CS306: Data Analysis and Visualization

Lab 1: Report

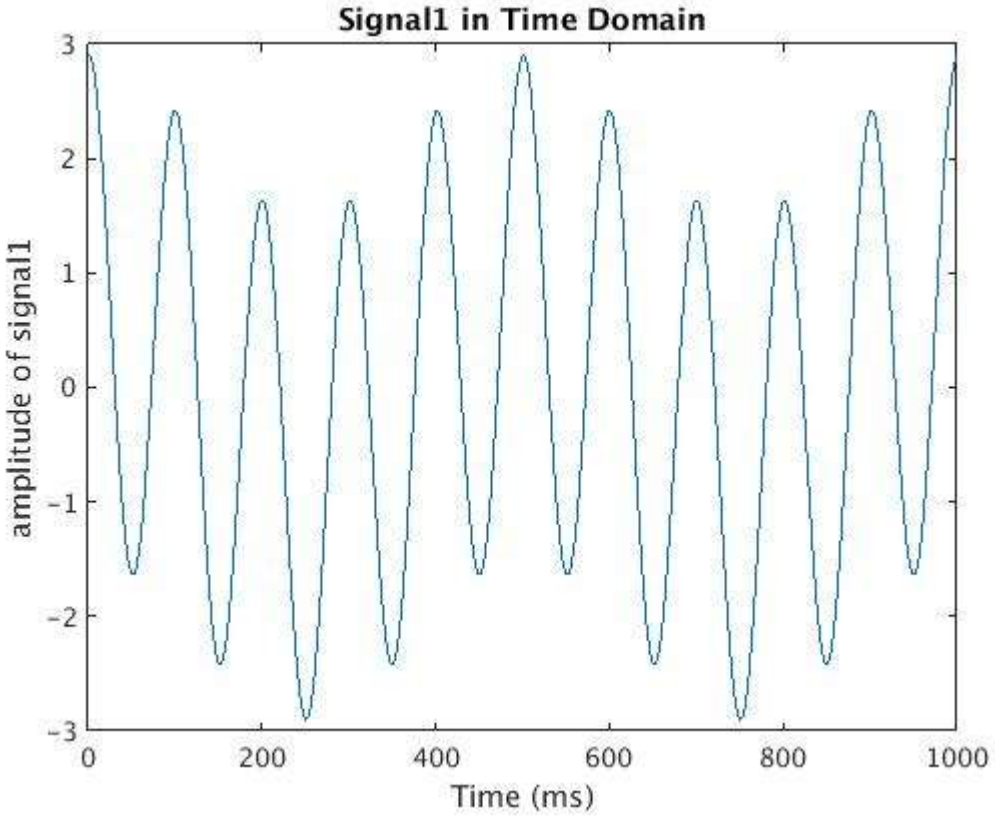
Rajdeep Pinge 201401103

Aditya Joglekar 201401086

## Objective: To study the effect of sampling in the domain of signal processing

## Experiment 1: Analysis of given vector 'signal1' of length 1 second formed by linear combinations of two sinusoids of frequencies f1 and f2.

Plot of the given vector

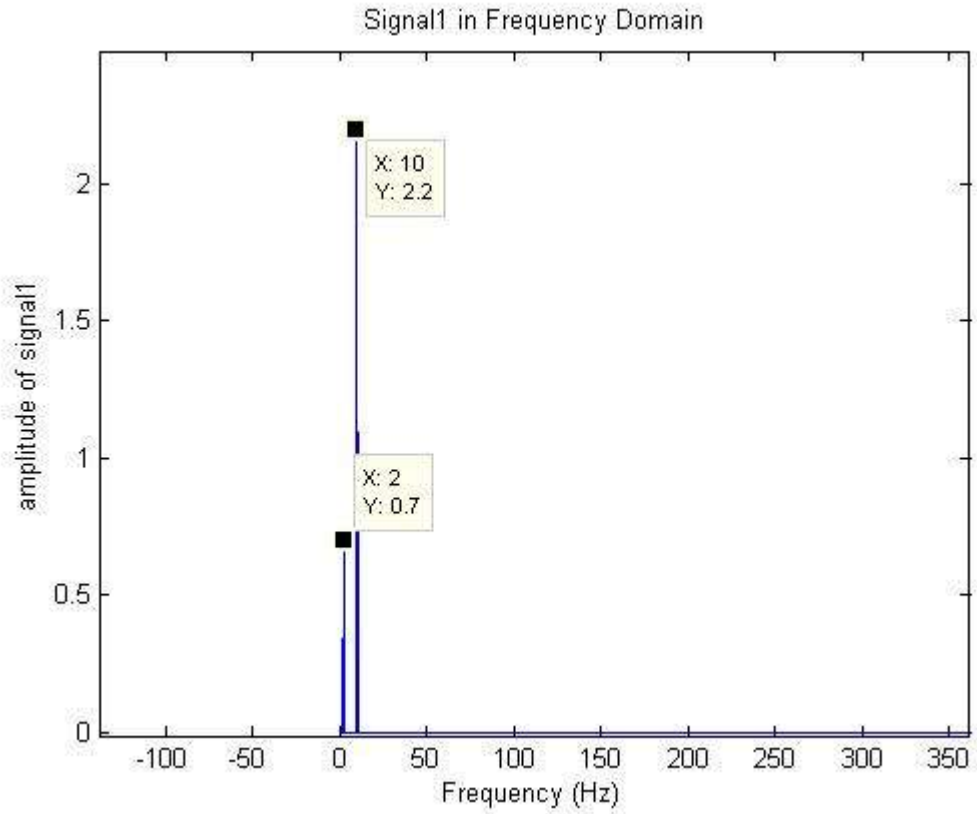



The above graph shows that the signal is made up of two sinusoids. See the wavy nature of the visible sinusoid to see the other sinusoidal wave.

1. **Determine f1, f2 and the Nyquist rate.**

   To determine the frequencies f1 and f2, we need to perform the Fourier transform of the time domain vector signal which gives us the frequency domain signal.

There is an in-built function in the MATLAB to calculate Fast Fourier Transform. We use this function to find the one-sided frequency spectrum of the signal. This gives us the following graph.




From the above figure the two spikes show that the two frequencies which make the given signal are

Freq f1 = 2 Hz

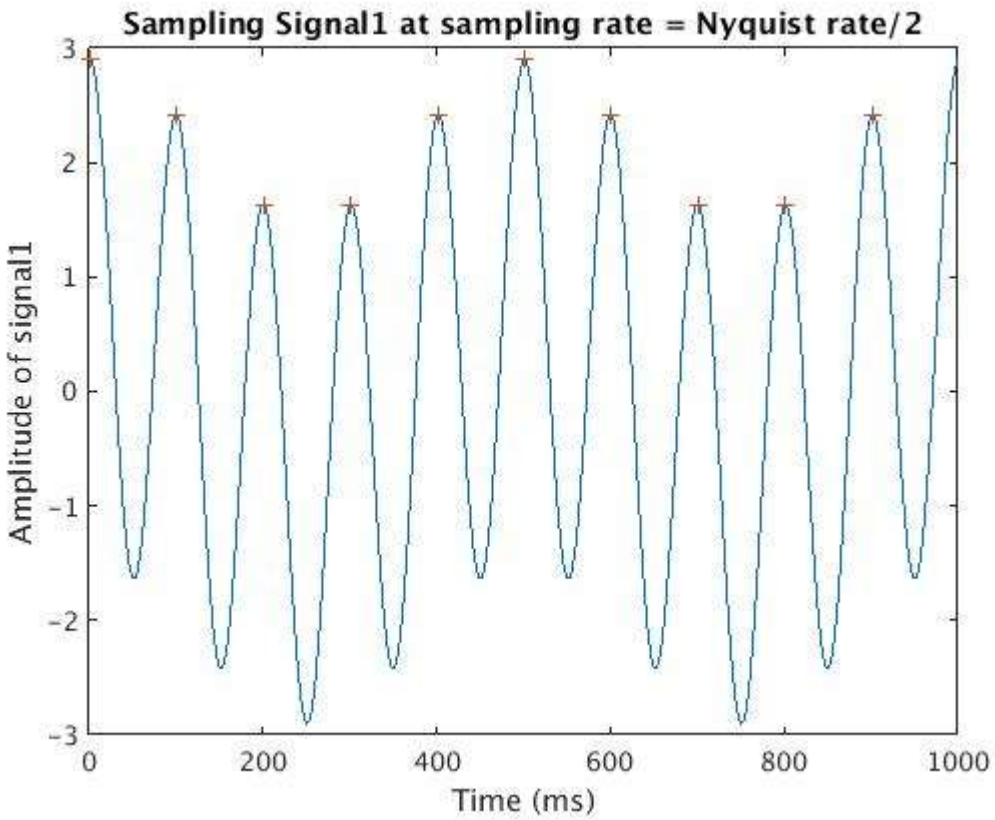
Freq f2 = 10 Hz

Nyquist Rate = 2 * max(f1, f2) = 2 * max(2, 10) = 2 * 10 = 20 Hz

2. **Sample the given signal <u>uniformly</u> at 3 different rates: In all the 3 cases, show a plot with sampled points overlaid on the original signal.**

   **(a) half the Nyquist rate = 20 / 2 = 10 Hz**

   For the 1 second signal, at this sampling rate there will be 10 data sample points. Furthermore, as this is uniform sampling, the 10 data points will be equally space from each other and will be uniformly distributed over the entire range of the signal. These are shown in the below graph.

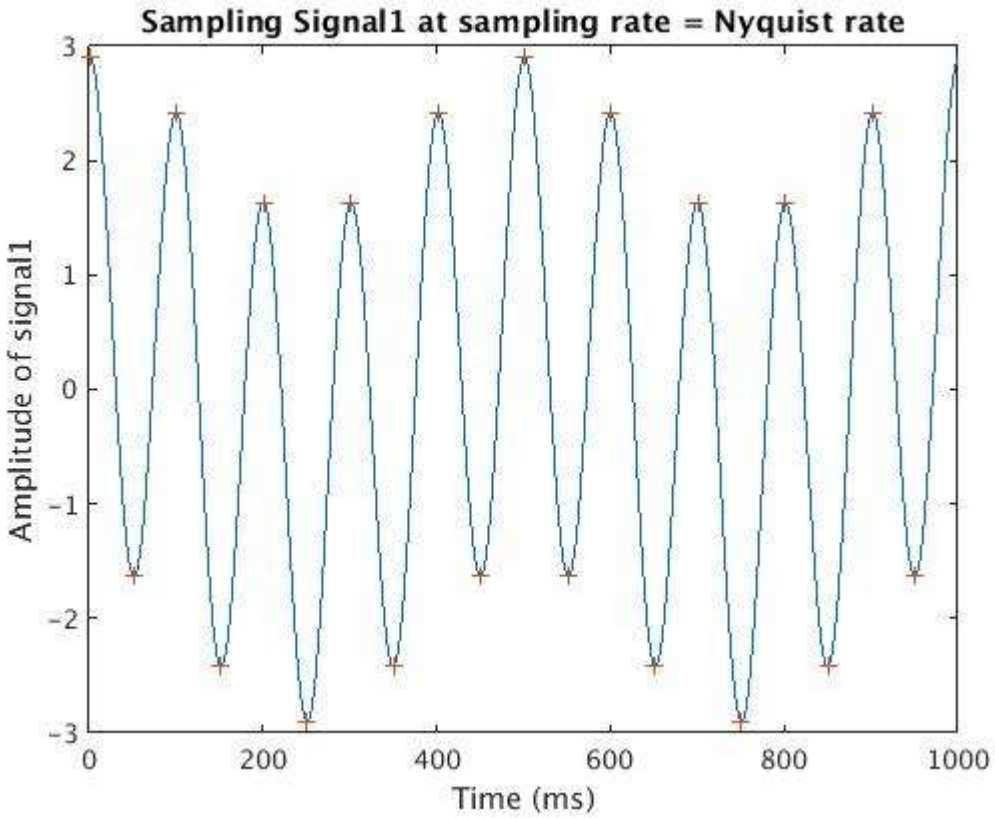



The '+' signs in the above graph represent the uniformly sampled values overlaid on the original signal.

Incidentally, we have got all the peaks in each cycle as the sample points. But this is just a coincidence. If we shift the starting sample by some amount, then the further samples will vary accordingly.

**(b) at Nyquist rate = 20 Hz**

Here for the 1 second signal, at this sampling rate there will be 20 data sample points.

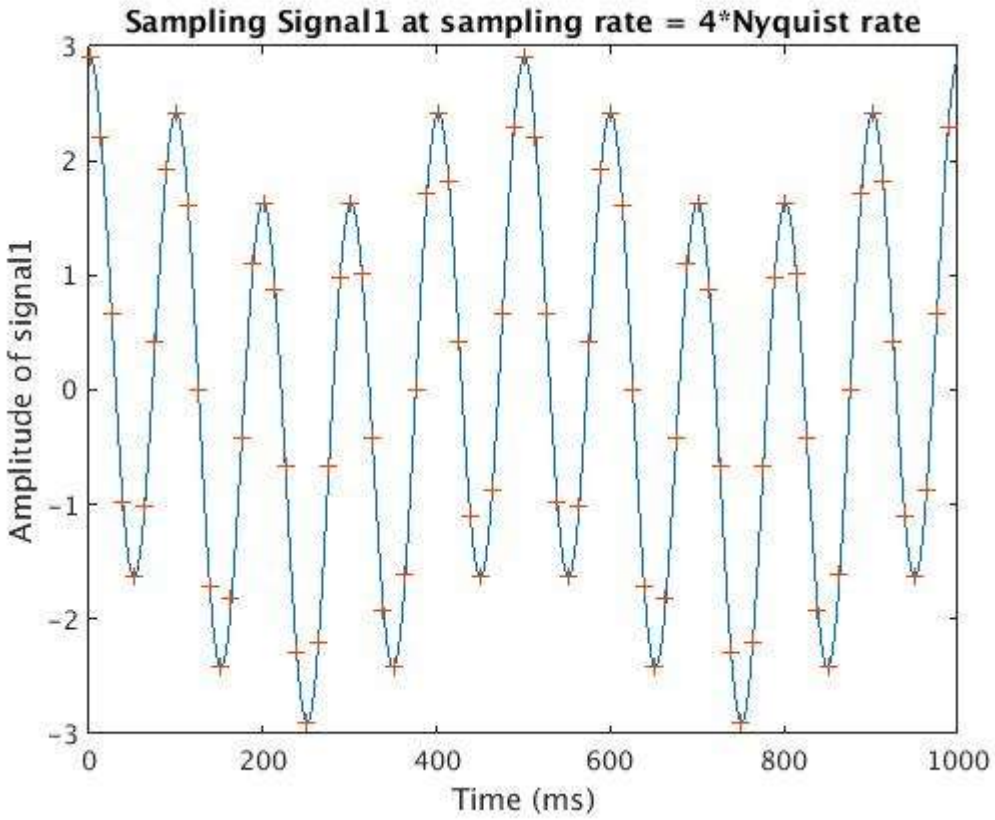



The above graph shows the 20 sampled data points.

Again the maximum and minimum points sampled in each cycle is just a coincidence. If we shift the start point, the rest of the points will change.

**(c) 4 times Nyquist rate = 20 * 4 = 80 Hz**

Here for the 1 second signal, at this sampling rate there will be 80 data sample points.

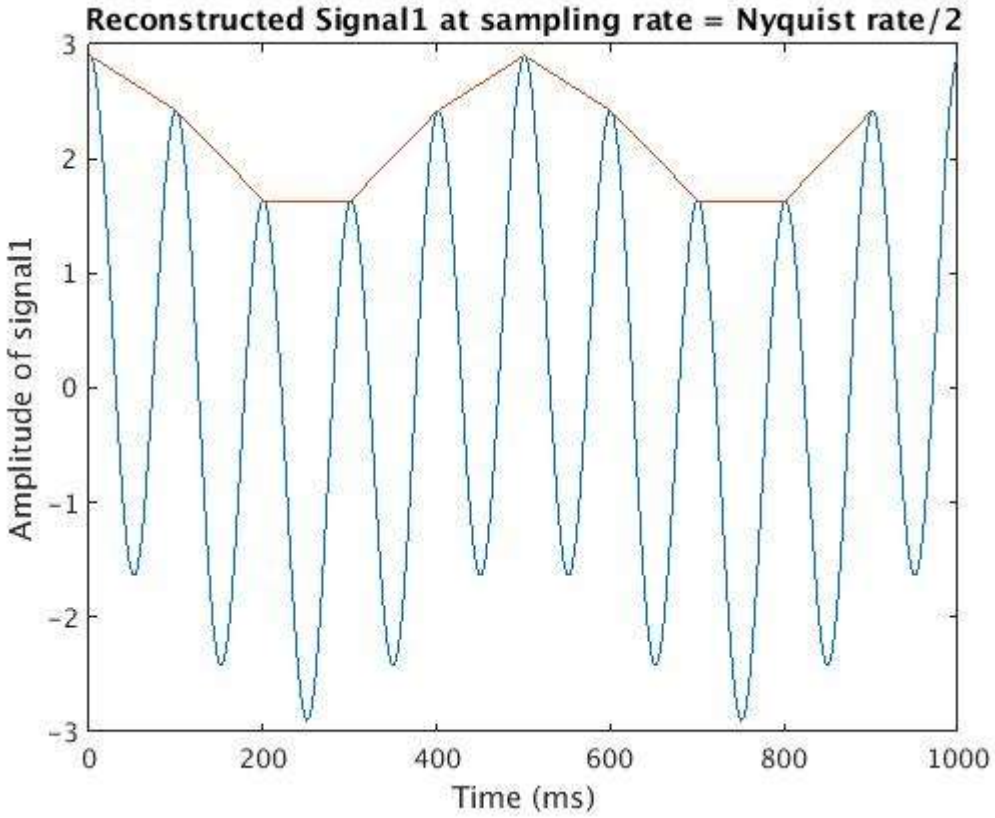



The above graphs shows the 80 sample points.
These points more uniformly and more densely cover the entire range of the signal

**3. Continuing from previous part, reconstruct the original signal from the sampled versions using linear interpolation. In each of the 3 cases, compute the reconstruction error (mean squared error MSE). Hint: use interp1 command for interpolation.**

For generating a continuous signal from the above sampled points we need to perform interpolation. As per the Hint, we have used the interp1 command to perform interpolation. We use the simplest form of interpolation i.e. linear interpolation. The same function can be used for some other types of interpolations as well.

**(a) half the Nyquist rate = 20 / 2 = 10 Hz**




The above graph shows the nature of the reconstructed signal. This shows that for this sampling rate there is a lot of error in the reconstructed signal.

Note again that if the starting sample is different, the rest of the samples will also change but the overall nature of the graph would remain the same.

**Finding Mean Squared Error:**

Observe that we don't have a sample towards the end of the graph. Therefore, interpolation cannot be performed here. For this part the reconstructed signal in MATLAB gives value NaN: Not a Number. Which means, we cannot take this part for calculating the error.

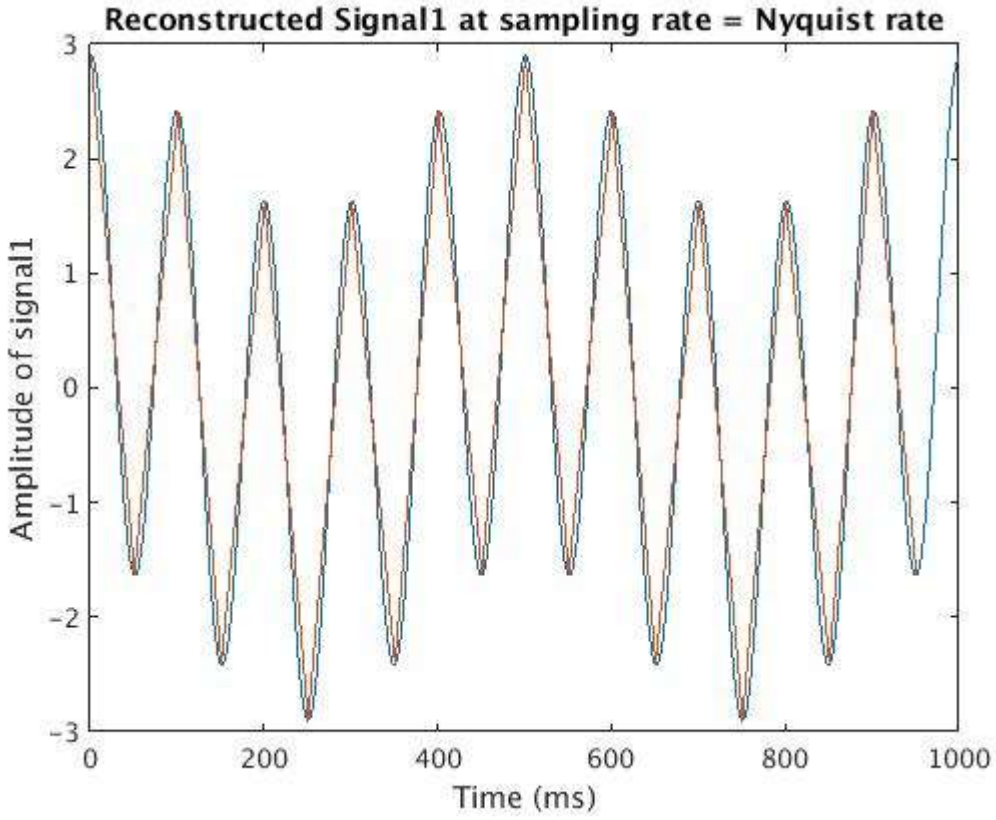
Solution: We have used isnan() MATLAB function to check if the reconstructed value is valid and we find MSE only for these valid points

In this case, we get 901 valid points using which the above red line signal has been reconstructed.

Mean Squared Error (MSE) = 7.3022

This implies that at each point, the square if the error averages 7.3022. This error is significant as we had determined earlier from the observation because, the max amplitude of the signal is 2.9

**(b) at Nyquist rate = 20 Hz**



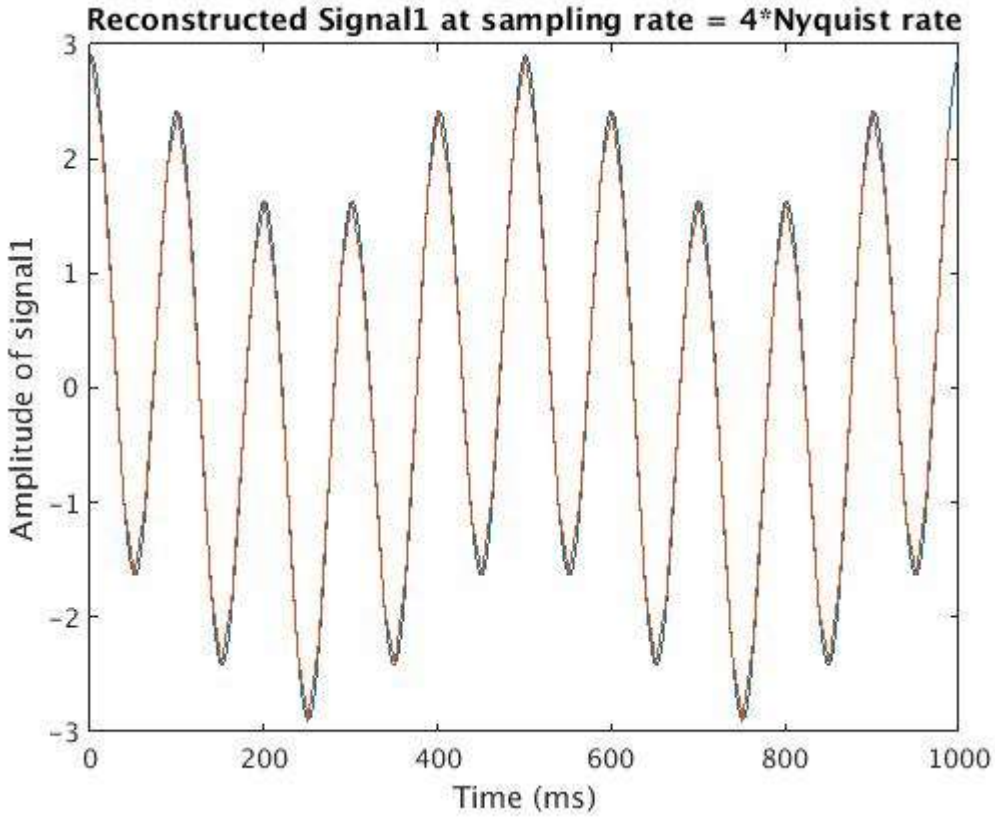

The red line in the above graph is a much better reconstruction of the original signal represented by the blue line. This error may increase if the sample points are changed but the MSE would still be much less than the previous case. Even if the sample points are changed, the nature of the reconstructed signal would be similar to the original signal.

Mean Squared Error (MSE) = 0.1103

This implies that at each point the error is much less than 1 and the reconstructed signal closely approximates the original signal.

**(c) 4 times Nyquist rate = 20 * 4 = 80 Hz**

Theoretically, since case has more sampled points, this reconstructed signal should be closer to the original signal.




The red and blue lines in the above graph almost overlap indicating high accuracy of the reconstructed signal.

Mean Squared Error (MSE) = 0.0083

This is very small verifying the observation that the reconstructed signal is almost a replica of the original signal.

**4. QA. Which sampling rate results in highest error and why?**

By observing the above three cases it is evident that the highest error is in the case where sampling rate is half of Nyquist rate. Theoretically also, we need to sample the signal by at least the Nyquist rate so as to get nearly accurate signal. For Nyquist rate / 2, it would not give the accurate result.

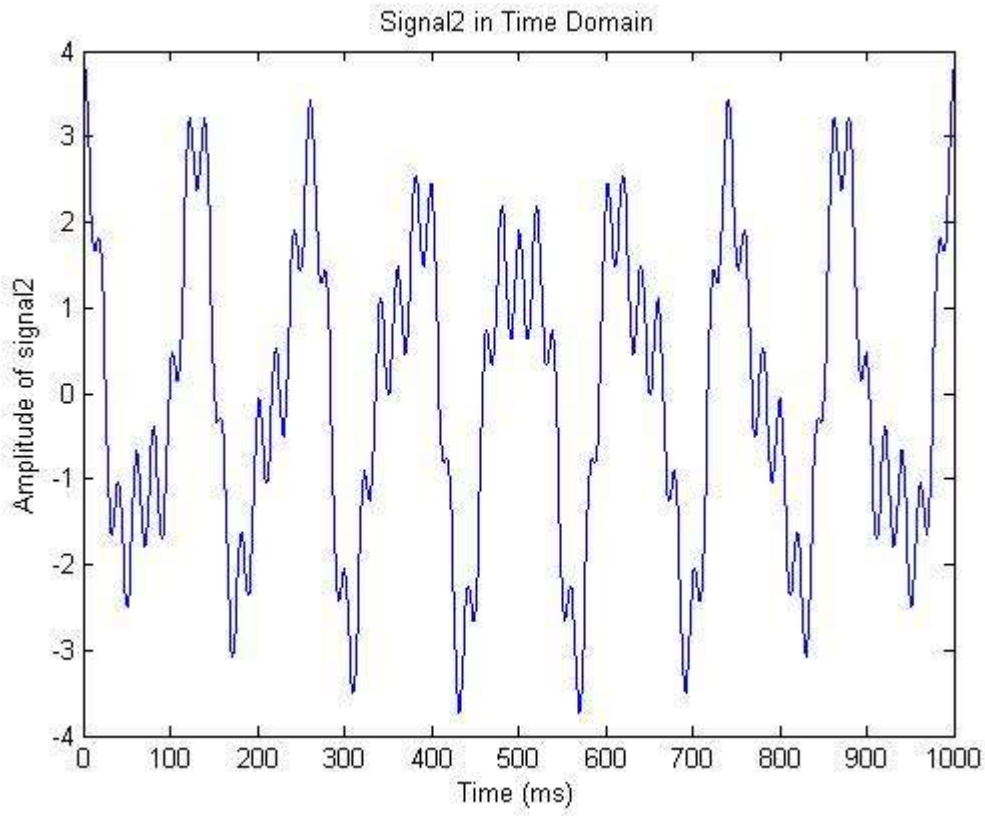
As the number of samples increase the MSE decreases. This is because, more samples approximate the original signal more accurately.

**QB. Do you observe perfect reconstruction (error = 0) at Nyquist rate? if not what are the possible reasons?**

We don't get perfect reconstruction at the Nyquist rate. This is because, Nyquist rate is just the limiting case to get the reconstructed signal to have the same nature as the original signal. It is not necessary and in general, it would be very difficult to obtain zero error at Nyquist rate except for some special cases.
Another factor might be the discreteness of the original signal which we are assuming to be continuous, though this might not be as much a significant factor.

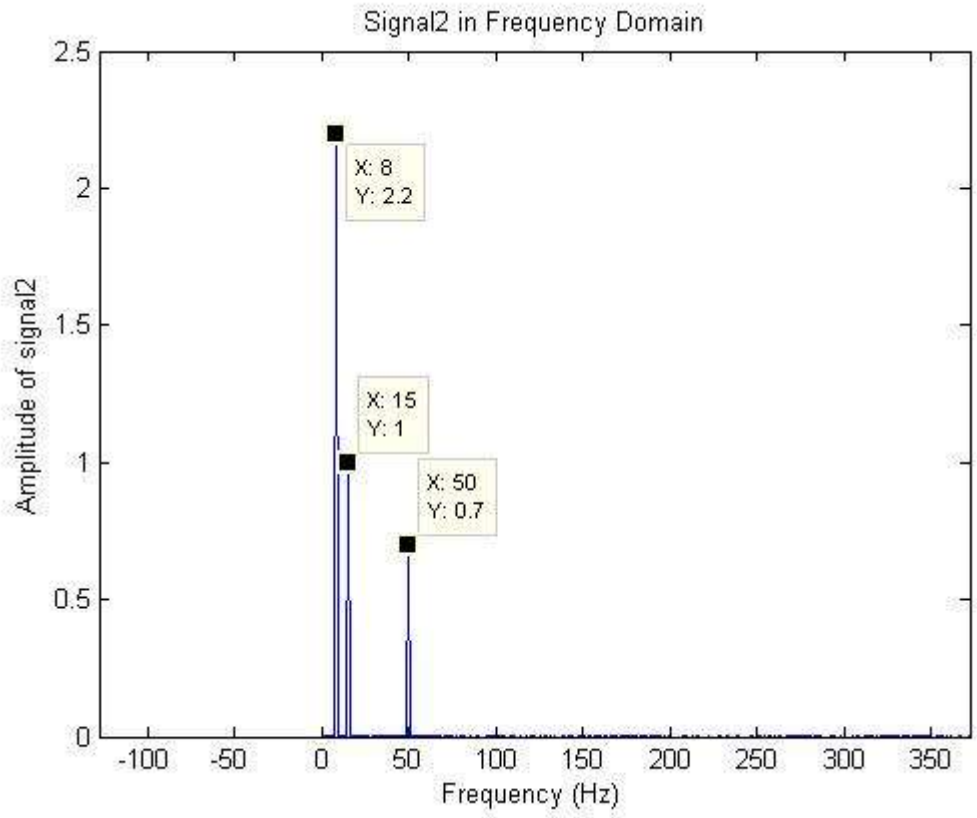
## Experiment 2: Analysis of given vector 'signal2' of length 1 second formed by linear combinations of three sinusoids of frequencies f1, f2 and f3.

Plot of the given vector



1. **Determine f1, f2, f3 and the Nyquist rate. Show the frequency plots that you use.**

   As done in the above experiment, we use the in-built MATLAB function of fast Fourier transform to find the signal in frequency domain and get the three constituent frequency signals




   From the above graph, we observe the three spikes which make up the three frequencies of the signal

   Freq f1 = 8 Hz

   Freq f2 = 15 Hz

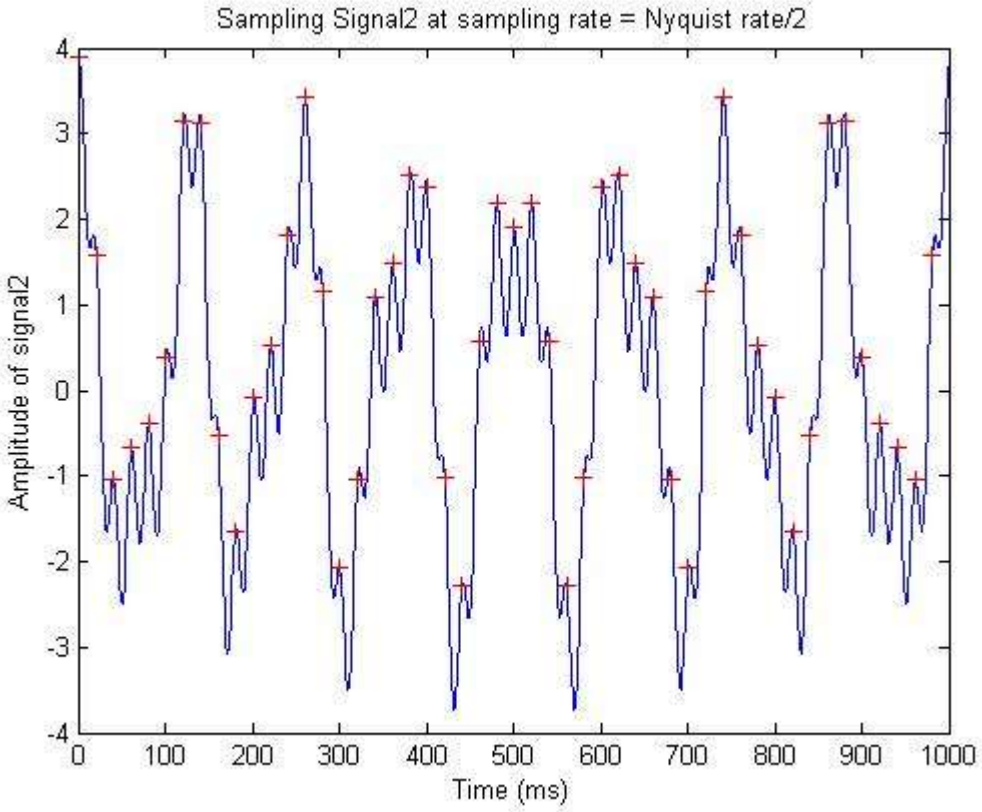
   Freq f3 = 50 Hz

   Nyquist Rate = 2 * max(f1, f2, f3) = 2 * max(8, 15, 50) = 2 * 50 = 100 Hz

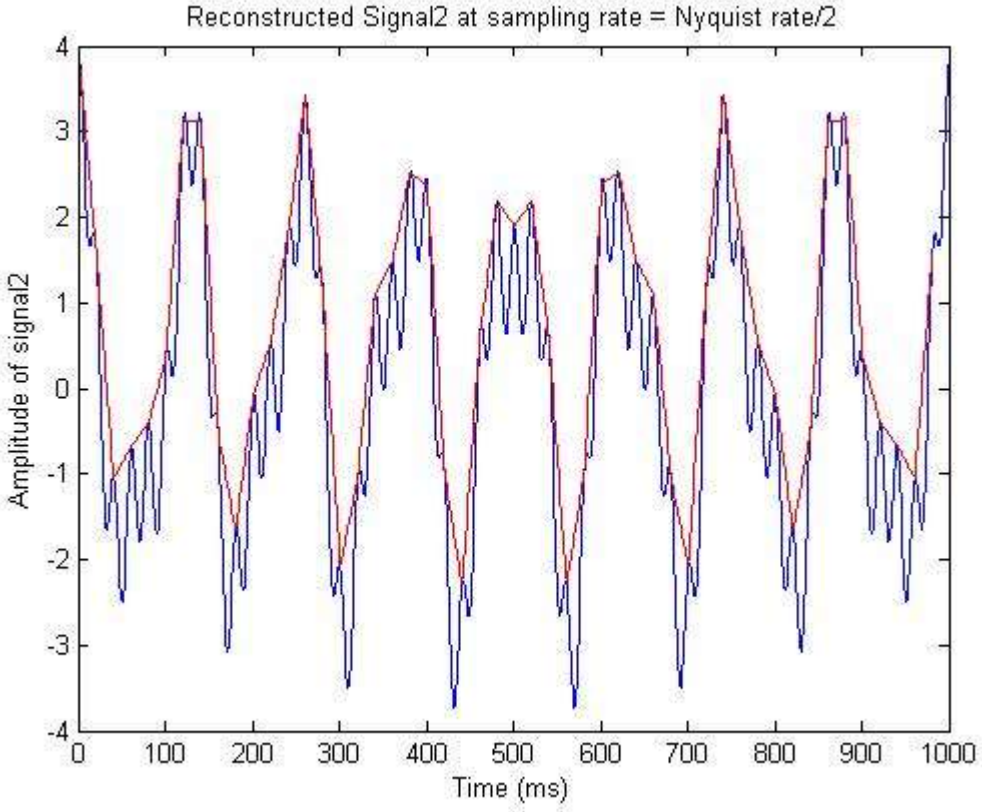
2. **Sample the given signal <u>non uniformly</u> at 2 different rates: To implement non-uniform sampling, use randomly generated points (at the 2 given rates). For each of the 2 cases, compute the reconstruction error (use linear interpolation), and compare it with the error obtained due to uniform sampling (at the 2 given rates).**

**(a) max(f1,f2,f3) = 50 Hz**

Uniform Sampling:




Uniform sampled reconstructed signal



The above two graphs are the instances of the uniform sampling over the original signal and the corresponding reconstructed signal.

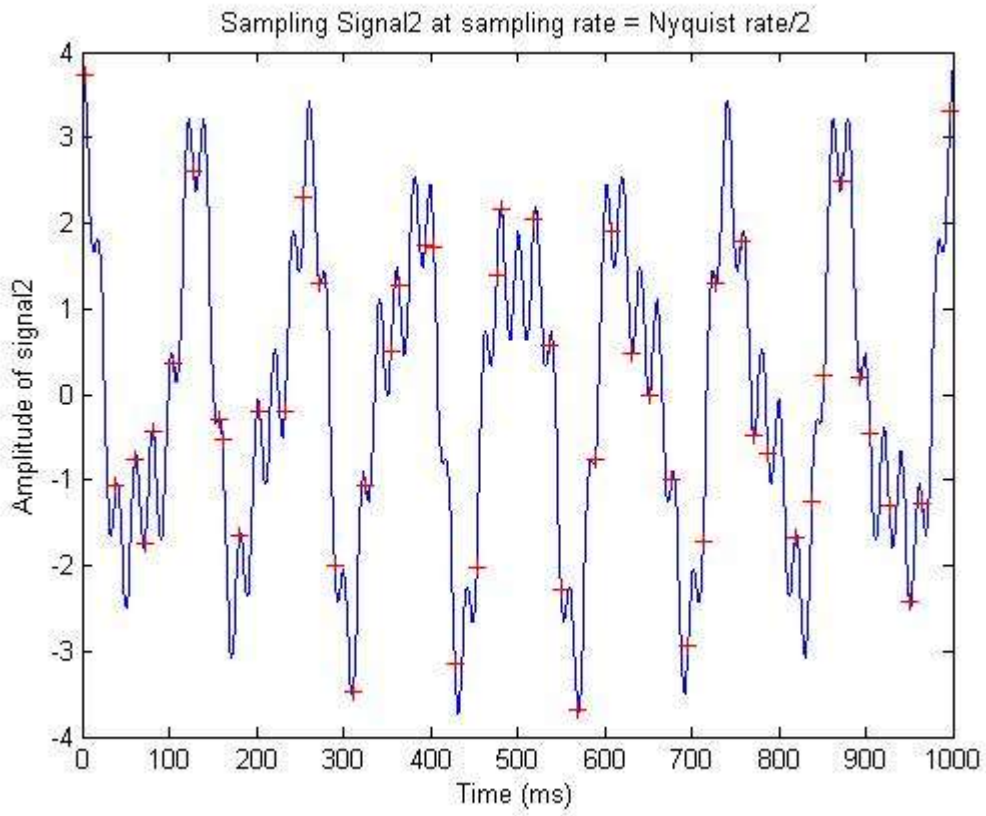
The reconstructed signal is not as accurate as the original signal. Hence the error should be significant.
Mean Squared Error (MSE) = 0.8096

Considering there are 981 points which have been used for sampling, and since the maximum amplitude of signal is 3.9, this reconstructed signal with MSE ~ 1 is quite significant and not a good approximation.

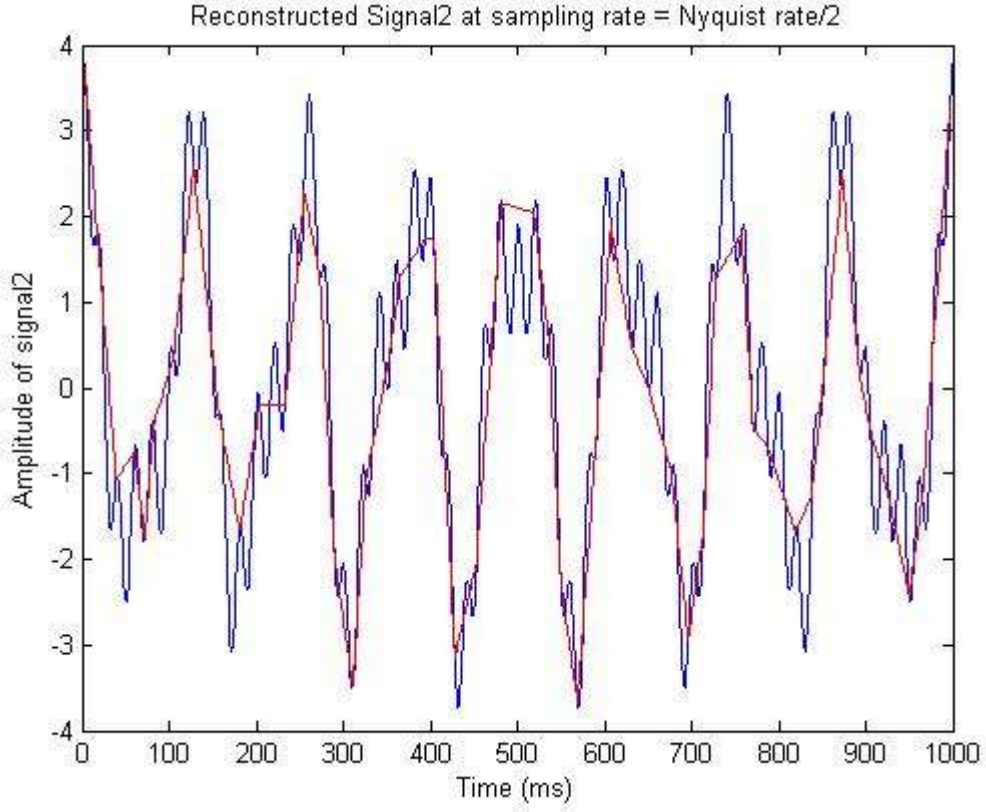
**Method used for not uniform sampling:**
For a given sampling rate (here 50 Hz), for gives 1 second signal, we should get 50 data points. There are several methods of randomly finding these samples. We have used a specific method which intuitively seems more accurate than the simple brute force method.

We divide the total signal in 50 parts (here). So that each part represents 0.02 sec or 20 ms signal. We have given sampled signal at sampling frequency = 1000 Hz. Therefore we have 1000 data points. Therefore for each millisecond we have 1 data point. We have 20 points corresponding to 20 ms signal. Of which we choose a point randomly. Similarly we choose random samples from each subsequent interval of 20 to get all the 50 samples.

<u>An instance of Non-uniform sampling</u>
Note that this graph is not unique because of the randomness present in the problem




Reconstructed signal for the above sampling




The above graph is one instance of the reconstructed signal
Because of the randomness present in the problem, this reconstructed signal is not unique

Mean Squared Error (MSE) = 0.5747

**Comparison of the MSE in the above two methods:**

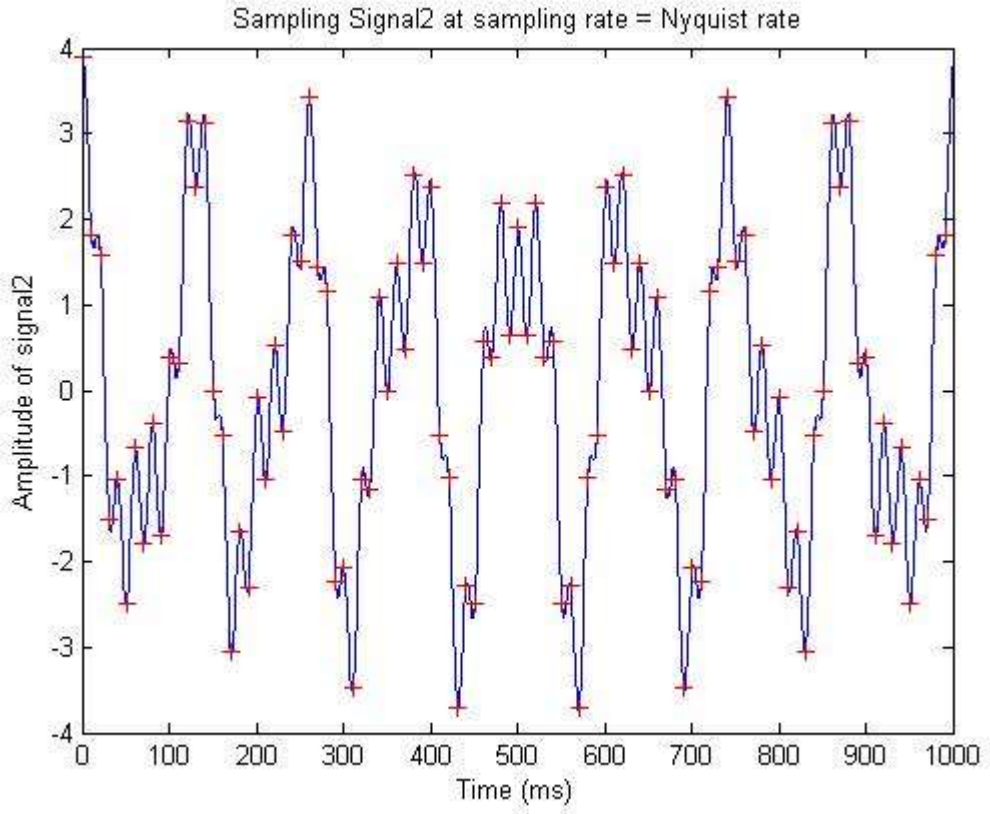
Uniform case: Mean Squared Error (MSE) = 0.8096

Non-uniform case: Mean Squared Error (MSE) = 0.5747

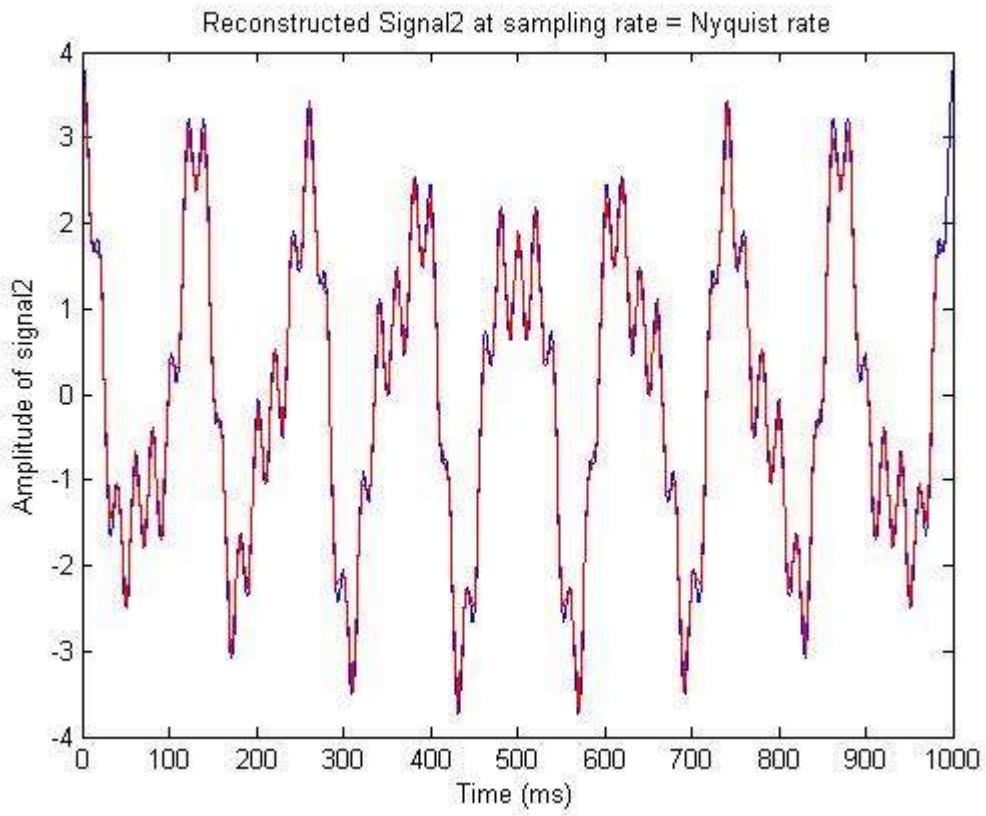
Here the error in the non-uniform case is much less.
This may be because of the sampling rate which is much less than the Nyquist rate. Therefore uniform sampling is as it is error-prone. The improvement in the non-uniform case may be because of the nature of the original graph which is why the randomness may favour this case.

**(b) At Nyquist rate = 100 Hz**

Uniform Sampling




Uniform sampled reconstructed signal




The above two graphs are the instances of the uniform sampling over the original signal and the corresponding reconstructed signal.

The reconstructed signal (red line) is closely resembles the original signal(blue line). Hence the error should be less.

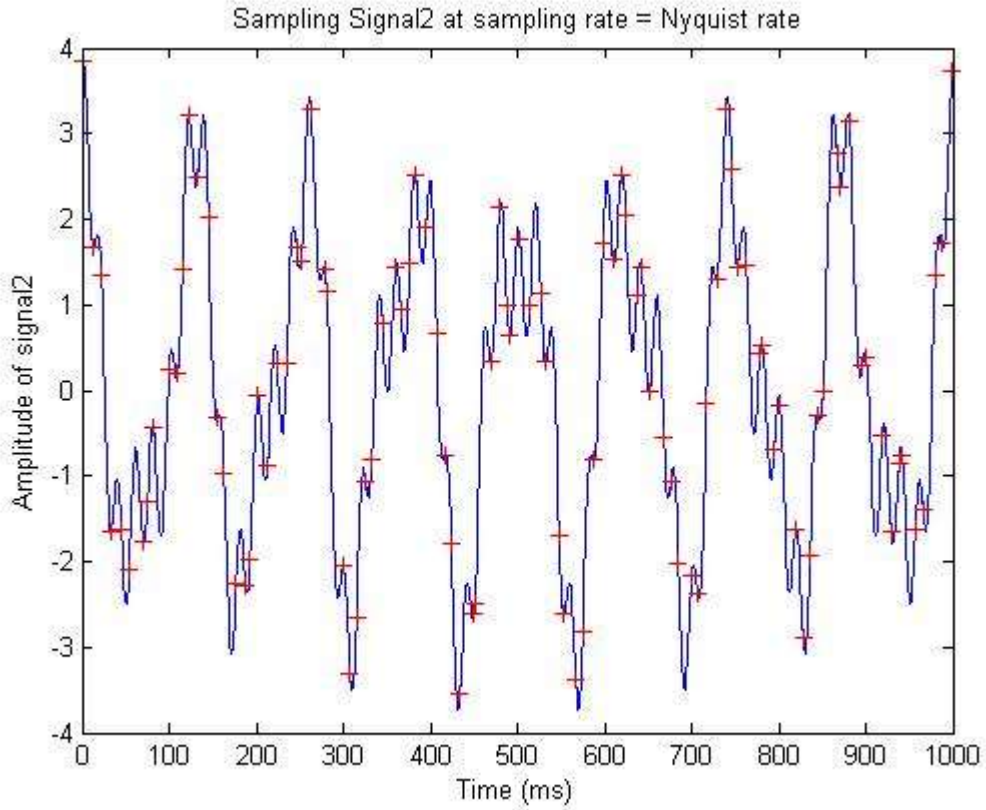
Mean Squared Error (MSE) = 0.0155, (for 991 sample points)

Similar to the observations, the MSE here is nearly zero which means the average error at each point is very less.

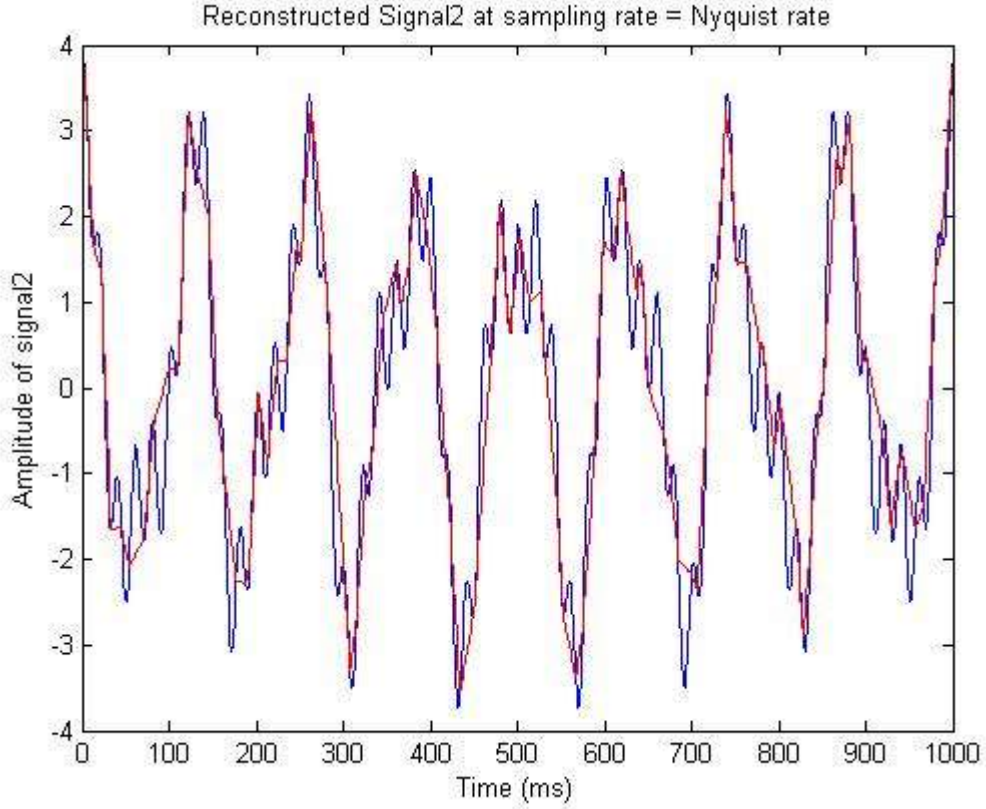
This again shows the decrease in error and increased resemblance of the reconstructed signal with the original signal at the Nyquist rate.

**Method of Sampling:**
The method used for sampling is same as in the above case.

<u>An instance of Non-uniform sampling</u>
Note that this graph is not unique because of the randomness present in the problem




Reconstructed signal for the above sampling instance




The above graph is one instance of the reconstructed signal
Because of the randomness present in the problem, this reconstructed signal is not unique

By comparing with the graph of uniformly reconstructed signal, this signal seems to have more error.

Mean Squared Error (MSE) = 0.2746

**Comparison of the MSE in the above two methods:**

Uniform case: Mean Squared Error (MSE) = 0.0155

Non-uniform case: Mean Squared Error (MSE) = 0.2746

This is the opposite situation to the previous case. Here, the error is the non-uniform case is significant as compared to the uniform case. The primary reason for this is the sampling rate which is equal to the Nyquist rate. Hence the uniform signal accurately represents the original signal and has error ~ 0.02. The randomness in the non-uniform case actually reduces the accuracy of approximation and by a large extent. Hence the error is large in comparison.

**QA. Does non-uniform sampling lead to lower reconstruction error?**

The above two cases show that non-uniform sampling may not necessarily lead to lower reconstruction error. In fact, as the sampling rate increases, due to randomness the error in the non-uniform case may not reduce as significantly as the uniform case and might exceed the error in the uniform case.

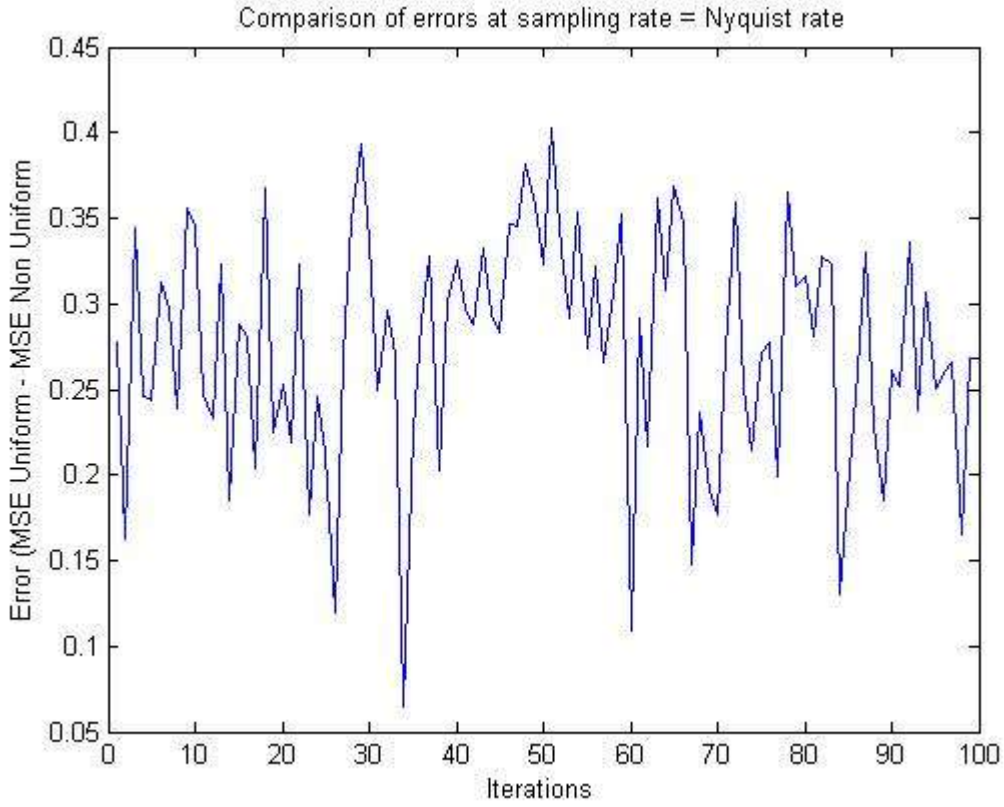
**QB. Can it lead to lower reconstruction error in any case?**

Yes. It is possible to have a lower reconstruction error in some case. The first case where sampling rate = Nyquist rate / 2 is clearly and example. Even in the second case where we have some minor reconstruction error, if the randomly generated points closely approximate the variations in actual function, then it might be possible to have a lower reconstructions error.

**Comparing Errors in both the methods for both the case over a number of iterations:**

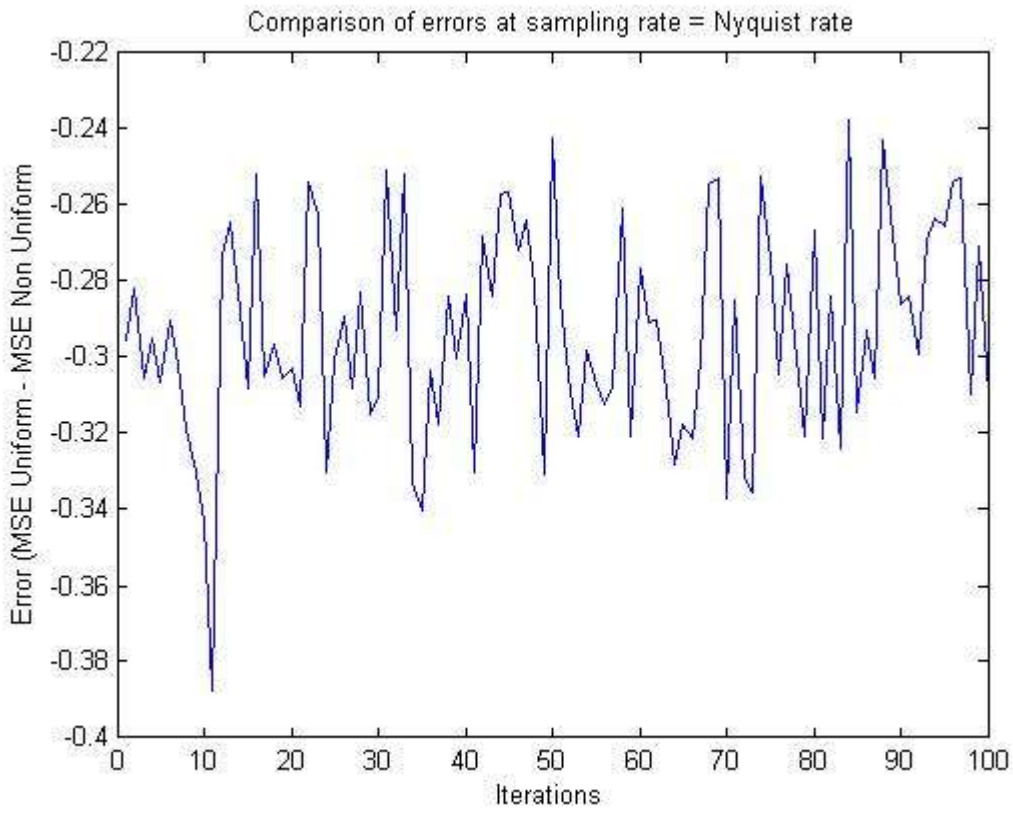
Here we are performing the non-uniform reconstruction for 100 iterations and comparing the error obtained with the error in the uniform case.

**Case a: Sample rate = max(f1,f2,f3)**




Here, the error in the uniform case is more that the non-uniform case although in the 33$^{rd}$ iteration, the error goes close to 0 implying the increase in error of non-uniform case. But overall it is less than uniform case.

**Case b: Sample rate = Nyquist rate**




Here, the error in the non-uniform case is much more that the nuniform case although in the 85th iteration, the error goes close to 0 implying the decrease in error of non-uniform case. But overall it is less than uniform case.

This shows that largely the error in the uniform case is less. But it might be possible to have less error in the non-uniform case if the randomly chose points follow the actual signal.

## Conclusion:

In both the experiments, we learned to convert the signal from time domain to frequency domain so as to get the constituent signals.

In the first experiment, we established that as the sampling rate increases, the reconstruction error decreases and the reconstructed signal closely resembles the original signal.

In the second experiment, we studied the effect of non-uniform sampling on the reconstruction of signal for different sampling rates and established that as the sampling rate increases, the uniform reconstructions becomes better than the non-uniform reconstruction.

## Codes:

### Experiment 1: signal1

#### Frequency domain conversion of original signal

```
% Basic program to study the effect of sampling in the domain of signal
% testing

clear;
close all;

% load the data from given .mat file
load('signal_lab1.mat');

% plot the original signal
figure
plot(signal1)
title('Signal1 in Time Domain')
xlabel('Time (ms)')
ylabel('amplitude of signal1')

% 1 sec signal, vector length 1000
% Therefore 1000 samples per sec
% Therefore sampling freq = 1000 Hz
Fs = 1000;      % Sampling frequency

L = length(signal1);    % length of signal


Y = fft(signal1);    % frequency domain signal complex domain
```

```
P2 = abs(Y/L);      % frequency domain signal converted to real domain

% The above one is a two sided spectrum of the signal
% we need to convert it into a one sided spectrum.
P1 = P2(1:L/2+1);

% adding the amplitude of the other side of the spectrum
P1(2:end-1) = 2*P1(2:end-1);

% frequency range, single side spectrum
f = Fs*(0:(L/2))/L;

% plot the signal in frequency domain
figure
plot(f,P1)
title('Signal1 in Frequency Domain')
xlabel('Frequency (Hz)')
ylabel('amplitude of signal1')
```

## Sampling and Reconstruction

```
clear;
close all;

% load the data from given .mat file
load('signal_lab1.mat');

L = length(signal1);    % length of original signal

Nq = 20;       % Nyquist rate 20 Hz

% 3 sampling rates
sampleRate_a = Nq/2;
sampleRate_b = Nq;
sampleRate_c = 4*Nq;

% uniformly sampled signals
sample_a = signal1(1 : abs(L/sampleRate_a) : L);     % L/sampleRate_a =
delta_t, the time step in sampling

sample_b = signal1(1 : abs(L/sampleRate_b) : L);

sample_c = signal1(1 : abs(L/sampleRate_c) : L);


% plotting the sampled signals on original signal
% smaple
figure
plot(signal1)
hold on
plot( (1 : abs(L/sampleRate_a) : L), sample_a, 'r+')
title('Sampling Signal1 at sampling rate = Nyquist rate/2')
xlabel('Time (ms)')
```

```
ylabel('Amplitude of signal1')

figure
plot(signal1)
hold on
plot( (1 : abs(L/sampleRate_b) : L), sample_b, 'r+')
title('Sampling Signal1 at sampling rate = Nyquist rate')
xlabel('Time (ms)')
ylabel('Amplitude of signal1')

figure
plot(signal1)
hold on
plot( (1 : abs(L/sampleRate_c) : L), sample_c, 'r+')
title('Sampling Signal1 at sampling rate = 4*Nyquist rate')
xlabel('Time (ms)')
ylabel('Amplitude of signal1')




%%%%%%%%%%%%%% Part 3 Interpolating to get original signal %%%%%%%%%%%%%%%%

% time intervals for each case
time_a = (1 : abs(L/sampleRate_a) : L);
time_b = (1 : abs(L/sampleRate_b) : L);
time_c = (1 : abs(L/sampleRate_c) : L);

% perform linear interpolation over sampled data in each case
regen_a = interp1(time_a, sample_a, 1:L);
regen_b = interp1(time_b, sample_b, 1:L);
regen_c = interp1(time_c, sample_c, 1:L);

% plot the interpolated signals
figure
plot(signal1)
hold on
plot(regen_a, 'r')
title('Reconstructed Signal1 at sampling rate = Nyquist rate/2')
xlabel('Time (ms)')
ylabel('Amplitude of signal1')

figure
plot(signal1)
hold on
plot(regen_b, 'r')
title('Reconstructed Signal1 at sampling rate = Nyquist rate')
xlabel('Time (ms)')
ylabel('Amplitude of signal1')

figure
plot(signal1)
hold on
plot(regen_c, 'r')
title('Reconstructed Signal1 at sampling rate = 4*Nyquist rate')
xlabel('Time (ms)')
ylabel('Amplitude of signal1')
```

```
%%%%%%%%%%%%% computing reconstruction error %%%%%%%%%%%%%

% Here we are taking the error to be the mean squared error MSE

% variables storing total squared error
totalError_a = 0;
totalError_b = 0;
totalError_c = 0;

% calculating the number of elements of array which have been interpolated
% successfully and over which the error can be found out
elem_a = 0;
elem_b = 0;
elem_c = 0;

for i = 1:L
    if ~isnan(regen_a(i))   % check if the interpolation has happened
        totalError_a = totalError_a + (signal1(i) - regen_a(i)) *
(signal1(i) - regen_a(i));
        elem_a = elem_a + 1;
    end
end

MSE_a = totalError_a / elem_a;

for i = 1:L
    if ~isnan(regen_b(i))
        totalError_b = totalError_b + (signal1(i) - regen_b(i)) *
(signal1(i) - regen_b(i));
        elem_b = elem_b + 1;
    end
end

MSE_b = totalError_b / elem_b;

for i = 1:L
    if ~isnan(regen_c(i))
        totalError_c = totalError_c + (signal1(i) - regen_c(i)) *
(signal1(i) - regen_c(i));
        elem_c = elem_c + 1;
    end
end

MSE_c = totalError_c / elem_c;
```

## Experiment 2: signal2

### Frequency domain conversion of original signal

```
% Basic program to study the effect of sampling in the domain of signal
% testing

clear;
close all;

% load the data from given .mat file
load('signal_lab1.mat');

% plot the original signal
figure
plot(signal2)
title('Signal2 in Time Domain')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')

% 1 sec signal, vector length 1000
% Therefore 1000 samples per sec
% Therefore sampling freq = 1000 Hz
Fs = 1000;      % Sampling frequency

L = length(signal2);    % length of signal

Y = fft(signal2);    % frequency domain signal complex domain

P2 = abs(Y/L);      % frequency domain signal converted to real domain

% The above one is a two sided spectrum of the signal
% we need to convert it into a one sided spectrum.
P1 = P2(1:L/2+1);

% adding the amplitude of the other side of the spectrum
P1(2:end-1) = 2*P1(2:end-1);

% frequency range, single side spectrum
f = Fs*(0:(L/2))/L;

% plot the signal in frequency domain
figure
plot(f,P1)
title('Signal2 in Frequency Domain')
xlabel('Frequency (Hz)')
ylabel('Amplitude of signal2')
```

## Uniform Sampling and reconstruction:

```
clear;
close all;

% load the data from given .mat file
load('signal_lab1.mat');

L = length(signal2);    % length of original signal

Nq = 100;       % Nyquist rate 100 Hz = 2 * max(8, 15, 50)

% 2 sampling rates
sampleRate_a = Nq/2;    % max(f1,f2,f3) = Nyquist rate/2
sampleRate_b = Nq;

% uniformly sampled signals
sample_a = signal2(1 : abs(L/sampleRate_a) : L);     % L/sampleRate_a =
delta_t, the time step in sampling

sample_b = signal2(1 : abs(L/sampleRate_b) : L);


% plotting the sampled signals on original signal
% smaple
figure
plot(signal2)
hold on
plot( (1 : abs(L/sampleRate_a) : L), sample_a, 'r+')
title('Sampling Signal2 at sampling rate = Nyquist rate/2')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')

figure
plot(signal2)
hold on
plot( (1 : abs(L/sampleRate_b) : L), sample_b, 'r+')
title('Sampling Signal2 at sampling rate = Nyquist rate')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')




%%%%%%%%%%%%%% Part 3 Interpolating to get original signal %%%%%%%%%%%%%%%

% time intervals for each case
time_a = (1 : abs(L/sampleRate_a) : L);
time_b = (1 : abs(L/sampleRate_b) : L);

% perform linear interpolation over sampled data in each case
regen_a = interp1(time_a, sample_a, 1:L);
regen_b = interp1(time_b, sample_b, 1:L);

% plot the interpolated signals
figure
plot(signal2)
hold on
```

```
plot(regen_a, 'r')
title('Reconstructed Signal2 at sampling rate = Nyquist rate/2')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')

figure
plot(signal2)
hold on
plot(regen_b, 'r')
title('Reconstructed Signal2 at sampling rate = Nyquist rate')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')




%%%%%%%%%%%%% computing reconstruction error %%%%%%%%%%%%%

% Here we are taking the error to be the mean squared error MSE

totalError_a = 0;
totalError_b = 0;

% calculating the number of elements of array which have been interpolated
% successfully and over which the error can be found out
elem_a = 0;
elem_b = 0;

for i = 1:L
    if ~isnan(regen_a(i))   % check if the interpolation has happened
        totalError_a = totalError_a + (signal2(i) - regen_a(i)) *
(signal2(i) - regen_a(i));
        elem_a = elem_a + 1;
    end
end

MSE_a = totalError_a / elem_a;

for i = 1:L
    if ~isnan(regen_b(i))
        totalError_b = totalError_b + (signal2(i) - regen_b(i)) *
(signal2(i) - regen_b(i));
        elem_b = elem_b + 1;
    end
end

MSE_b = totalError_b / elem_b;
```

## Non-uniform Sampling and reconstruction:

```
clear;
close all;

% load the data from given .mat file
load('signal_lab1.mat');

L = length(signal2);    % length of original signal

Nq = 100;       % Nyquist rate 100 Hz = 2 * max(8, 15, 50)

% 2 sampling rates
sampleRate_a = Nq/2;    % max(f1,f2,f3) = Nyquist rate/2
sampleRate_b = Nq;
% These are no. of intervals in each case
% These many no. of samples we need out of 1000
% we take 1 sample randomly from each interval

% size of interval from which we select sample
intervalSize_a = L / sampleRate_a;
intervalSize_b = L / sampleRate_b;

% non uniform sampling
sample_a = zeros(1, sampleRate_a);
index_a = zeros(1, sampleRate_a);
for i = 1 : sampleRate_a
    index_a(i) = intervalSize_a * (i-1) + randi(intervalSize_a,1);
    sample_a(i) = signal2(index_a(i));
end


sample_b = zeros(1, sampleRate_b);
index_b = zeros(1, sampleRate_b);
for i = 1 : sampleRate_b
    index_b(i) = intervalSize_b * (i-1) + randi(intervalSize_b,1);
    sample_b(i) = signal2(index_b(i));
end


% plotting the sampled signals on original signal
% smaple
figure
plot(signal2)
hold on
plot( index_a, sample_a, 'r+')
title('Sampling Signal2 at sampling rate = Nyquist rate/2')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')

figure
plot(signal2)
hold on
plot( index_b, sample_b, 'r+')
title('Sampling Signal2 at sampling rate = Nyquist rate')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')
```

```
%%%%%%%%%%%%%%% Part 3 Interpolating to get original signal %%%%%%%%%%%%%%%%


% perform linear interpolation over sampled data in each case
regen_a = interp1(index_a, sample_a, 1:L);
regen_b = interp1(index_b, sample_b, 1:L);

% plot the interpolated signals
figure
plot(signal2)
hold on
plot(regen_a, 'r')
title('Reconstructed Signal2 at sampling rate = Nyquist rate/2')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')

figure
plot(signal2)
hold on
plot(regen_b, 'r')
title('Reconstructed Signal2 at sampling rate = Nyquist rate')
xlabel('Time (ms)')
ylabel('Amplitude of signal2')




%%%%%%%%%%%%%% computing reconstruction error %%%%%%%%%%%%%%

% Here we are taking the error to be the mean squared error MSE
totalError_a = 0;
totalError_b = 0;

% calculating the number of elements of array which have been interpolated
% successfully and over which the error can be found out
elem_a = 0;
elem_b = 0;

for i = 1:L
    if ~isnan(regen_a(i))   % check if the interpolation has happened
        totalError_a = totalError_a + (signal2(i) - regen_a(i)) *
(signal2(i) - regen_a(i));
        elem_a = elem_a + 1;
    end
end

MSE_a = totalError_a / elem_a;

for i = 1:L
    if ~isnan(regen_b(i))
        totalError_b = totalError_b + (signal2(i) - regen_b(i)) *
(signal2(i) - regen_b(i));
        elem_b = elem_b + 1;
    end
end
```

```
MSE_b = totalError_b / elem_b;
```

Code to check error difference over 100 of iterations.

```
clear;
close all;

% load the data from given .mat file
load('signal_lab1.mat');

L = length(signal2);    % length of original signal

Nq = 100;       % Nyquist rate 100 Hz = 2 * max(8, 15, 50)

% 2 sampling rates
sampleRate_a = Nq/2;    % max(f1,f2,f3) = Nyquist rate/2
sampleRate_b = Nq;
% These are no. of intervals in each case
% These many no. of samples we need out of 1000
% we take 1 sample randomly from each interval

% size of interval from which we select sample
intervalSize_a = L / sampleRate_a;
intervalSize_b = L / sampleRate_b;


% iterating to find the variations in answer due to randomness
iterations = 100;

MSE_a = zeros(iterations, 1);
MSE_b = zeros(iterations, 1);

for iter = 1:iterations
    % non uniform sampling
    sample_a = zeros(1, sampleRate_a);
    index_a = zeros(1, sampleRate_a);
    for i = 1 : sampleRate_a
        index_a(i) = intervalSize_a * (i-1) + randi(intervalSize_a,1);
        sample_a(i) = signal2(index_a(i));
    end


    sample_b = zeros(1, sampleRate_b);
    index_b = zeros(1, sampleRate_b);
    for i = 1 : sampleRate_b
        index_b(i) = intervalSize_b * (i-1) + randi(intervalSize_b,1);
        sample_b(i) = signal2(index_b(i));
    end
```

```
    %%%%%%%%%%%%%% Part 3 Interpolating to get original signal
%%%%%%%%%%%%%%%%


    % perform linear interpolation over sampled data in each case
    regen_a = interp1(index_a, sample_a, 1:L);
    regen_b = interp1(index_b, sample_b, 1:L);



    %%%%%%%%%%%%% computing reconstruction error %%%%%%%%%%%%%%

    % Here we are taking the error to be the mean squared error MSE
    totalError_a = 0;
    totalError_b = 0;

    % calculating the number of elements of array which have been
interpolated
    % successfully and over which the error can be found out
    elem_a = 0;
    elem_b = 0;

    for i = 1:L
        if ~isnan(regen_a(i))   % check if the interpolation has happened
            totalError_a = totalError_a + (signal2(i) - regen_a(i)) *
(signal2(i) - regen_a(i));
            elem_a = elem_a + 1;
        end
    end

    MSE_a(iter) = totalError_a / elem_a;

    for i = 1:L
        if ~isnan(regen_b(i))
            totalError_b = totalError_b + (signal2(i) - regen_b(i)) *
(signal2(i) - regen_b(i));
            elem_b = elem_b + 1;
        end
    end

    MSE_b(iter) = totalError_b / elem_b;

end

MSE_uni_a = 0.8096;
MSE_uni_b = 0.0155;

figure
plot(1:iterations, MSE_uni_a-MSE_a)
title('Comparison of errors at sampling rate = Nyquist rate')
xlabel('Iterations')
ylabel('Error (MSE Uniform - MSE Non Uniform')

figure
plot(1:iterations, MSE_uni_b-MSE_b)
title('Comparison of errors at sampling rate = Nyquist rate')
xlabel('Iterations')
ylabel('Error (MSE Uniform - MSE Non Uniform')
```