Experiment 1: Sampling

Digital Signal Processing Lab

Avinab Saha,

Roll: 15EC10071, Group: 28

11th, February 2018

Introduction

In signal processing, sampling is the reduction of a continuous-time signal to a discrete-time signal. A common example is the conversion of a sound wave (a continuous signal) to a sequence of samples (a discrete-time signal). A sample is a value or set of values at a point in time and/or space. A sampler is a subsystem or operation that extracts samples from a continuous signal. A theoretical ideal sampler produces samples equivalent to the instantaneous value of the continuous signal at the desired points.

Moving on, the problem statement required us to perform the following simulations:

A. Input an analog signal waveform and obtain the DFT of the waveform for N = 12, 64, 128, 256 and to plot the magnitude spectra with Fs = 12KHz and observe the variation with N.
B. Repeat the above process for different sampling frequencies such as Fs= 4 KHz , 5 KHz and 8 KHz and observe aliasing.
C. Perform sampling of a Square Wave and obtain DFT of the sampled signal with Fs = 20 KHz and N = 256.
D. Sample the Square Wave signal at different sample frequencies and reconstruct the signal and compare the efficiency of reconstruction for different values of Fs.
E. Perform interpolation of a signal sampled at 12 KHz by upsampling it by a factor of 2 and to compare both its time domain waveform and frequency domain spectrum with that of the signal when sampled at 24 KHz.
F. Perform decimation of a signal sampled at 12 KHz by downsampling it by a factor of 2 and to observe the frequency domain spectrum of the downsampled signal in two cases – with and without the anti-aliasing filter. Also to perform signal reconstruction from the decimated signal.
G. Input an analog signal and perform Sample and Hold (ZOH) followed by low pass filtering for signal reconstruction.
H. Input a Bandpass Signal and perform bandpass sampling and to show DFT for different sampling frequencies.

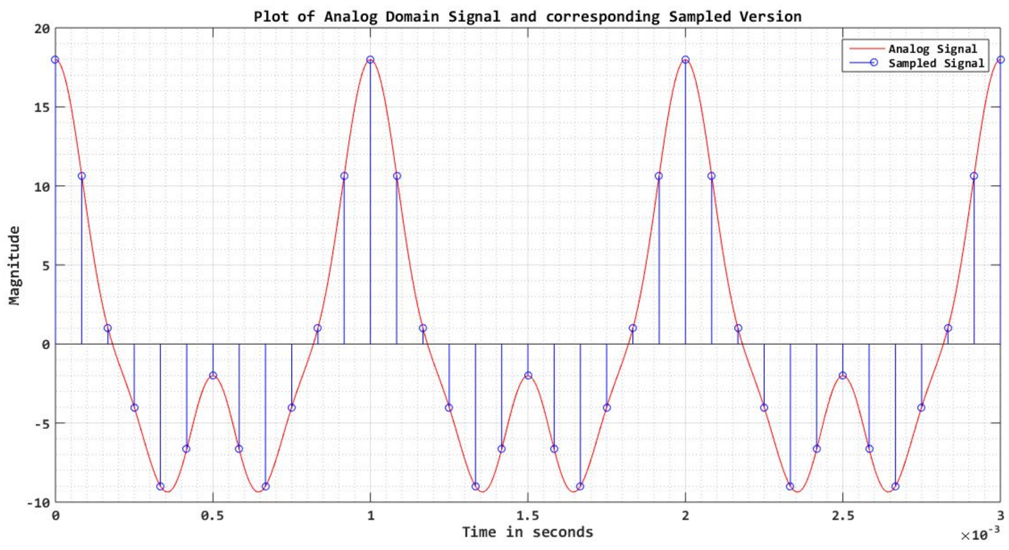
Part A :

Problem Statement:

Input an analog signal waveform and obtain the DFT of the waveform for N = 12, 64, 128, 256 and to plot the magnitude spectra with Fs = 12KHz and observe the variation with N.

Code and Observations :

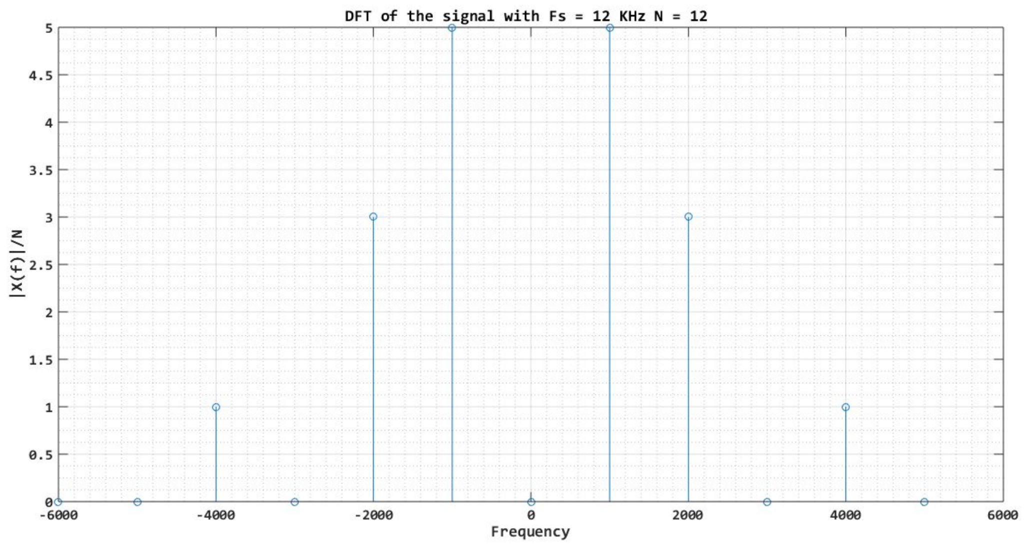
```
% Sampling frequency of 12KHz
f_max= 1.2e5;
f_min = 1e3;
cycles = 3;
% Forming the time axis
t = [0:1/f_max:(cycles/f_min)];
% Defining the Analog Signal
x = 10*cos(2*pi*1000*t) + 6*cos(2*pi*2000*t) + 2*cos(2*pi*4000*t);

% Plot Of Analog Signal
plot(t,x,'r');
hold on;
title('Plot of Analog Domain Signal and corresponding Sampled Version');
xlabel('Time in seconds');
ylabel('Magnitude');
grid on;
grid minor;
```



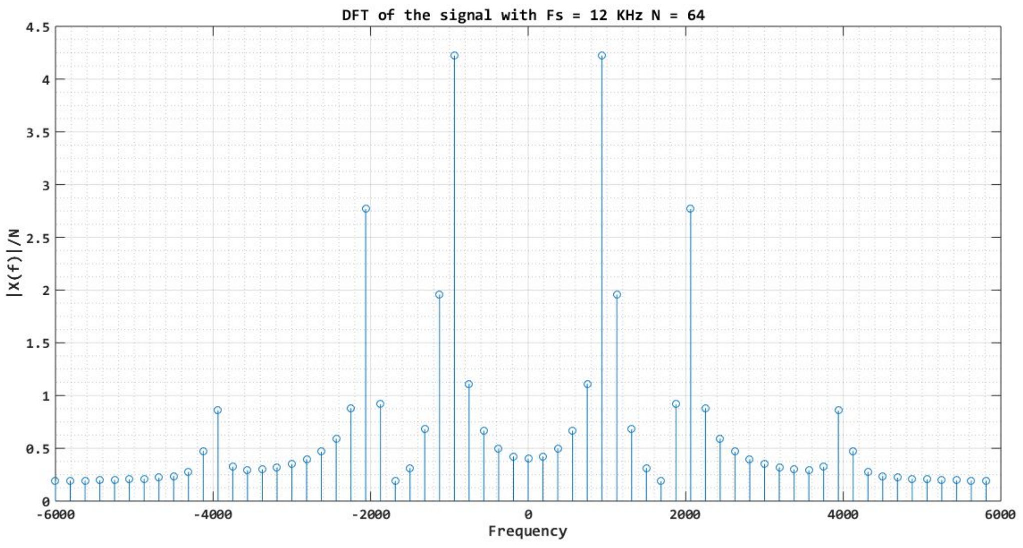
Plot Of Analog Input Signal and the corresponding Sampled Signal

```
% Plotting of sampled signal
fs= 12000;
```

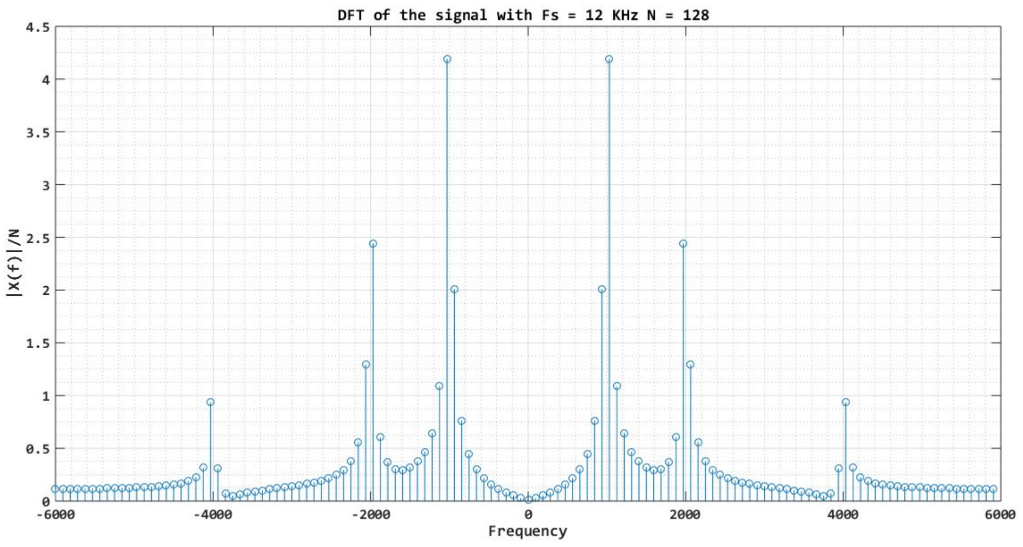
```
ts = [0:1/fs:cycles/f_min];
xs = 10*cos(2*pi*1000*ts) + 6*cos(2*pi*2000*ts) + 2*cos(2*pi*4000*ts);
stem(ts,xs,'b');
legend('Analog Signal','Sampled Signal');

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');

% DFT for different values of N
% List for different values of N
N = [12 64 128 256];

for i=1:length(N)
ts = [0:1/fs:N(i)/f_min];
xs = 10*cos(2*pi*1000*ts) + 6*cos(2*pi*2000*ts) + 2*cos(2*pi*4000*ts);
f_range = [-fs/2:fs/N(i):fs/2-fs/N(i)];
y_fft = fftshift(abs(fft(xs,N(i))/N(i)));
% Plotting on a new figure for each value of N.
figure();
stem(f_range,y_fft);
xlabel('Frequency');
ylabel('|X(f)|/N');
title(['DFT of the signal with Fs = 12 KHz N = ' num2str(N(i)) ' ']);

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');
grid on;
grid minor;
end
```



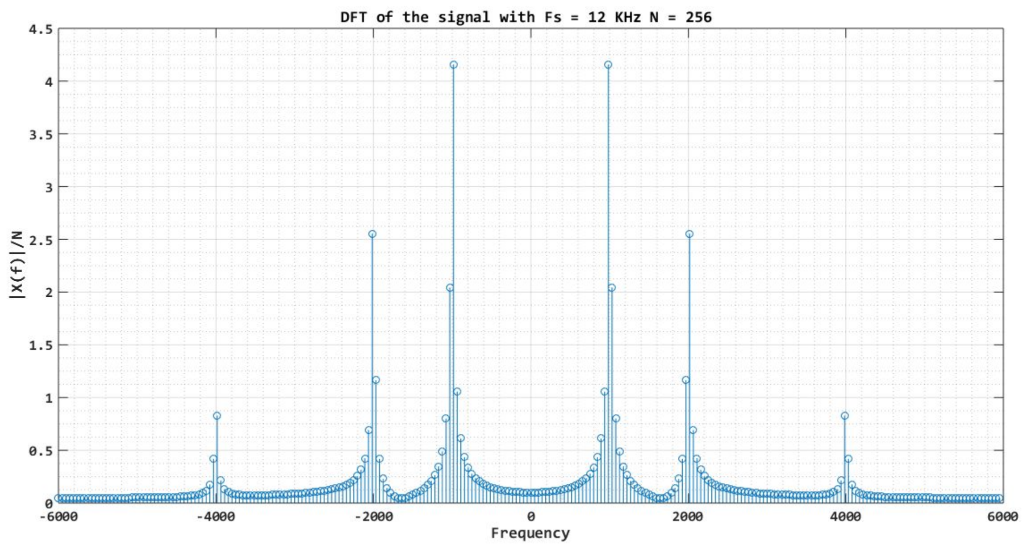
DFT with Sampling Frequency 12 kHz and N=12



DFT with Sampling Frequency 12 kHz and N=64



DFT with Sampling Frequency 12 kHz and N=128



DFT with Sampling Frequency 12 kHz and N=256

Discussions :

In Part A, the normalised DFT of the given analog waveform was plotted by using the inbuilt function **fft** for computing the fast fourier transform of the signal. The function **fftshift** was used to make the spectrum of the signal symmetric about the origin. It is observed that as the number of sample points is increased, the error between the theoretical magnitude and the obtained spectrum reduces which occurs because on incrementing N..

The analog signal contained three sinusoids corresponding to frequencies **1 KHz, 2 KHz and 4 KHz**, the frequency spectrum of the signal clearly exhibits a peaking at those frequencies which become more prominent as the value of N goes . As DFT is periodic with 2[] with the complete spectrum information captured in the range -[] to [], we have plotted the DFT in the analog frequency range -Fs/2 to Fs/2 (Fs is frequency of sampling) which essentially maps to the discrete frequency range -[] to [].

Here, the maximum frequency content of the signal is 4 KHz and the sampling frequency is 8 KHz which satisfies the Nyquist sampling frequency criteria and hence the original signal can be reconstructed..

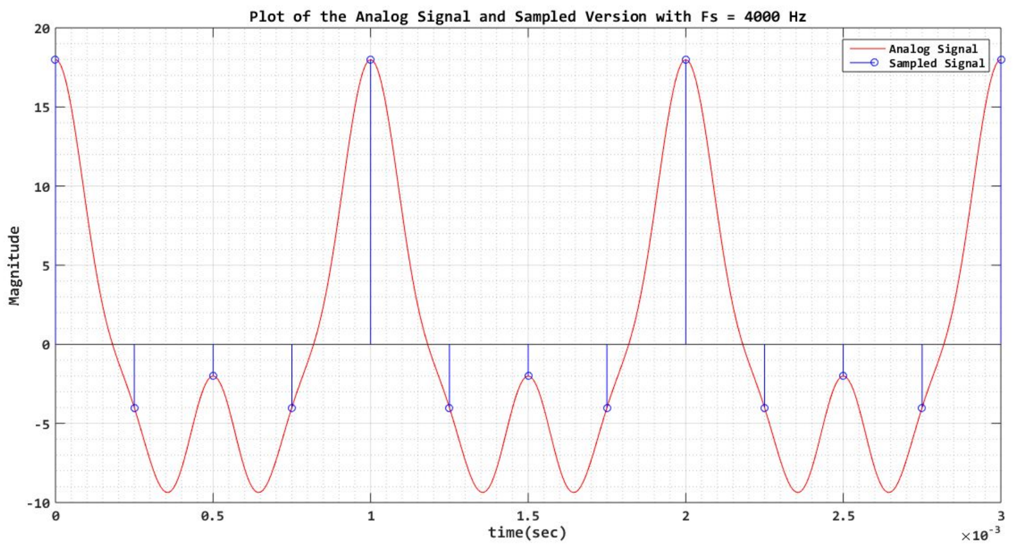
Part B :

Problem Statement:

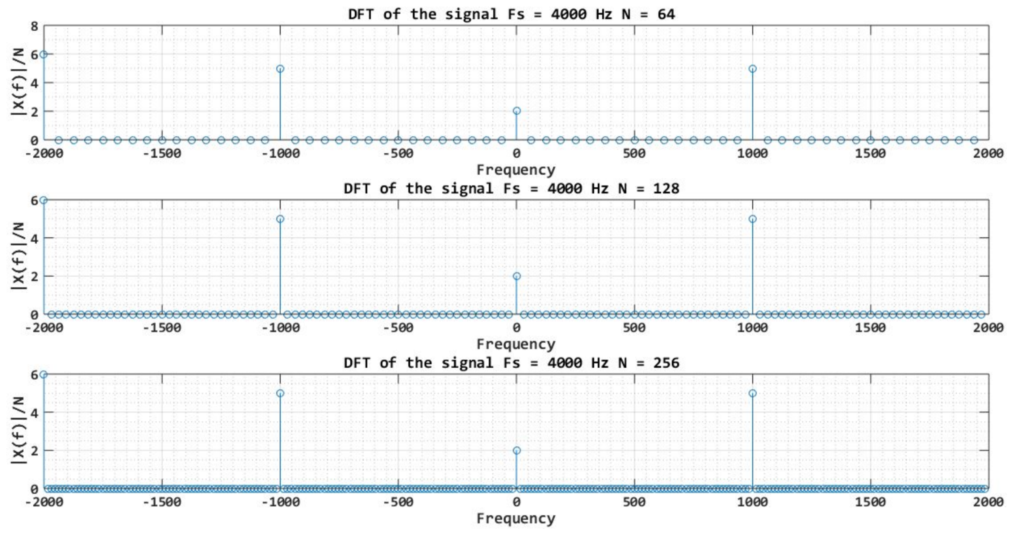
Repeat the above process for different sampling frequencies such as Fs= 4 KHz , 5 KHz and 8 KHz and observe aliasing.

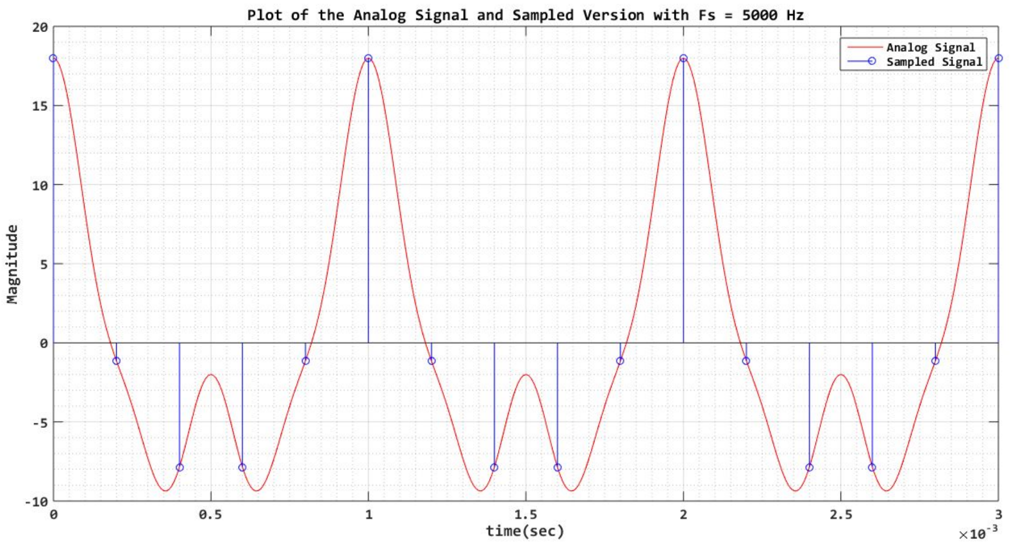
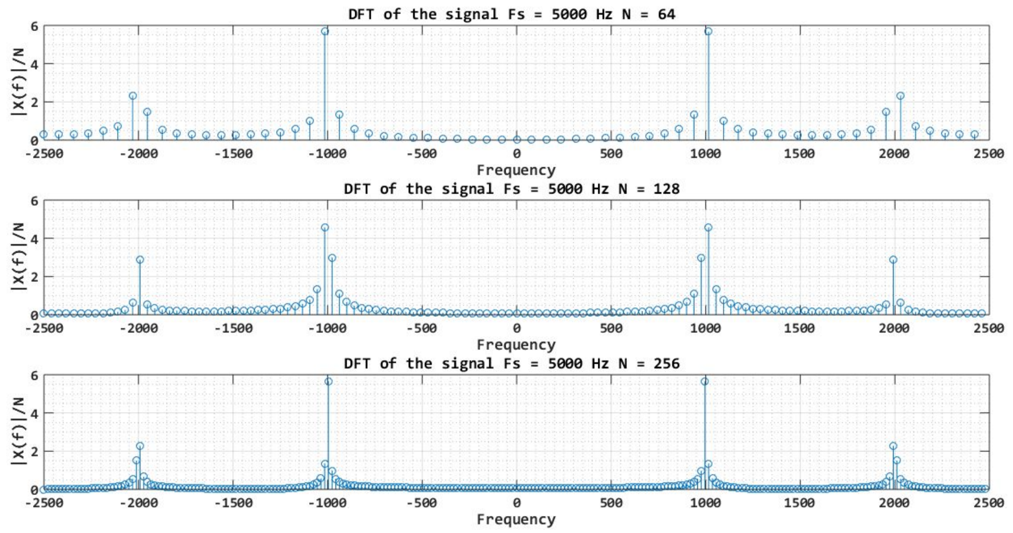
Code and Observations :

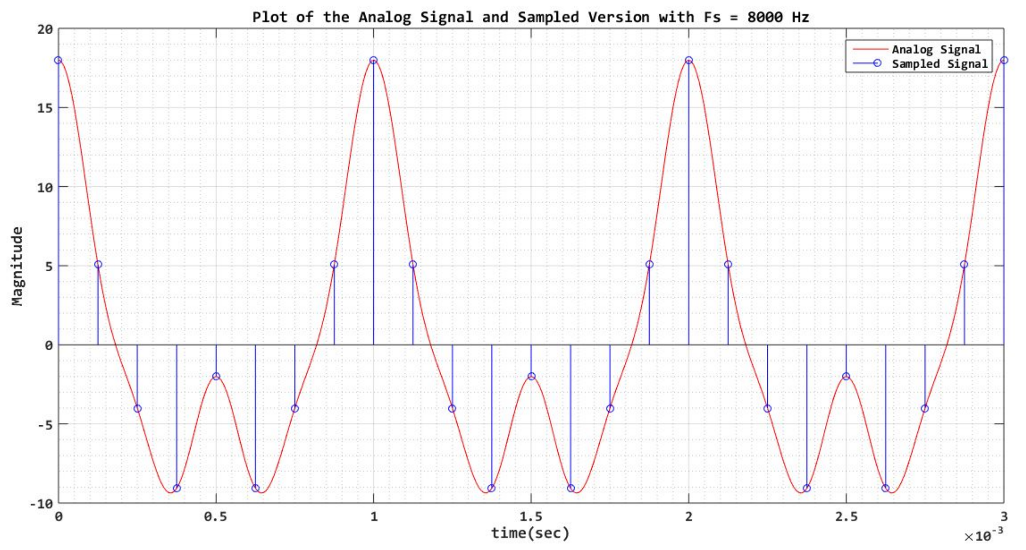
```
% Setting up a few parameters
f_max= 1.2e5;
f_min = 1e3;
cycles = 3;
% Forming the time axis
t = [0:1/f_max:(cycles/f_min)];
% Defining the Analog Signal
x = 10*cos(2*pi*1000*t) + 6*cos(2*pi*2000*t) + 2*cos(2*pi*4000*t);

% List to store different sampling frequencies
fs = [4000 5000 8000];

% Plotting of sampled signal
for i=1:length(fs)
    ts = [0:1/fs(i):cycles/f_min];
    xs = 10*cos(2*pi*1000*ts) + 6*cos(2*pi*2000*ts) + 2*cos(2*pi*4000*ts);
    figure();
    plot(t,x,'r');
    hold on;
    title(['Plot of the Analog Signal and Sampled Version with Fs = ' num2str(fs(i)) '
Hz']);
    xlabel('time(sec)');
    ylabel('Magnitude');
    stem(ts,xs,'b');
    legend('Analog Signal','Sampled Signal');

    % Setting Font Size, Font Name, Font weight and Background color
    set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
    set(gcf,'color','w');
    grid on;
    grid minor;

    % DFT for different values of N
    % List for different values of N

    N = [64 128 256];
    figure();
    for j=1:length(N)
        ts = [0:1/fs(i):(N(j)-1)/fs(i)];
        xs = 10*cos(2*pi*1000*ts) + 6*cos(2*pi*2000*ts) + 2*cos(2*pi*4000*ts);
        f_range = [-fs(i)/2:fs(i)/N(j):fs(i)/2-fs/N(j)];
        y_fft = fftshift(abs(fft(xs)/N(j)));
        % Subplot created
        subplot(3,1,j);
        stem(f_range,y_fft);
        xlabel('Frequency');
        ylabel('|X(f)|/N');
        title(['DFT of the signal Fs = ' num2str(fs(i)) ' Hz N = ' num2str(N(j)) ' ']);

        % Setting Font Size, Font Name, Font weight and Background color
        set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
        set(gcf,'color','w');
        grid on;
        grid minor;
    end
end
```



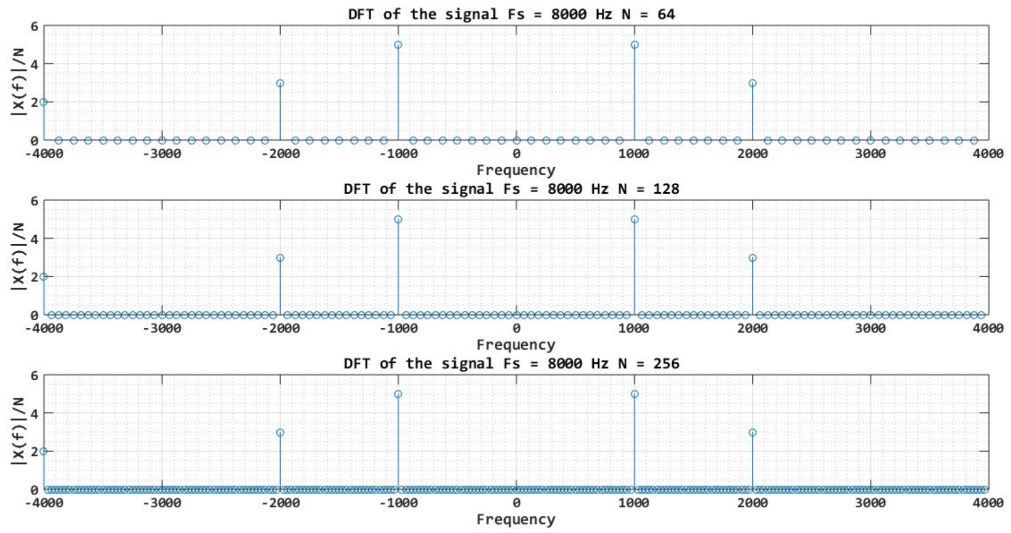
Plot Of Analog Input Signal and the corresponding Sampled Signal (Fs = 4 kHz)



DFT with Sampling Frequency 4 kHz and N= 64,128,256

Plot Of Analog Input Signal and the corresponding Sampled Signal (Fs = 5 kHz)

DFT with Sampling Frequency 5 kHz and N= 64,128,256



Plot Of Analog Input Signal and the corresponding Sampled Signal (Fs = 8 kHz)



DFT with Sampling Frequency 8 kHz and N= 64,128,256

Discussions :

In Part B, we tried to demonstrate aliasing. When a signal is sampled at a frequency lower than twice the maximum frequency content of the signal, aliasing may occurs which means that the spectrum of the original signal gets folded and its images overlaps with the actual spectrum thereby affecting the amplitude of the different frequencies. Thus, the original sequence cannot be reconstructed.

For Fs = 5 KHz, we observe peaks only at frequencies 1 KHz and 2 KHz. However, the amplitude of the spectrum corresponding to 1 KHz is slightly more than that was expected in case of Nyquist sampling rate which indicates that the spectrum of the 4 KHz sinusoid coincided with that of the 1 KHz sinusoid resulting in aliasing. For Fs = 8 KHz, we did not observe any aliasing as sampling is being done at the Nyquist sampling rate and hence all the frequency components are reliably produced.

Part C :

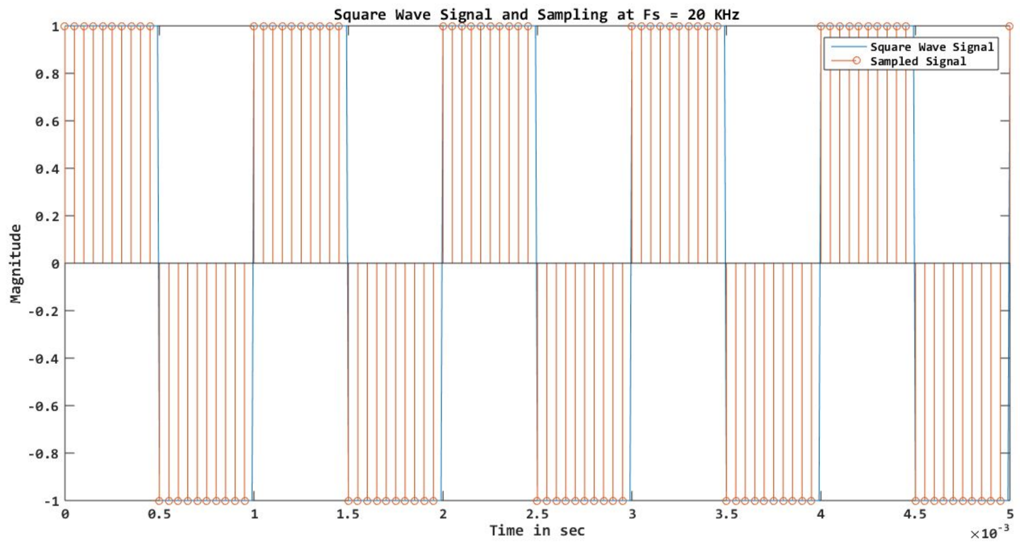
Problem Statement:

Perform sampling of a Square Wave and obtain DFT of the sampled signal with Fs = 20 KHz and N = 256.

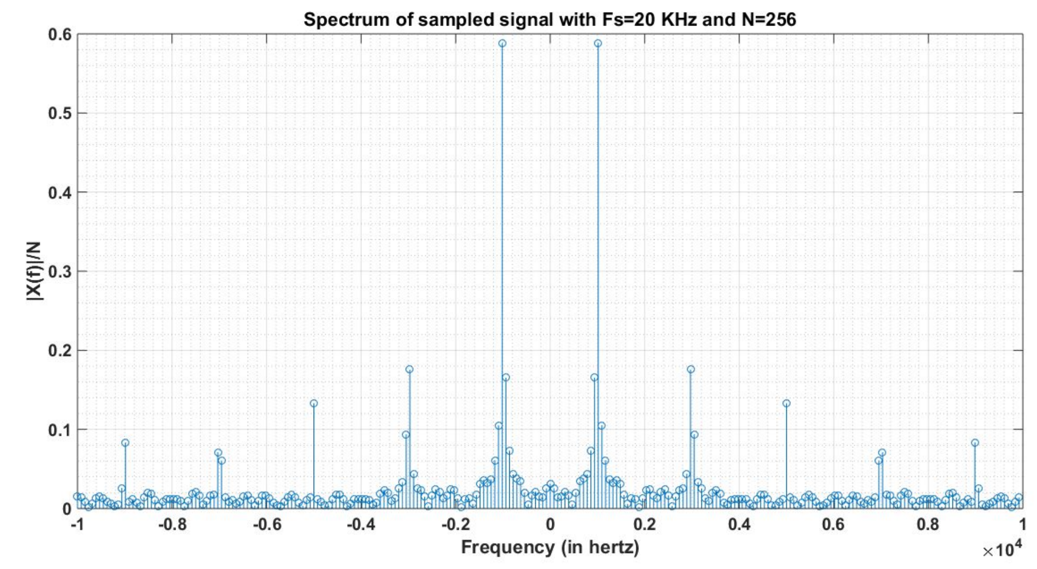
Code and Observations :

```
% Setting up a few parameters
f = 1e5;
f_max = 1e3;
Ncycle = 5;
% Forming the time axis
t = [0:1/f:Ncycle/f_max];
% Defining the Analog Square Signal
squareWeb = square(2*pi*f_max*t,50);
figure();
% Plotting of sampled signal
plot(t,squareWeb);
fs = 20000;
ts = [0:1/fs:Ncycle/f_max];
% Sampled Square web
sq_s = square(2*pi*f_max*ts,50);
hold on;
stem(ts,sq_s);

title(['Square Wave Signal and Sampling at Fs = 20 KHz']);
xlabel('Time in sec');
ylabel('Magnitude');
legend('Square Wave Signal','Sampled Signal');

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');

% DFT of square wave with 256 Samples
N = 256;
ts = [0:1/fs:N/f_max];
sq_s = square(2*pi*f_max*ts,50);
y_dft = fft(sq_s,N);
```

```
freq = [-fs/2:fs/N:fs/2-fs/N];
y_dft_val = fftshift(abs(y_dft)/N);
figure();
stem(freq,y_dft_val);
xlabel('Frequency (in hertz)');
ylabel('|X(f)|/N');
title('Spectrum of sampled signal with Fs=20 KHz and N=256');

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Arial','fontsize',14,'FontWeight','bold');
set(gcf,'color','w');
grid on;
grid minor;
```



Plot Of Analog Square Signal and the corresponding Sampled Signal (Fs = 20 kHz)



DFT with Sampling Frequency 20 kHz and N=256

Discussions :

We observe the DFT of a square wave of time period 1 millisecond with a 50% Duty Cycle. The fourier series expansion of a square wave, the DFT spectrum comprises of sinusoids with the odd harmonics of the fundamental frequency 1 KHz. As we observe, the magnitude of each subsequent harmonic goes on decreasing which indicates the diminishing contribution of the higher frequency harmonics.

An ideal square wave is comprised of infinite number of harmonics and on sampling ata rate of 20 KHz, we could observe only the first 10 harmonics (till 10 KHz). The 11th harmonic corresponding to 11kHz was not observed as it would occur at 11 KHz frequency and the sampling rate doesn't fulfil the Nyquist criterion for the 11th harmonic.

The higher harmonics are aliased and overlap on the original spectrum resulting in some amplitudes at the frequencies where ideally no component should have been present.

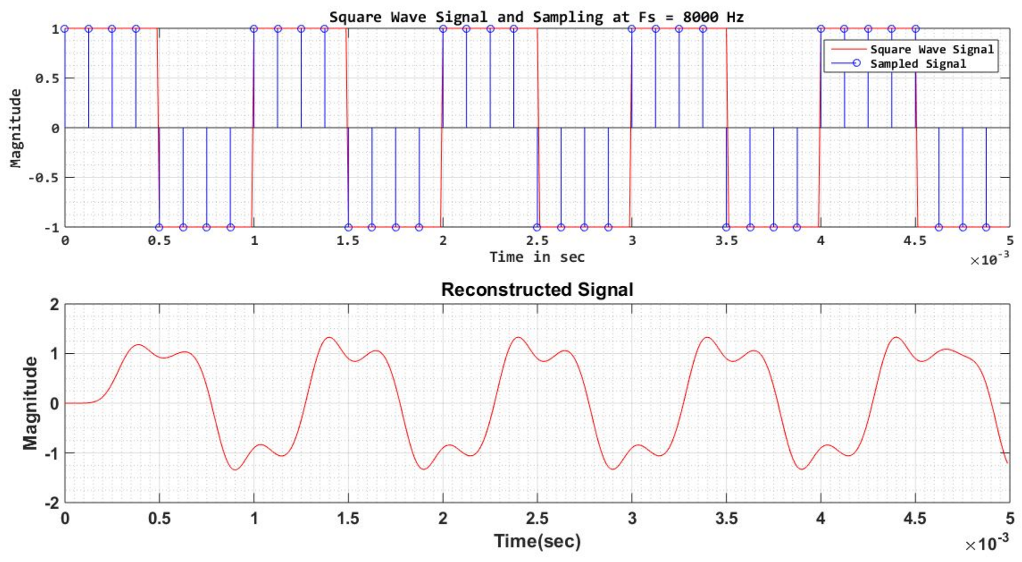
Part D

Problem Statement:

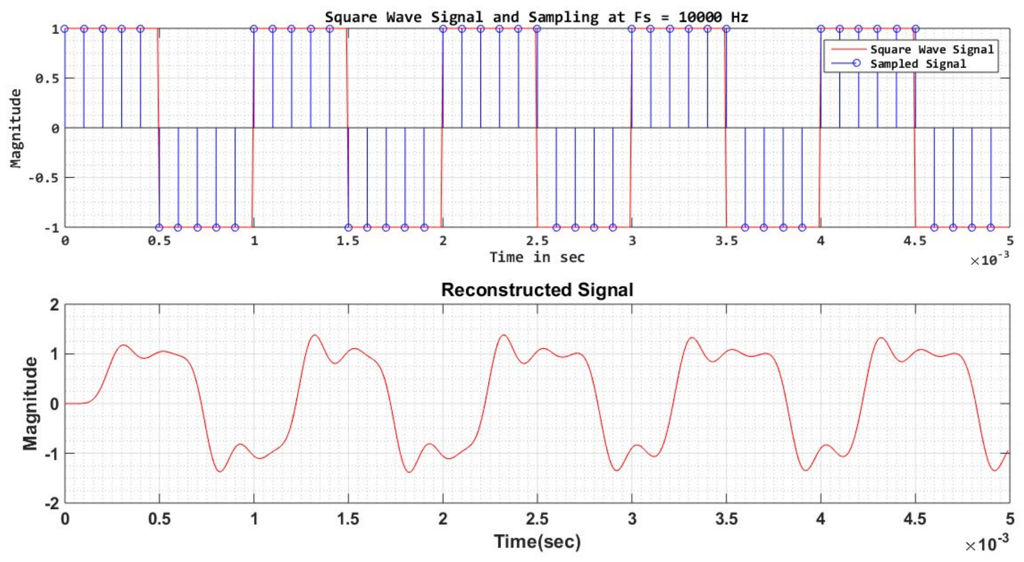
Sample the Square Wave signal at different sample frequencies and reconstruct the signal and compare the efficiency of reconstruction for different values of Fs.

Code and Observation:

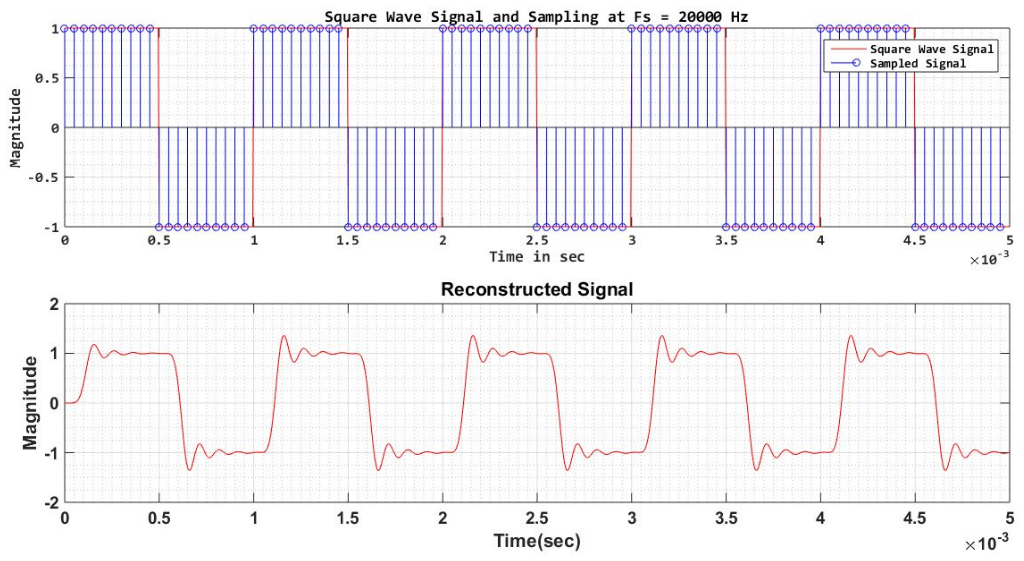
```
% Setting up a few parameters
% List for storing different values of frequency
fs = [8e3 10e3 20e3 40e3 60e3];
for z=1:length(fs)
    f = fs(z)*10;
    f_max = 1e3;
    Ncycle = 5;
    t = [0:1/f:Ncycle/f_max-1/f];
    % Analog Square wave generation
    sq = square(2*pi*f_max*t,50);
    fs1 = fs(z);
    % Forming the time axis
    ts = [0:1/fs1:Ncycle/f_max-1/fs1];
    xs = square(2*pi*f_max*ts,50);
    % Sample and Hold Signal
    j=1;
    % Upsampling
    t_new=upsample(ts,10);
    for i=1:length(t_new)

     xs_hold(i)=xs(j);

     for k=2:length(ts)
     if t_new(i)==ts(k)
     j=j+1;
     end
     end
    end

    % Using a Low Pass Filter to move out higher frequency spectrum
    figure();
    subplot(2,1,1);
    plot(t,sq,'r');
    hold on;
    stem(ts,xs,'b');
    grid on;
    grid minor;
    title(['Square Wave Signal and Sampling at Fs = ' num2str(fs(z)) ' Hz']);
    xlabel('Time in sec');
    ylabel('Magnitude');
    legend('Square Wave Signal','Sampled Signal');
    set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
    set(gcf,'color','w');

    %Using a Butterworth filter
    [b,a] = butter(10, 0.1,'low');

    % Reconstructed Signal
    Y_lpf = filter(b,a,xs_hold);
    subplot(2,1,2);
    plot(t,Y_lpf,'r');
    title('Reconstructed Signal');
    xlabel('Time(sec)');
    ylabel('Magnitude');

    % Setting Font Size, Font Name, Font weight and Background color
    set(gca, 'FontName', 'Arial','fontsize',14,'FontWeight','bold');
    set(gcf,'color','w');
    grid on;
    grid minor;

    % Calculating overshoot percentage
    over = max(Y_lpf);
    overshoot_percent = (over-1)*100;
    fprintf('Overshoot percentage for fs: %d Hz = %f\n',fs(z),overshoot_percent);
    % Using 2 percent settling time
    t_index_all = find(abs(Y_lpf)<=0.02);
    q = t_index_all(end-2);
    t_index=q;
    while(1)
     if(Y_lpf(q)>Y_lpf(q+1) && Y_lpf(q+2)<Y_lpf(q+1))
     if (Y_lpf(q+1)<=1.02)
     qnew = q+1;
     break;
     end
     end
     q=q+1;
    end
    settle_time= t(qnew)-t(t_index);
    % Printing the settling time
    fprintf('Settling time(2 percent) for fs: %d Hz = %.8f\n',fs(z),settle_time);
end
```



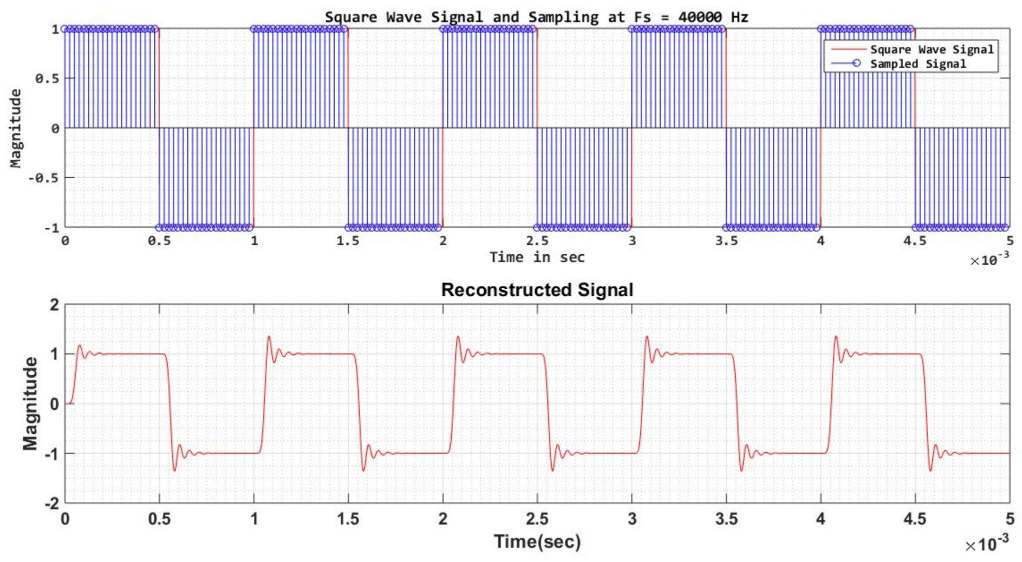
Analog Square Wave with Sampling Frequency 8 kHz and Corresponding Reconstructed Signal



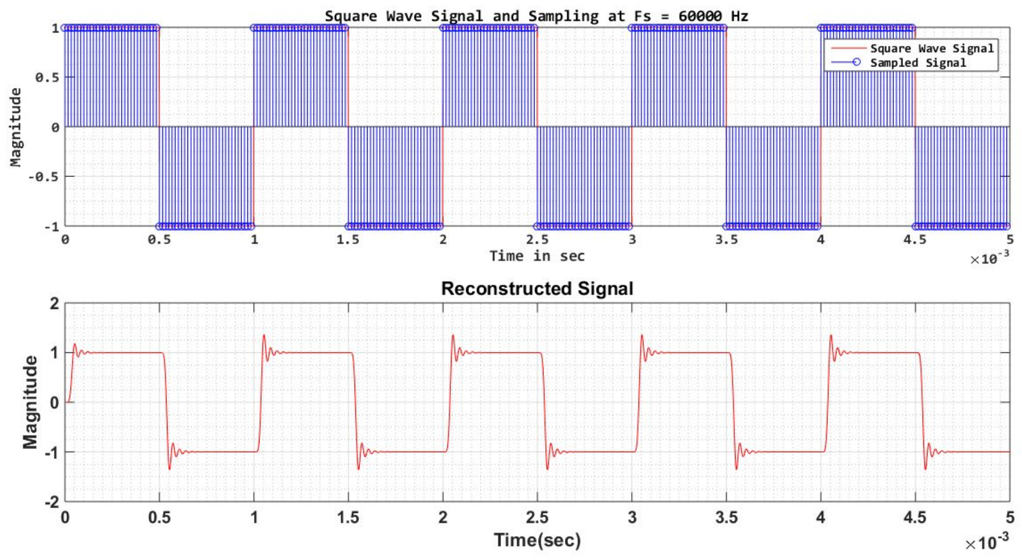
Analog Square Wave with Sampling Frequency 10 kHz and Corresponding Reconstructed Signal



Analog Square Wave with Sampling Frequency 20kHz and Corresponding Reconstructed Signal



Analog Square Wave with Sampling Frequency 40kHz and Corresponding Reconstructed Signal



Analog Square Wave with Sampling Frequency 60kHz and Corresponding Reconstructed Signal

Comparison Of Performance:

Overshoot percentage for fs: 8000 Hz = 33.268685
Settling time(2 percent) for fs: 8000 Hz = 0.00033750
Overshoot percentage for fs: 10000 Hz = 38.347958
Settling time(2 percent) for fs: 10000 Hz = 0.00001000
Overshoot percentage for fs: 20000 Hz = 36.050900
Settling time(2 percent) for fs: 20000 Hz = 0.00000500
Overshoot percentage for fs: 40000 Hz = 36.200743

```
Settling time(2 percent) for fs: 40000 Hz = 0.00000250
Overshoot percentage for fs: 60000 Hz = 36.201431
Settling time(2 percent) for fs: 60000 Hz = 0.00000167
```

Discussions:

We observed the reconstruction of the square wave from its samples for different sampling rates 8kHz, 10kHz, 20kHz, 40kHz, 60kHz. The performance measure of the reconstruction of the square wave were: **Overshoot percentage and the 2% settling time**. As the sampling rate is increased, we observe that the settling time goes on reducing. However, the overshoot percentage remains almost constant. This can be explained easily by Gibbs Phenomenon. So, the sampling rate must be chosen depending upon the allowable tolerance of settling time based on design specifications. For an ideal reconstruction we would need to include infinite harmonics of the signal which is not possible to realize on hardware.

Part E

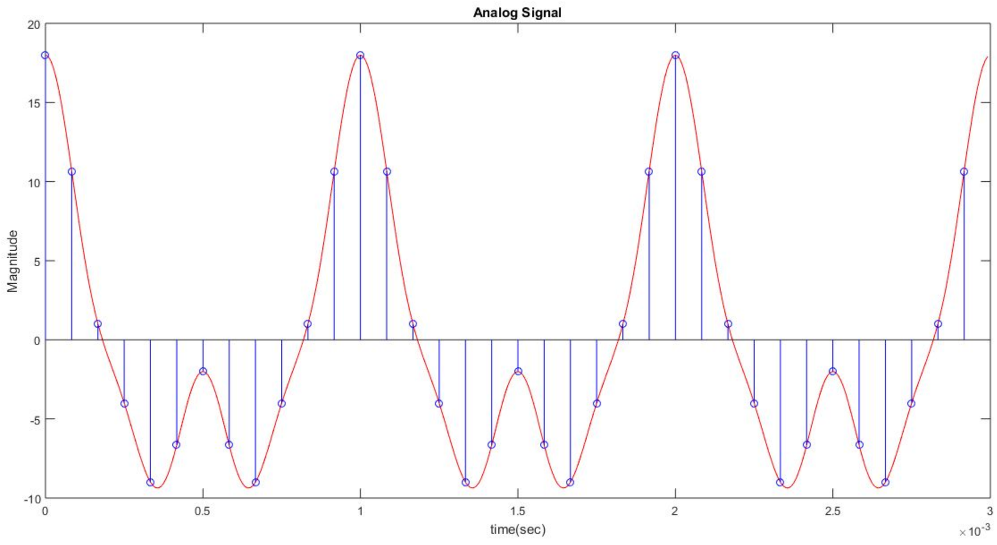
Problem Statement:

Perform interpolation of a signal sampled at 12 KHz by upsampling it by a factor of 2 and to compare both its time domain waveform and frequency domain spectrum with that of the signal when sampled at 24 KHz.

Code and Observation:

```
% Setting up a few parameters
f= 1.2e5;
f_max = 1e3;
cycles = 3;
t = [0:1/f:(cycles/f_max)-1/f];

% Plotting of Analog signal
x = 10*cos(2*pi*1000*t) + 6*cos(2*pi*2000*t) + 2*cos(2*pi*4000*t);
plot(t,x,'r');
hold on;
title('Analog Signal');
xlabel('time(sec)');
ylabel('Magnitude');

% Plotting of sampled signal 12 KHz
fs= 12000;
ts = [0:1/fs:cycles/f_max-1/fs];
xs = 10*cos(2*pi*1000*ts) + 6*cos(2*pi*2000*ts) + 2*cos(2*pi*4000*ts);
```

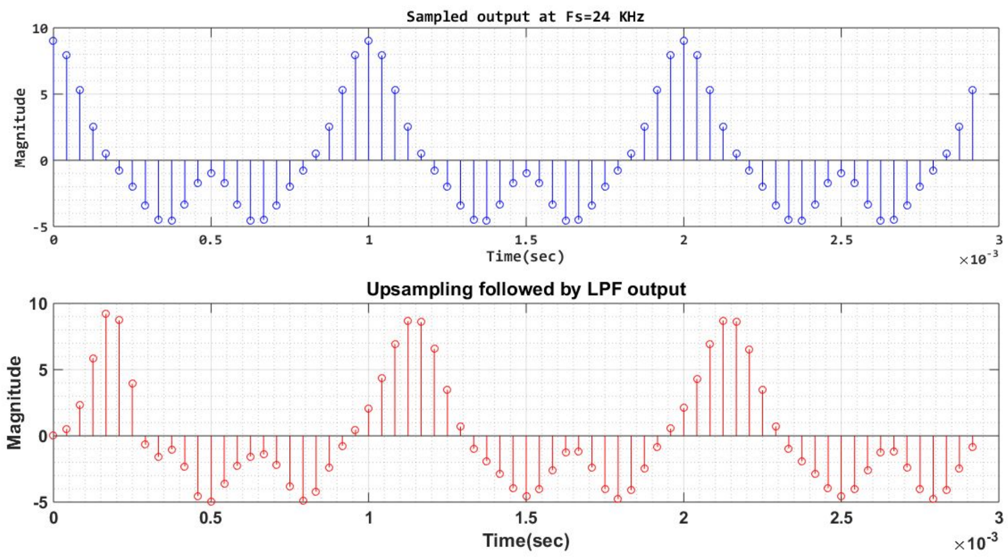
```
stem(ts,xs,'b');
```



Analog Signal And Sampled at 12 kHz

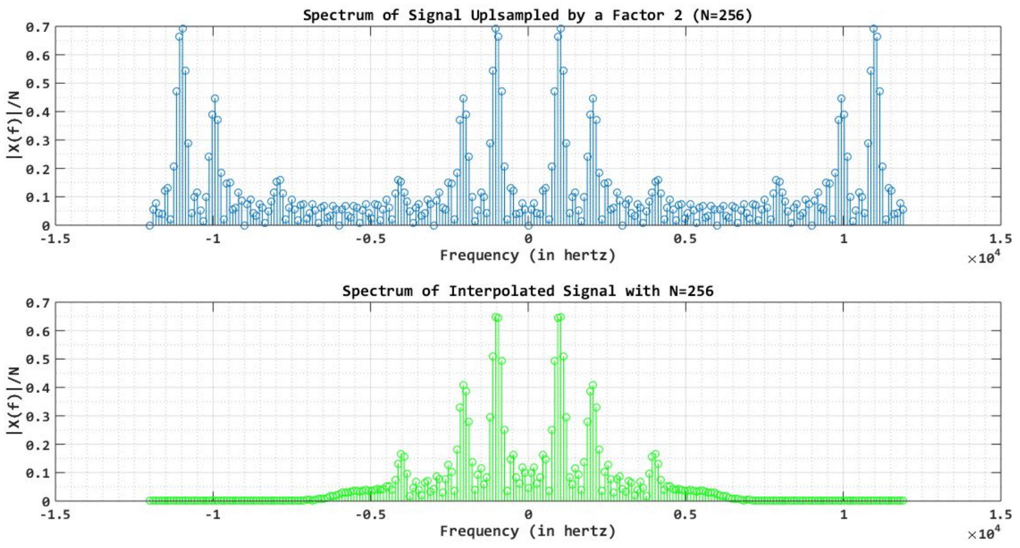
```
% Sampling at 24 KHz
fs2= 24000;
ts2 = [0:1/fs2:cycles/f_max-2/fs2];
xs2 = 10*cos(2*pi*1000*ts2) + 6*cos(2*pi*2000*ts2) + 2*cos(2*pi*4000*ts2);
figure();
% Plotting of sampled signal 24 KHz
subplot(2,1,1);
stem(ts2,xs2/2,'b');
title('Sampled output at Fs=24 KHz');
xlabel('Time(sec)');
ylabel('Magnitude');
grid on;
grid minor;

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');

% Upsampling Process
xs_up = zeros(1,2*(length(ts)-1));
for i=1:length(ts)
 xs_up(2*i-1) = xs(i);
end
% Butterworth Low Pass Filtering to remove high frequency components
[b,a] = butter(10, 6000/(24000/2),'low');
Y_lpf = filter(b,a,xs_up);
subplot(2,1,2);
stem(ts2,Y_lpf,'r');
title('Upsampling followed by LPF output');
xlabel('Time(sec)');
ylabel('Magnitude');
grid on;
grid minor;
```

```
% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Arial','fontsize',14,'FontWeight','bold');
set(gcf,'color','w');
```



Sampled Output at Fs=24 kHz and Upsampled followed by LPF Filter

```
% Performing Discrete Time Fourier Transform
N = 256;
y_dft1 = fft(xs_up,N);
freq = [-fs2/2:fs2/N:fs2/2-fs2/N];
y_dft_val1 = fftshift(abs(y_dft1)/N);
figure();
subplot(2,1,1);
stem(freq,y_dft_val1);
xlabel('Frequency (in hertz)');
ylabel('|X(f)|/N');
title('Spectrum of Signal Upsampled by a Factor 2 (N=256)');
grid on;
grid minor;

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');
y_dft2 = fft(Y_lpf,N);
freq = [-fs2/2:fs2/N:fs2/2-fs2/N];
y_dft_val2 = fftshift(abs(y_dft2)/N);
subplot(2,1,2);
stem(freq,y_dft_val2,'g');
xlabel('Frequency (in hertz)');
ylabel('|X(f)|/N');
title('Spectrum of Interpolated Signal with N=256');
grid on;
grid minor;

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');
```



Upsampling by a factor of 2 and Spectrum of Interpolated Signal

Discussions:

We can clearly see, that the time domain interpolated signal that is, upsampling followed by low pass filtering matches with the signal sampled at twice the initial sample rate. However, there is a certain phase shift between these two signals which is introduced by the passive components of the low pass filter which induces delay in the interpolated signal.

After upsampling, when the output is passed through an interpolation filter which is basically a low pass filter with cut-off frequency 6 KHz , so as to obtain only one part of the spectrum in the range -$F_s/2$ to $F_s/2$.

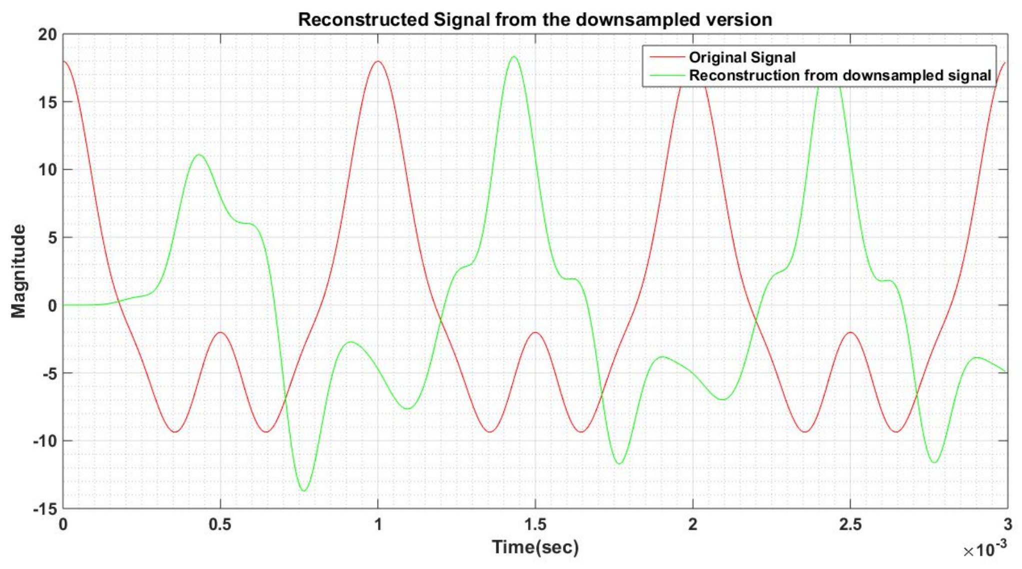
Part F

Problem Statement:

Perform decimation of a signal sampled at 12 KHz by downsampling it by a factor of 2 and to observe the frequency domain spectrum of the downsampled signal in two cases – with and without the anti-aliasing filter. Also to perform signal reconstruction from the decimated signal.

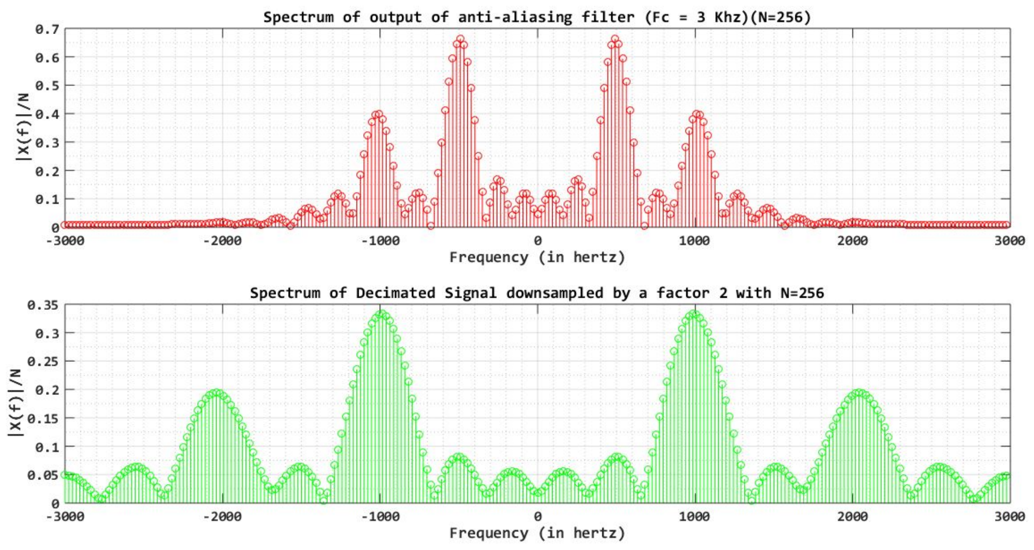
Code and Observation:

```
% Setting up a few parameters
f= 1.2e5;
f_max = 1e3;
cycles = 3;
% Defining time axis
t = [0:1/f:(cycles/f_max)-1/f];
% Defining the analog signal
x = 10*cos(2*pi*1000*t) + 6*cos(2*pi*2000*t) + 2*cos(2*pi*4000*t);
% Sampling at 12 kHz
fs= 12000;
ts = [0:1/fs:cycles/f_max-1/fs];
% Forming the sampled signal
xs = 10*cos(2*pi*1000*ts) + 6*cos(2*pi*2000*ts) + 2*cos(2*pi*4000*ts);
% Anti-aliasing filter
[b,a] = butter(6,3000/(12000/2),'low');
x_alias = filter(b,a,xs);
% Downsampling by a factor 2
xs_down = downsample(x_alias,2);
ts_down = downsample(ts,2);
% Downsampling without anti-aliasing filter
xs_down_wa = downsample(xs,2);
% Sample and hold of Downsampled Signal
j=1;
ts_hold = upsample(ts_down,20);
for i=1:length(t)

 xs_hold(i)=xs_down(j);

 for k=2:length(ts_down)
     if ts_hold(i)==ts_down(k)
        j=j+1;
     end
 end
end

% Low Pass Filtering with Butterworth filter
[b,a] = butter(10, 0.1,'low');
Y_lpf = filter(b,a,xs_hold);
figure();
plot(t,x,'r');
hold on;
plot(t,Y_lpf,'g');
title('Reconstructed Signal from the downsampled version');
xlabel('Time(sec)');
ylabel('Magnitude');
legend('Original Signal','Reconstruction from downsampled signal');
grid on;
grid minor;

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Arial','fontsize',14,'FontWeight','bold');
set(gcf,'color','w');
```



Reconstruction Signal from Downsampled Version

```
% Discrete Time Fourier Transform
N = 256;
y_dft1 = fft(x_alias,N);
freq = [-0.5*fs/2:0.5*fs/N:0.5*fs/2-0.5*fs/N];
y_dft_val1 = fftshift(abs(y_dft1)/N);
figure();
subplot(2,1,1);
stem(freq,y_dft_val1,'r');
xlabel('Frequency (in hertz)');
ylabel('|X(f)|/N');
title('Spectrum of output of anti-aliasing filter (Fc = 3 Khz)(N=256)');
grid on;
grid minor;

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');
y_dft2 = fft(xs_down,N);
freq = [-0.5*fs/2:0.5*fs/N:0.5*fs/2-0.5*fs/N];
y_dft_val2 = fftshift(abs(y_dft2)/N);
subplot(2,1,2);
stem(freq,y_dft_val2,'g');
xlabel('Frequency (in hertz)');
ylabel('|X(f)|/N');
title('Spectrum of Decimated Signal downsampled by a factor 2 with N=256');
grid on;
grid minor;

% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');
```



Spectrum of output of Anti aliasing filter and Decimated Signal

Discussions:

We observed the effect of aliasing on the frequency spectrum of a signal after downsampling. We attempted to reconstruct the signal from the down-sampled version of the signal and as we used an anti-aliasing filter of cut-off frequency 3 KHz, the 4 KHz sinusoid present in the original analog signal got missed in the reconstruction. Hence, on downsampling, we have an advantage of reducing hardware complexity as we are sampling at a lower rate. However, this eliminates the higher frequency components. Downsampling is useful only when the higher frequency elements are not very important.

Part G:

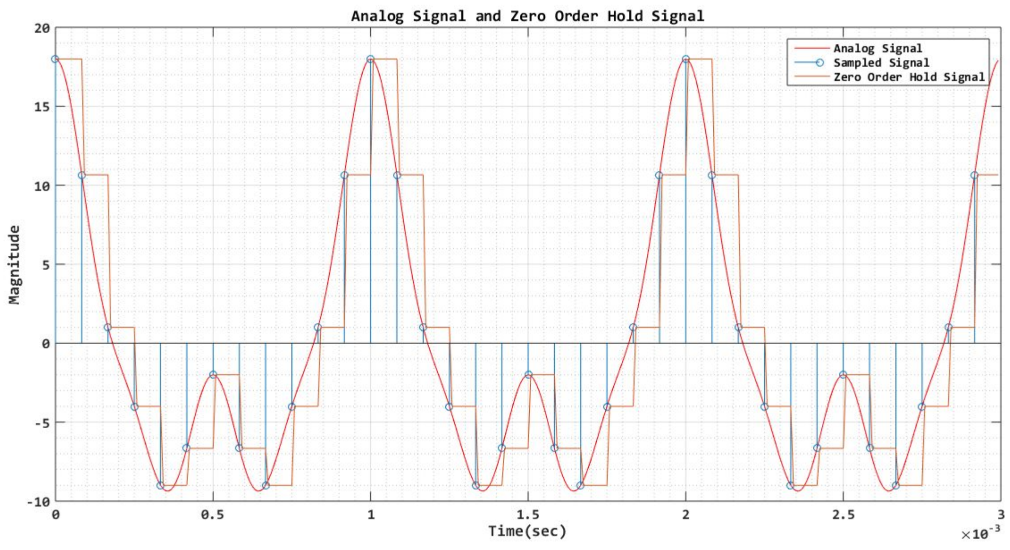
Problem Statement:

Input an analog signal and perform Sample and Hold (ZOH) followed by low pass filtering for signal reconstruction.

Code and Observations

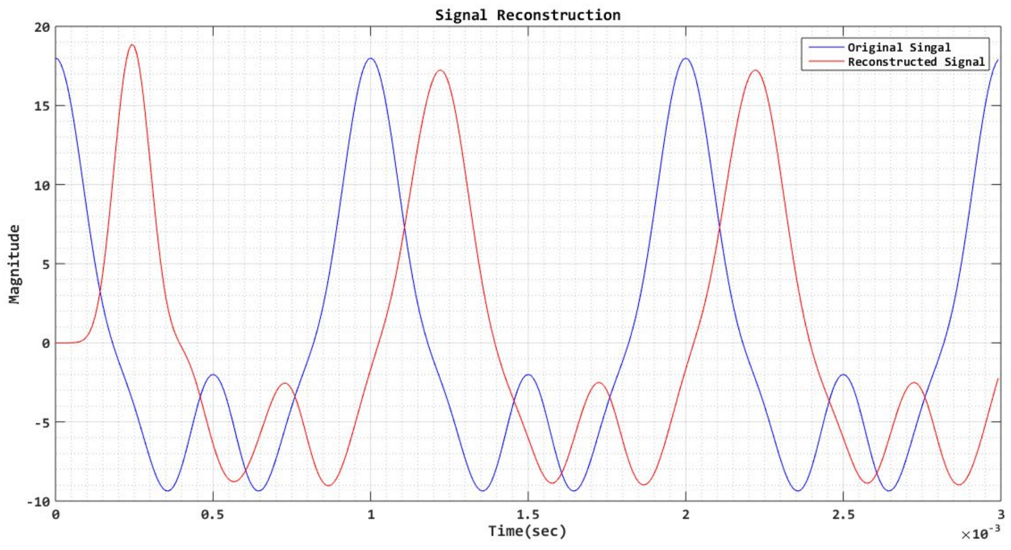
```
% Setting up a few parameters
```

```
f= 1.2e5;
f_max = 1e3;
cycles = 3;
t = [0:1/f:(cycles/f_max)-1/f];
% Defining the analog signal
x = 10*cos(2*pi*1000*t) + 6*cos(2*pi*2000*t) + 2*cos(2*pi*4000*t);
% Plotting of Main signal
plot(t,x,'r');
hold on;
title('Analog Signal and Zero Order Hold Signal');
xlabel('Time(sec)');
ylabel('Magnitude');
grid on;
grid minor;
% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');

% Sampled signal
fs= 12000;
ts = [0:1/fs:cycles/f_max-1/fs];
% Sampled Signal Created
xs = 10*cos(2*pi*1000*ts) + 6*cos(2*pi*2000*ts) + 2*cos(2*pi*4000*ts);
% Sample and hold operation
j=1;
t_new=upsample(ts,10);
for i=1:length(t_new)

 xs_hold(i)=xs(j);

 for k=2:length(ts)
     if t_new(i)==ts(k)
        j=j+1;
     end
 end
end
```



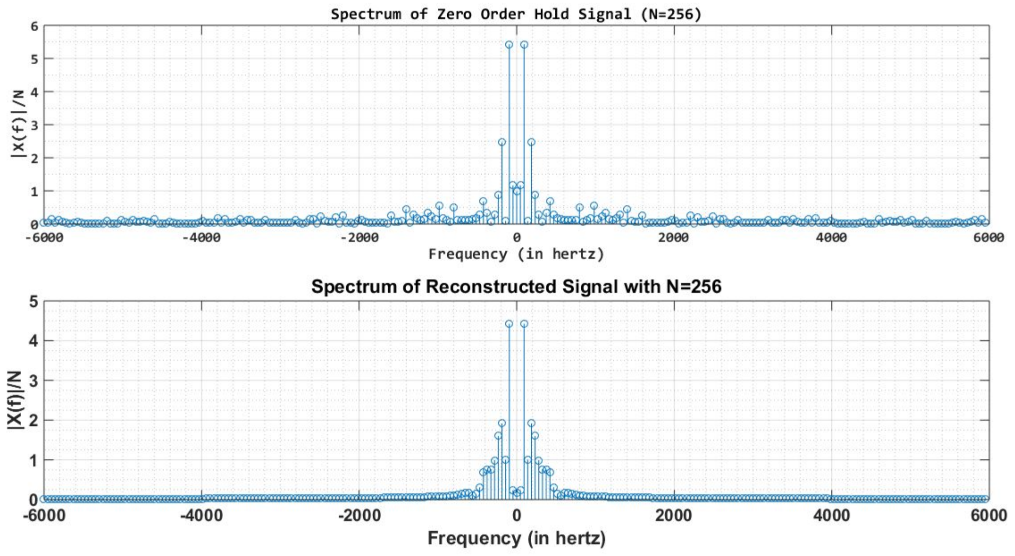
Analog Signal with Zero Order Hold Signal

```
% Low pass filtering using butterworth filter
stem(ts,xs);
hold on;
plot(t,xs_hold);
legend('Analog Signal','Sampled Signal','Zero Order Hold Signal');
[b,a] = butter(10, 0.1,'low');
Y_lpf = filter(b,a,xs_hold);
figure();
plot(t,x,'b');
hold on;
plot(t,Y_lpf,'r');
title('Signal Reconstruction');
xlabel('Time(sec)');
ylabel('Magnitude');
grid on;
grid minor;
% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');
legend('Original Signal','Reconstructed Signal');
```



Original Signal and Reconstructed Signal

```
% Discrete Fourier Transform
N = 256;
y_dft1 = fft(xs_hold,N);
freq = [-fs/2:fs/N:fs/2-fs/N];
y_dft_val1 = fftshift(abs(y_dft1)/N);
figure();
subplot(2,1,1);
stem(freq,y_dft_val1);
xlabel('Frequency (in hertz)');
ylabel('|X(f)|/N');
title('Spectrum of Zero Order Hold Signal (N=256)');
grid on;
grid minor;
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w');
y_dft2 = fft(Y_lpf,N);
freq = [-fs/2:fs/N:fs/2-fs/N];
y_dft_val2 = fftshift(abs(y_dft2)/N);
subplot(2,1,2);
stem(freq,y_dft_val2);
xlabel('Frequency (in hertz)');
ylabel('|X(f)|/N');
title('Spectrum of Reconstructed Signal with N=256');
grid on;
grid minor;
% Setting Font Size, Font Name, Font weight and Background color
set(gca, 'FontName', 'Consolas','fontsize',12,'FontWeight','bold');
set(gcf,'color','w')
```



Spectrum of ZOH and Reconstructed Signal

Discussions:

In this part we reconstruct an analog waveform from its samples after sampling
it above the Nyquist rate, after LPF is done the sampled and held i.e., **ZOH done**
signal. We can observe that the zero order hold signal has almost the same frequency content as the originally sampled signal. When the ZOH signal is passed through a low pass filter, the original signal is reconstructed. However, we observe some phase shift due to the the presence of passive components in the filter.

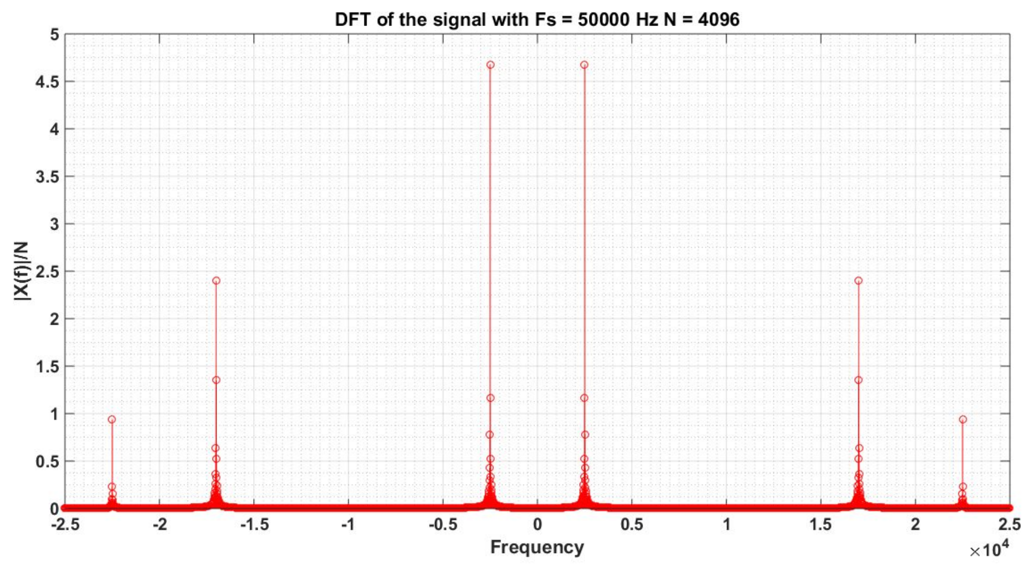
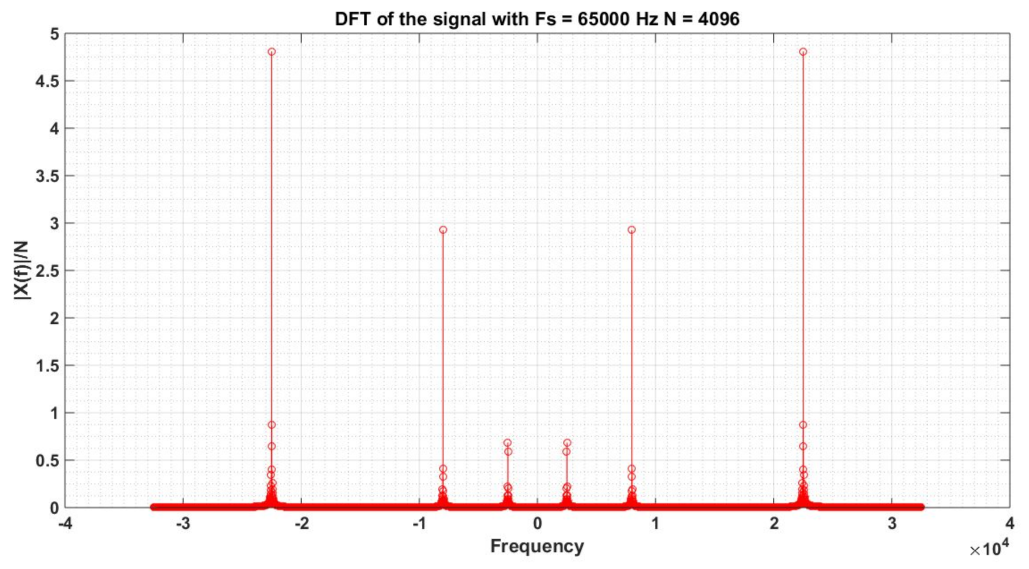
Part H:

Problem Statement:

Input a Bandpass Signal and perform bandpass sampling and to show DFT for different sampling frequencies.

Code and Observations:

```
% Setting up a few parameters
f_max= 1.2e8;
f_min = 1000*10702.5;
N=4096;
t = [0:1/f_max:(N/f_min)];
% Defining the bandpass analog signal
x = 10*cos(2*pi*1000*10702.5*t)+6*cos(2*pi*10717*1000*t)+2*cos(2* pi*10727.5*1000*t);
% List for two frequencies
fs = [50000 65000];
for i=1:length(fs)
    ts = [0:1/fs(i):(N-1)/fs(i)];
    xs = 10*cos(2*pi*1000*10702.5*ts)+6*cos(2*pi*10717*1000*ts)+2*cos(2*pi*10727.5*1000*ts);
    f_range = [-fs(i)/2:fs(i)/N:fs(i)/2-fs(i)/N];
    y_fft = fftshift(abs(fft(xs)/N));
    figure();
    stem(f_range,y_fft,'r');
    xlabel('Frequency');
    ylabel('|X(f)|/N');
    title(['DFT of the signal with Fs = ' num2str(fs(i)) ' Hz N = 4096']);
    % Setting Font Size, Font Name, Font weight and Background color
    set(gca, 'FontName','Arial','fontsize',14,'FontWeight','bold');
    set(gcf,'color','w');
    grid on;
    grid minor;
end
```


DFT Spectrum of Signal at 50kHz

DFT Spectrum of Signal at 65kHz

Discussions:

We input a bandpass signal of bandwidth 25 KHz.. For sampling of bandpass signals, it is sufficient to sample at a rate twice the bandwidth of the signal to facilitate reliable reconstruction.We observed that for different sampling rates, we obtain either a positive or a negative (flipped) image of the original spectrum. For Fs = 50 KHz, we obtained a positive image while for Fs = 65 KHz, we obtained a flipped image of the spectrum.