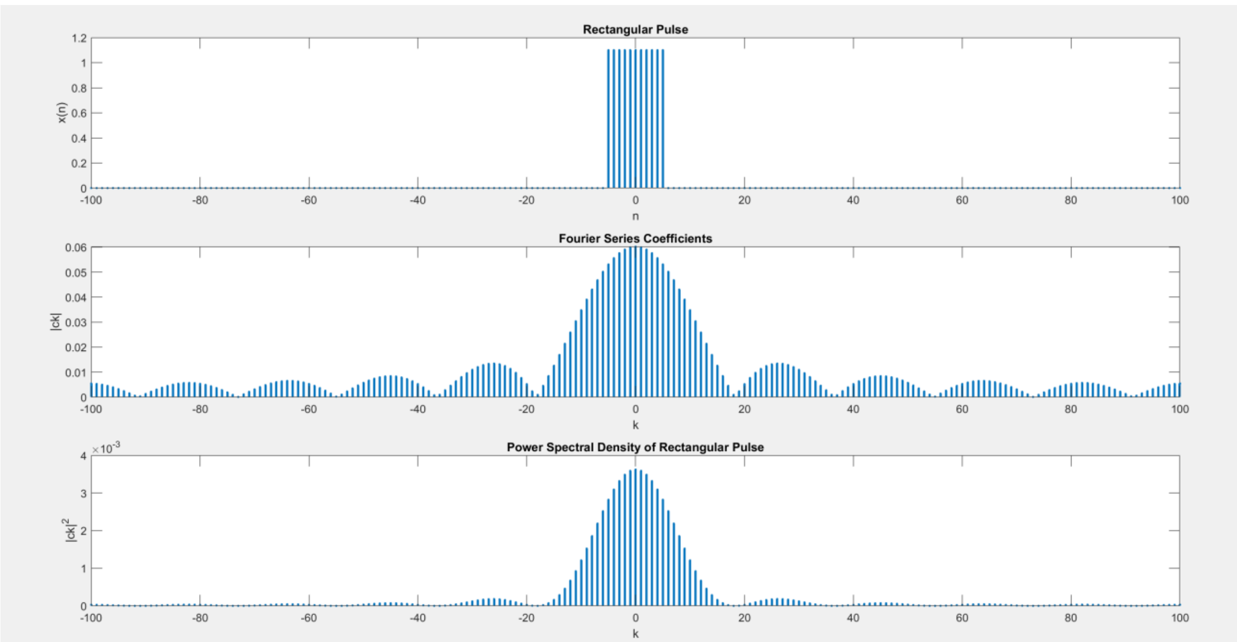
Problem 4:

Code:

```
clc;
clearvars;

Fs = 100e3;
dt = 1/Fs;
% GENERATING THE RECTANGULAR PULSE TRAIN
T = 1e-3; %PERIOD OF THE PULSE TRAIN
D = 0.1; %DUTY CYCLE
PW = D*T; %PULSE WIDTH
f = 1/T; %ANALOG FREQUENCY
t = -T:dt:T; %TIME INTERVAL FOR A PERIOD
n = t/dt;   %INDEX FOR DATA POINTS IN A PERIOD
L = PW/dt;  %DATA POINTS IN THE THE HIGH TIME
x = zeros(1,length(t));   %INITIALIZING A SINGLE
RECTANGULAR PULSE
x(abs(n)<=L/2) = 1.1;     %GENERATION OF A SINGLE
RECTANGULAR PULSE
% Fourier Coefficients Calculation
N = length(x);     %TOTAL NO DATA POINTS IN A PERIOD
Nc = N;
if mod(Nc,2) == 0
    k = -Nc/2:Nc/2-1;
else
    k = -(Nc-1)/2:(Nc-1)/2;
end
c = zeros(1,length(k));   %INITIALIZING FOURIER
COEFFICIENTS
for idx1 = 1:length(k)
    for idx2 = 1:length(x)
        c(idx1) = c(idx1) + 1/N*x(idx2)*exp(-
1i*2*pi*k(idx1)*n(idx2)/N);
    end
end
psd = c.*conj(c);

subplot(3,1,1)
```

```
stem(n,x,'.',"LineWidth",2); xlabel("n");
ylabel("x(n)"); title("Rectangular Pulse")

subplot(3,1,2)
stem(k,abs(c),'.',"LineWidth",2); xlabel("k");
ylabel("|ck|"); title("Fourier Series Coefficients")

subplot(3,1,3)
stem(k,psd,'.',"LineWidth",2); xlabel("k");
ylabel("|ck|^2"); title("Power Spectral Density of
Rectangular Pulse")
```

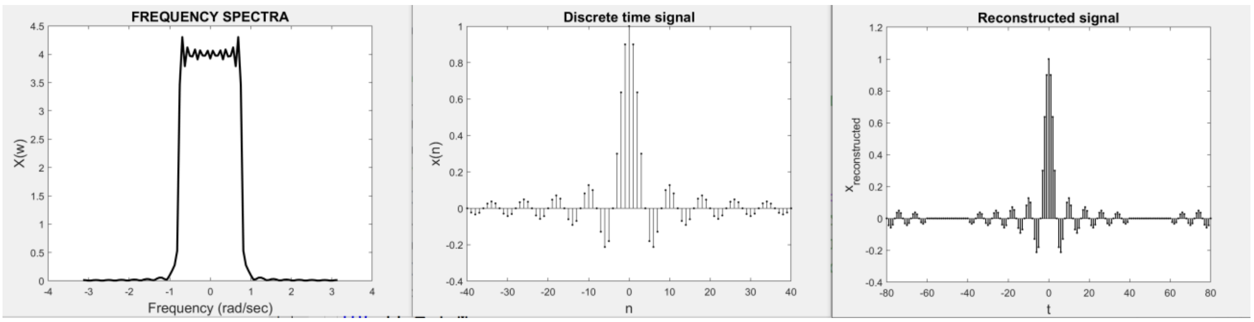
Output:



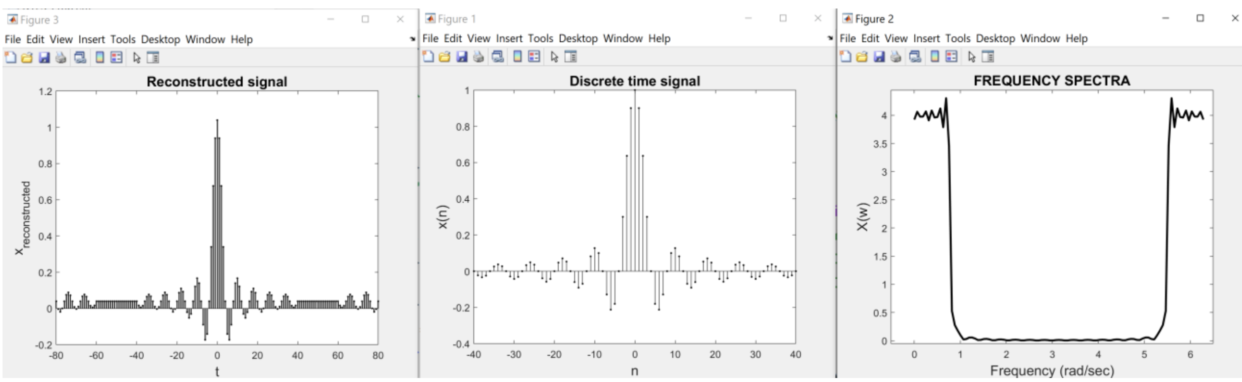
DTFT:

Problem 1 and 2:

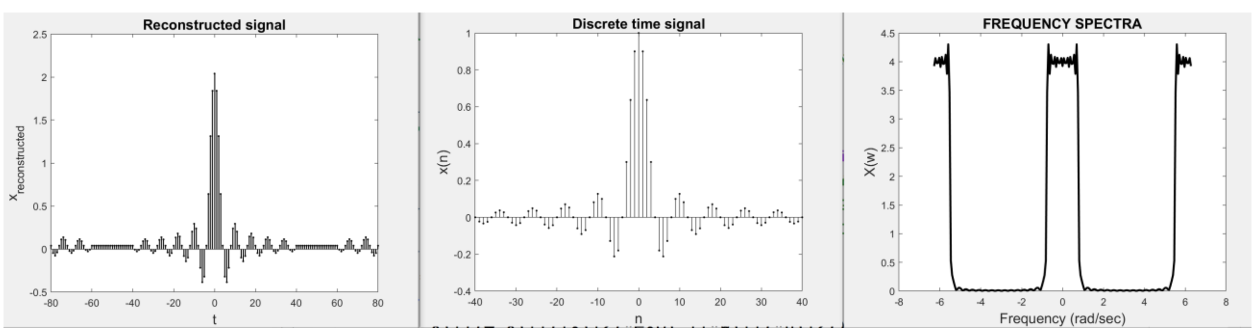
-pi to pi:



0 to 2pi:



-2*pi to 2*pi:



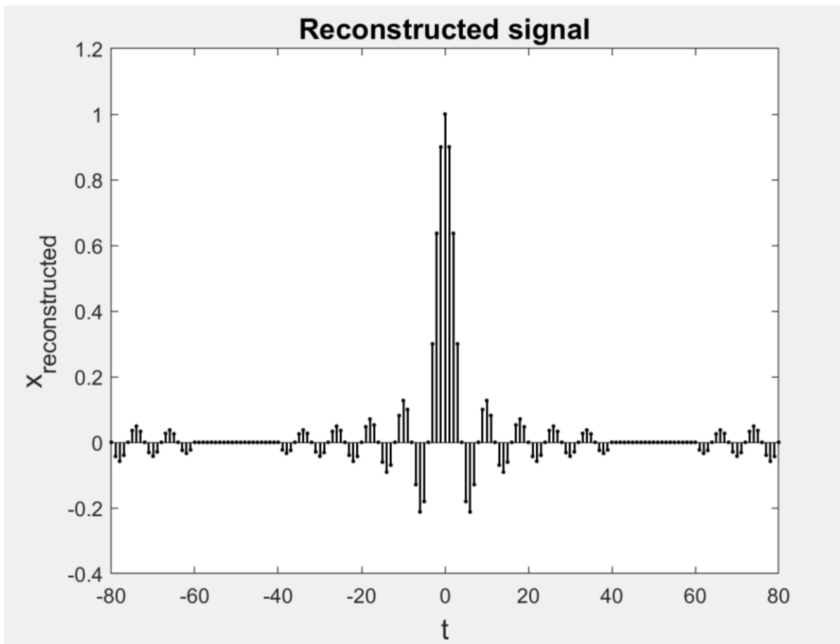
Observation:

So by increasing the frequency grid to 0 to 2pi, we have got the same discrete time and reconstructed signal but the frequency spectra is a bit shifted.

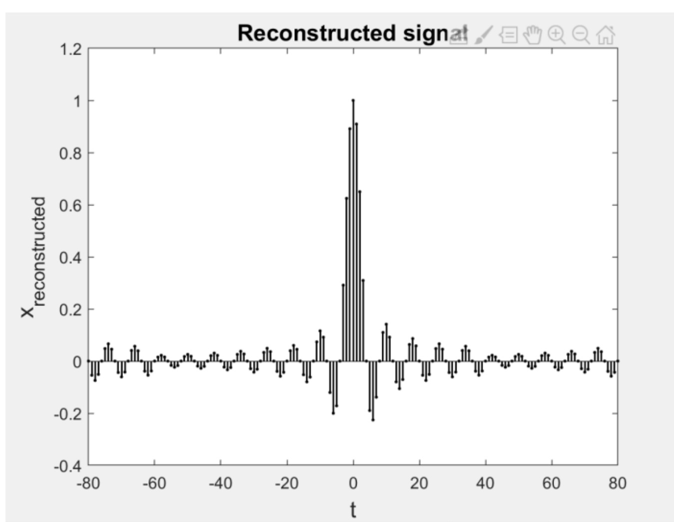
In the -2pi to the 2pi case, we have increased the frequency range and hence, we can observe the periodicity of DTFT in the output.

Problem 3:

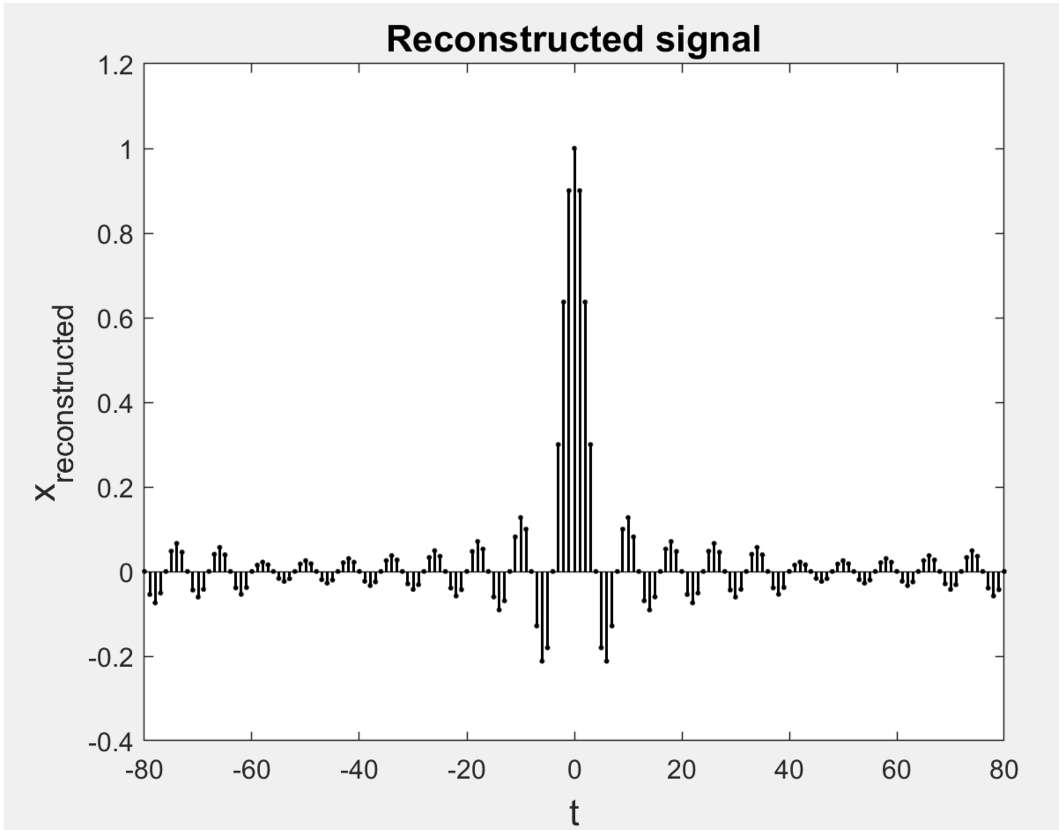
n= -40 to 40



n= -40:120



n= -40:80

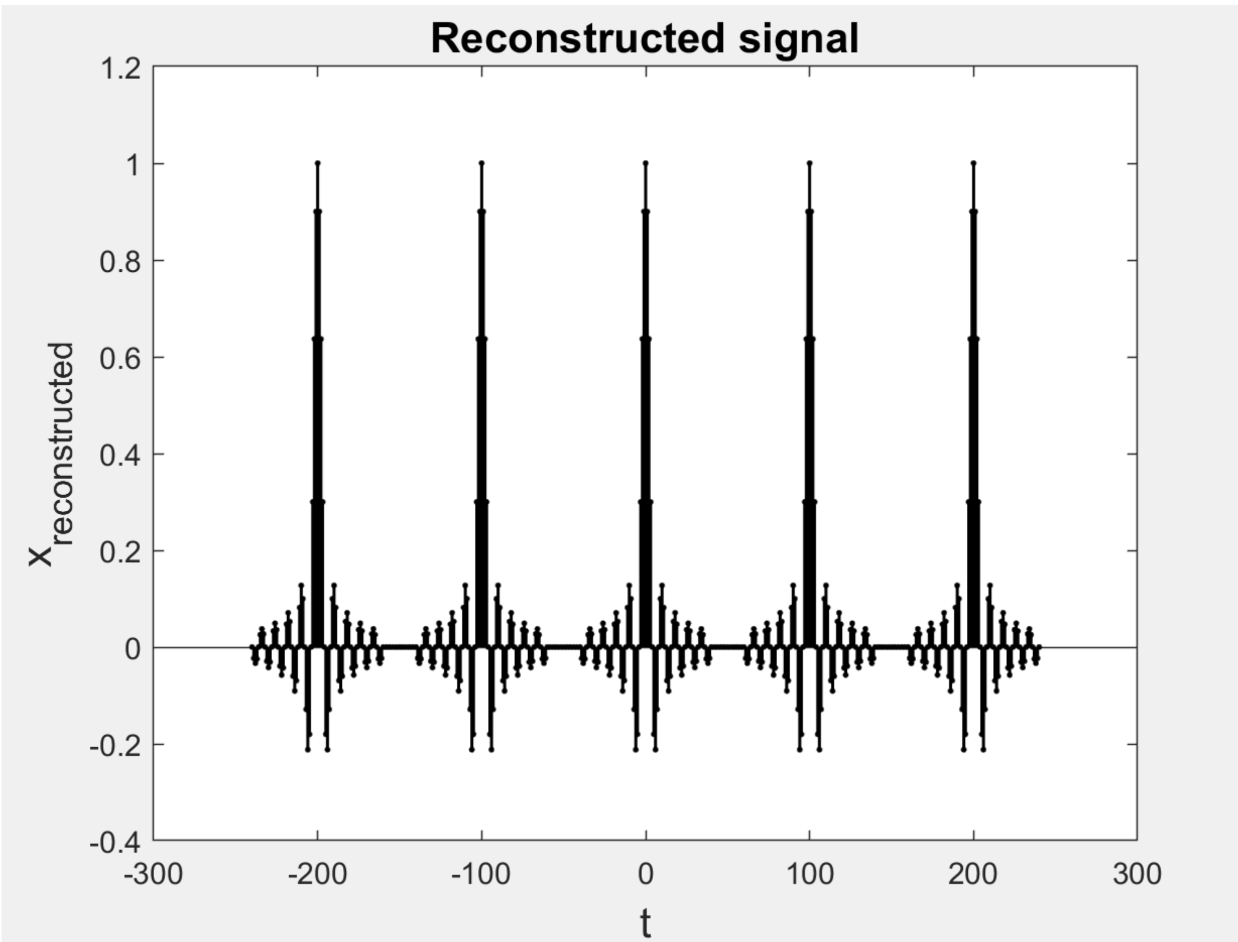


Observation:

In the first output where n is ranging from -40 to 40, we got a flat line in the reconstructed signal. But when we have increased the n, the flat line diminishes and we have retrieved a part of the signal. So, increasing n results in better reconstruction of the signal.

Problem 4:

n_re= -240:240



Observation:

Here, for this value of n_re, we have got a periodic reconstruction of a signal which is aperiodic basically. In fact, Matlab treated it as a periodic signal. The reason behind this is more mathematical than it seems like a software issue. The Fourier transform of any discrete time signal is a periodic function of frequency.

We can think of the discretization of a continuous time signal as amplitude modulation on an impulse train. An impulse train has a Fourier transform of impulse train, with impulses at the fundamental frequency and all its harmonics. Time domain multiplication is frequency domain convolution. Convolving any function with an impulse train gives us a periodic function.

(Source of the Understanding: Quora and Stackoverflow search)

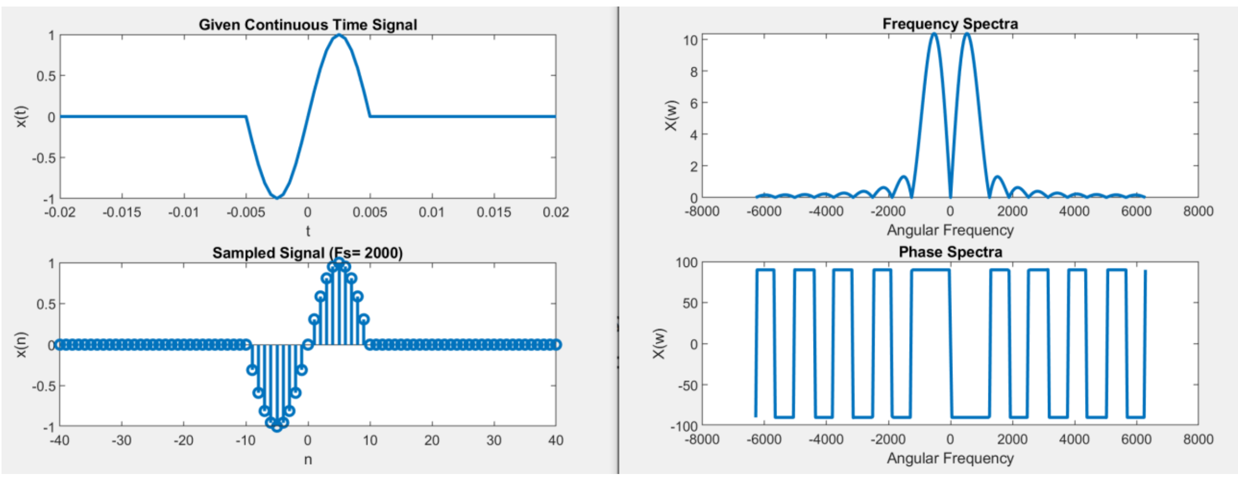
Problem 5:

Code:

```
clc;
clearvars;

f = 100;
Fs = 2000;
dt = 1/Fs;
t=-0.02:dt:0.02;
n = t/dt;
x = zeros(1,length(n));
for i = 1:length(t)
    if  ((t(i) >= -0.005) && (t(i) <= 0.005))
        x(i) = sin(2*pi*f*t(i));
    end
end

figure(1)
subplot(2,1,1)
plot(t,x,"LineWidth",2); xlabel("t"); ylabel("x(t)")
title("Given Continuous Time Signal")

subplot(2,1,2)
stem(n,x,"Linewidth",2); xlabel("n"); ylabel("x(n)")
title("Sampled Signal (Fs= 2000)")

M = 301;
w = linspace(-pi,pi,M);
dw = w(2) - w(1);
x_r = zeros(1,M);
for i = 1:length(x)
    for b = 1:M
        x_r(b) = x_r(b) + x(i)*exp(-1i*w(b)*n(i));
    end
end

figure(2)
subplot(2,1,1)
```

```
plot(w*Fs,abs(x_r),"LineWidth",2); xlabel("Angular
Frequency"); ylabel("X(w)");
title("Frequency Spectra")

subplot(2,1,2)
plot(w*Fs,angle(x_r)*180/pi,"LineWidth",2);
xlabel("Angular Frequency"); ylabel("X(w)");
title("Phase Spectra")
```

Output:



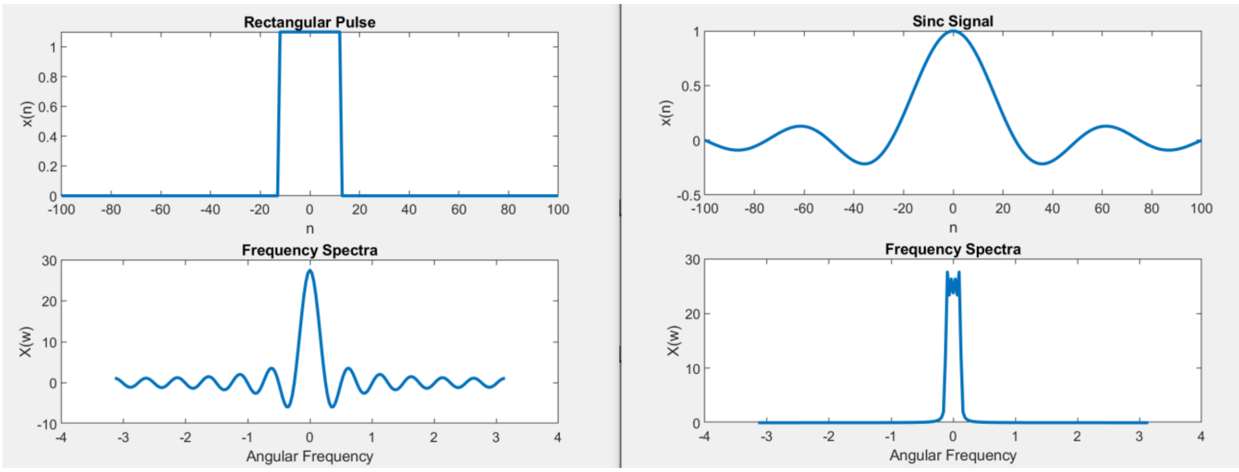
Problem 6:

Code:

```
clc;
clearvars;

%Rectangular Pulse
Fs = 5000;
dt = 1/Fs;
T = 0.01;
f = 100;
t = -0.02:dt:0.02;
n = t/dt;
N = length(t);
D = 0.5;
PW = D*T;
L = PW/dt;
x = zeros(1,N);
x(abs(n) <= L/2) = 1.1;
M = 201;
w = linspace(-pi,pi,M);
dw = w(2)- w(1);
X = zeros(1,M);
for i = 1:length(x)
    for j = 1:M
        X(j) = X(j) + x(i)*exp(-1i*w(j)*n(i));
    end
end

%Sinc funtion
Fs2 = 5000;
dt2 = 1/Fs2;
T2 = 0.01;
f2 = 100;
t2 = -0.02:dt2:0.02;
n2 = t2/dt2;
N2 = length(t);
x2 = sinc(2*f2*t2);
M2 = 201;
w2 = linspace(-pi,pi,M2);
```

```
dw2 = w2(2)- w2(1);
X2 = zeros(1,M2);
for i = 1:length(x)
    for j = 1:M2
        X2(j) = X2(j) + x2(i)*exp(-1i*w2(j)*n2(i));
    end
end

%Plots
figure(1)
subplot(2,1,1)
plot(n,x,"LineWidth",2); xlabel("n"); ylabel("x(n)")
title("Rectangular Pulse")

subplot(2,1,2)
plot(w,X,"LineWidth",2); xlabel("Angular Frequency");
ylabel("X(w)")
title("Frequency Spectra")

figure(2)
subplot(2,1,1)
plot(n2,x2,"LineWidth",2); xlabel("n"); ylabel("x(n)");
title("Sinc Signal")

subplot(2,1,2)
plot(w2,abs(X2),"LineWidth",2); xlabel("Angular
Frequency"); ylabel("X(w)")
title("Frequency Spectra")
```

Output:


Rectangular Pulse
x(n)
n
Frequency Spectra
X(w)
Angular Frequency
Sinc Signal
x(n)
n
Frequency Spectra
X(w)
Angular Frequency

DFT:

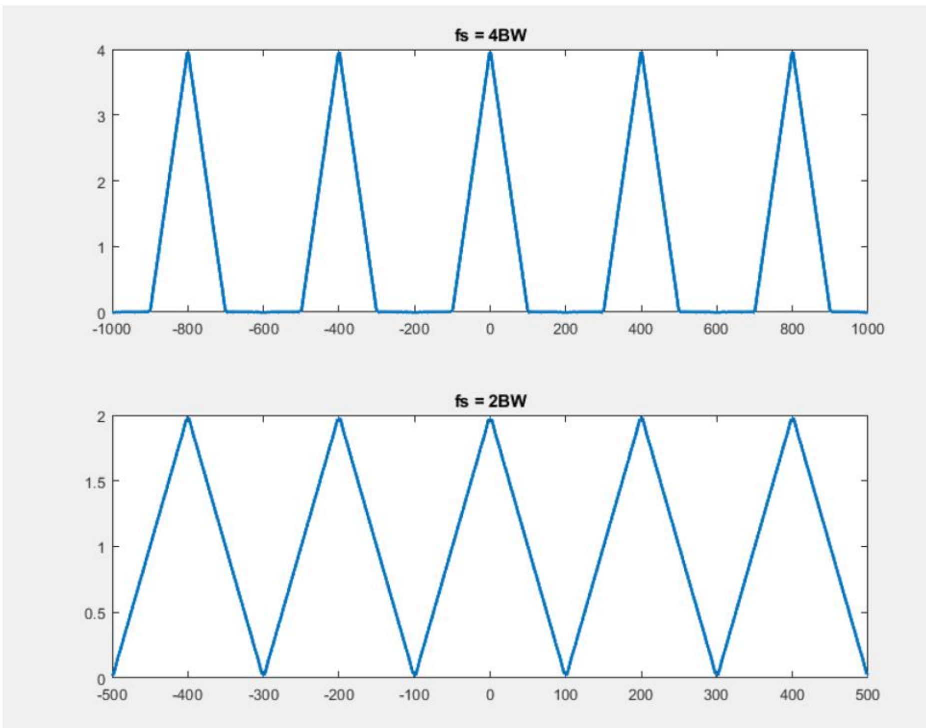
Code:

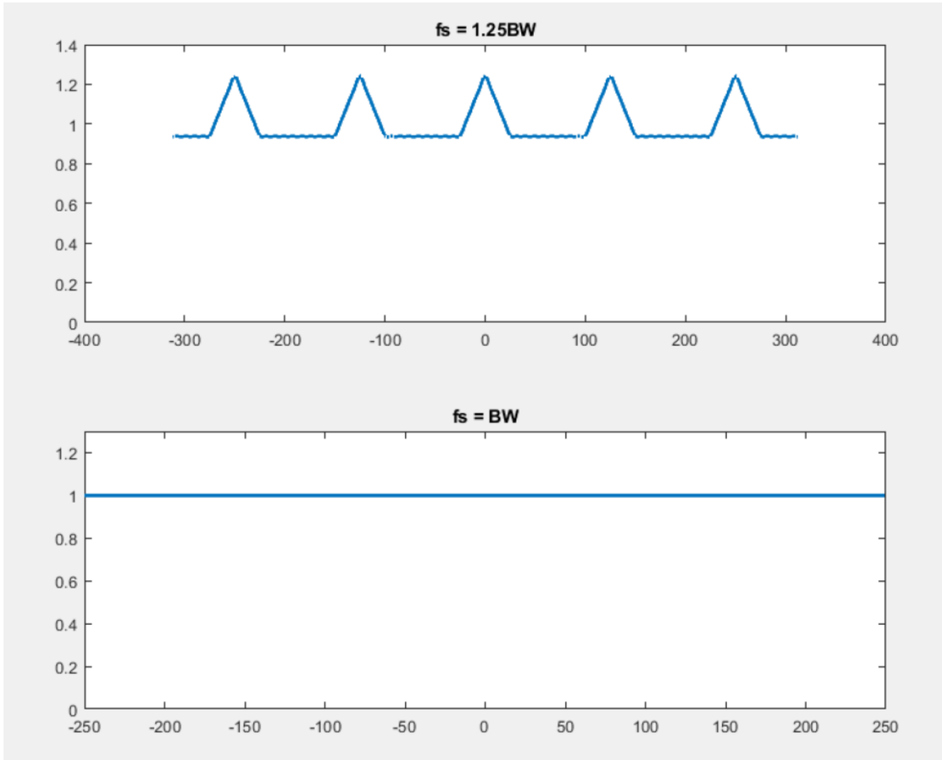
```
clc;
clearvars;

%fs=4*BW
to = .1;
ts = .0025;
fs = 1/ts;
nup = round(to/ts);
nlo = -nup; % generate time index
n = nlo:nup;
t = n*ts;
m = (sinc(100*t)).^2;
N = 1024;% FFT bin size ,after zero padding
fn = [0:1/N:5-1/N]*fs - 5*fs/2;
k = 0:5*N-1;
WN = exp(-1i*2*pi/N);
nk = n'*k;
W = WN.^nk;
M = m*W;
M = fft(m,1024);

figure(1)
subplot(2,1,1)
plot(fn,abs(fftshift(M)),0,0,"LineWidth",2)
title("fs = 4BW")
fs = 2*BW;
ts = 0.005;
fs = 1/ts;
nup = round(to/ts);
nlo = -nup;% generate time index
n = nlo:nup;
t = n*ts;
m = (sinc(100*t)).^2;
N = 1024;% FFT bin size ,after zero padding
fn = [0:1/N:5-1/N]*fs-5*fs/2;
k = 0:5*N-1;
WN = exp(-1i*2*pi/N);
```

```
nk = n'*k;
W = WN.^nk;
M = m*W;
%M=fft(m,1024);
subplot(212)
plot(fn,abs(fftshift(M)),0,0,"LineWidth",2)
title("fs = 2BW")
% fs=1.25*BW
ts = .008;
fs = 1/ts;
nup = round(to/ts);
nlo = -nup;% generate time index
n = nlo:nup;
t = n*ts;
m = (sinc(100*t)).^2;
N = 1024;% FFT bin size ,after zero padding
fn = [0:1/N:5-1/N]*fs-5*fs/2;
k = 0:5*N-1;
WN = exp(-1i*2*pi/N);
nk = n'*k;
W = WN.^nk;
M = m*W;
%M=fft(m,1024);
figure(2)
subplot(211)
plot(fn,abs(fftshift(M)),0,0,"LineWidth",2)
title("fs = 1.25BW")
% fs=BW
ts = .01;
fs = 1/ts;
nup = round(to/ts);
nlo = -nup;% generate time index
n = nlo:nup;
t = n*ts;
m = (sinc(100*t)).^2;
N = 1024;% FFT bin size ,after zero padding
fn = [0:1/N:5-1/N]*fs-5*fs/2;
k = 0:5*N-1;
WN = exp(-1i*2*pi/N);
nk = n'*k;
```

```
W = WN.^nk;
M = m*W;
%M=fft(m,1024);
subplot(212)
plot(fn,abs(fftshift(M)),0,0,"LineWidth",2); ylim([0
1.5])
title("fs = BW")
```

Output:





Observation:

The output we see here are the spectras sampled at different sampling frequencies. In this output, we can see that the sampling didn't provide us any benefits when the sampling rate is less than Nyquist rate (i.e, fs= 1.25 BW and fs = BW). On the other hand, we get a reasonably workable signal at fs = 2BW and exactly a correct signal when fs = 4BW > 2BW as recommended by Nyquist.