Name : Rahul Swar
Roll No. : 19ECE1028
Course : Digital Communication Laboratory (DC Lab)

Experiment No. -1

Sampling And Reconstruction

Aim :-

To study Sampling and Reconstruction of Bandpass and Passband Signals, and observe the various possibilities of Sampling, like Undersampling (M<2), Perfect Sampling (M=2) and Oversampling (M>2), by choosing different Sampling Frequencies, that can be obtained by varying the Nyquist constant M, which is the ratio of Sampling Frequency to that of maximum Signal Frequency ($M=f_S/fmax$).

Tools Used :-

Anaconda's Spyder Python Simulation Software.

Theory :-

Sampling is the conversion of a continuous-time signal into a discrete-time signal obtained by taking "samples" of the continuous time signal at discrete time instants. One of the sampling techniques to sample an analog signal is periodic or uniform sampling described by the relation,

$$x(n) = x_a(nT), \qquad -\infty < n < \infty \ ,$$

where $x(n)$ is the discrete time signal obtained by "taking samples" of the analog signal $x_a(t)$ every T seconds. The time interval T is called the sampling interval and its reciprocal $1/T = F_s$ is called the sampling rate. The desired relationship between the spectrum X(F) of the discrete-time signal and the spectrum $X_a(F)$ is given by,

$$X(F) = F_s \sum_{k=-\infty}^{\infty} X_a(F - kF_s)$$

Derivation :

The mathematical expression of sampled signal can be given as,

$$\mathbf{x(t)=x_a(t).}\delta(t)\mathbf{.....(1)}$$

The unit impulse function $\delta(t)$ can be represented in terms of Fourier Series as,

$$\delta(t) = a_0 + \sum_{n=1}^{\infty} (a_n cosn\omega_s t + b_n sinn\omega_s t) \text{.....(2)}$$

Where, $a_0 = \frac{1}{Ts} \int_{-T/2}^{T/2} \delta(t).dt = (\frac{1}{Ts}).(\delta(0)) = \frac{1}{Ts}$

$$an = \frac{1}{Ts} \int_{-T/2}^{T/2} \delta(t).cosn\mathbf{w_s}t dt = (\frac{2}{Ts}).\delta(0) cosn\omega_s 0 = \frac{2}{Ts}$$

$$bn = \frac{1}{Ts} \int_{-T/2}^{T/2} \delta(t).sinn\omega_s t dt = (\frac{2}{Ts}).\delta(0) sinn\omega_s 0 = 0$$

Putting the above values in equation **(2)**, we get,

$$\therefore \delta(t) = \frac{1}{Ts} + \sum_{n=1}^{\infty} (\frac{2}{Ts}(cosn\omega_s t) + 0)$$

Putting δ(t) in equation **(1)**, we get,

$$\rightarrow x(t) = x_a(t).\delta(t)$$

$$x(t) = xa(t)[\frac{1}{Ts} + \sum_{n=1}^{\infty} (\frac{2}{Ts} cosn\omega_s t)]$$

$$x(t) = \frac{1}{Ts} [xa(t) + 2 \sum_{n=1}^{\infty} (cosn\omega_s t) xa(t)]$$

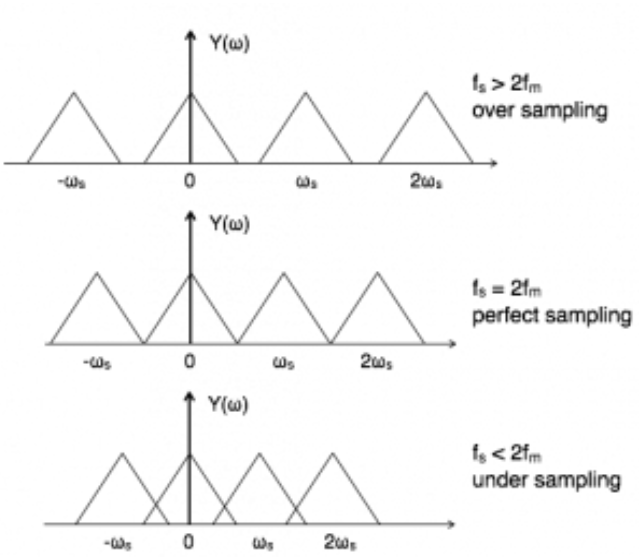
$$x(t) = \frac{1}{Ts}[xa(t) + 2cos\omega_s t.\ xa(t) + 2cos2\omega_s t.\ xa(t) + 2cos3\omega_s t.\ xa(t). . . . .]$$

Taking Fourier Transform on both sides of the above equation, we get,

$$X(\omega) = \frac{1}{Ts}[Xa(\omega) + Xa(\omega - \omega_s) + Xa(\omega + \omega_s) + Xa(\omega - 2\omega_s) + Xa(\omega + 2\omega_s) + . . .]$$

$$\therefore X(\omega) = \frac{1}{Ts} \sum_{n=-\infty}^{\infty} (\omega - n\omega_s)] \text{ ,where } n = 0, \pm 1, \pm 2, ...$$

$$\text{or, } X(f) = f_s \sum_{n=-\infty}^{\infty} X_a(f - nfs)$$



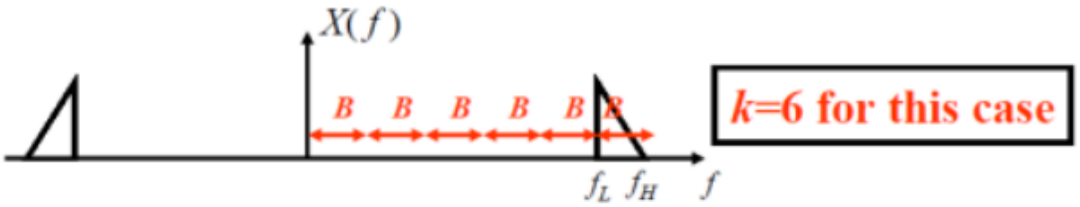
Sampling Theorem: A bandlimited continuous-time signal, with highest frequency B Hz, can be uniquely recovered from its samples provided that the sampling rate $F_s \geq 2B$ samples per second. If this condition is met, then $x_a(t)$ can be exactly reconstructed from,

$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a(\frac{n}{F_s}) g(t - \frac{n}{F_s}) \text{ where } g(t) \equiv \frac{\sin sin\, 2\pi Bt}{2\pi Bt}$$

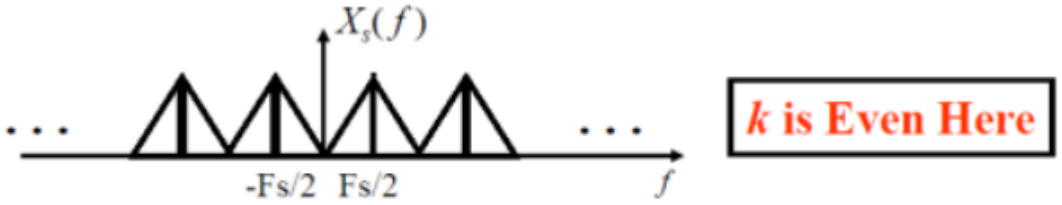
If the sampling frequency is less than two times the highest frequency component in the sampled signal, then it leads to an aliasing effect and the signal cannot be reconstructed perfectly.

Bandpass Signal: A continuous time bandpass signal with bandwidth B and center frequency F_C has its frequency content in the two frequency bands defined by $0 < F_L < |F| < F_H$, where $F_C = \frac{F_L + F_H}{2}$. The minimum sampling rate required to avoid aliasing is $F_{smax} = \frac{2F_H}{\lfloor \frac{F_H}{B} \rfloor}$. The range of acceptable sampling rates is determined by, $\frac{2F_H}{k} \leq F_s \leq \frac{2F_L}{k-1}$ where k is an integer number given by $1 \leq k \leq \lfloor \frac{F_H}{B} \rfloor$.

Consider the case where $f_H = LB$ (k an Even Integer)



Whenever $f_H = LB$, we can choose $Fs = 2B$ to perfectly "interweave" the shifted spectral replicas



Procedure :-

1. Generate a band limited signal x(t) of your choice with frequency fm.
2. Choose sampling frequency fs (more than Nyquist rate) and sample the signal x(t). Let $x_S(n)$ is the sampled signal.
3. Determine the DFT X s (K) of the sampled signal $x_S(n)$.
4. Pass the sampled signal through a low pass filter of cut-off frequency $f_S/2$.
5. Plot all the signals and spectrums with proper labelling.
6. Vary the sampling frequency to observe under sampling.
7. Repeat the same for the passband signal.

Code :-

(i) Bandlimited/ Bandpass Signal :

```
import numpy as np
import matplotlib.pyplot as mplt
from scipy.fft import fft
from scipy import signal


f=50 #baseband signal frequency in Hz.
M=10 #minimum, M=2, for Nyquist Criteria : fs>=2fmax.
```

```
fs=M*f #sampling frequency in Hz.
T=1/f #time period of baseband signal in s.
t=np.arange(0,5*T,0.1*T) #time of baseband signal in s.
ts=np.arange(0,5*T,1/fs) #sampling time of baseband signal in s.

x=np.cos(2*np.pi*f*t) #baseband signal construction.

mplt.plot(t,x) #baseband signal plotting.
mplt.title("Baseband Signal")
mplt.xlabel("t-->")
mplt.ylabel("Signal-->")
mplt.show()

xs=np.cos(2*np.pi*f*ts) #sampled baseband signal construction.

mplt.stem(ts,xs) #sampled baseband signal plotting.
mplt.title("Sampled Baseband Signal")
mplt.xlabel("nTs-->")
mplt.ylabel("Signal-->")
mplt.show()

x_f=fft(x) #fft of baseband signal.

mplt.plot(abs(x_f)) #plot of fft of baseband signal.
mplt.title("FFT/Magnitude Spectrum Of Baseband Signal")
mplt.show()

wp=(fs/2)/fs #pass-band frequency
ws=(fs-10)/fs #stop-band frequency
apb=3 #pass-band attenuation in dB
asb=60 #stop-band attenuation in dB
```

```
[N, Wn]=signal.buttord(wp,ws,apb,asb) #determining order and cut-off frequency of lpf.
[b, a]=signal.butter(N,Wn,'low') #finding constants for lpf.

xr=signal.lfilter(b,a,xs) #baseband signal reconstruction.

mplt.plot(t,xr) #reconstructed baseband signal plotting.
mplt.title("Reconstructed Baseband Signal")
mplt.xlabel("t-->")
mplt.ylabel("Signal-->")
mplt.show()

xr_f=fft(xr) #fft of reconstructed baseband signal.

mplt.plot(abs(xr_f)) #plot of fft of reconstructed baseband signal.
mplt.title("FFT/Magnitude Spectrum Of Reconstructed Baseband Signal")
mplt.show()
```

(ii) Passband Signal :

```
import numpy as np
import matplotlib.pyplot as mplt
from scipy.fft import fft
from scipy import signal

f=100 #passband signal frequency in Hz.
M=10 #minimum, M=2, for Nyquist Criteria : fs>=2fmax.
A=1 #Amplitude of passband signal.
fc=1000 #carrier signal frequency in Hz.
Ac=2 #carrier signal amplitude.
fs=M*(f+fc) #sampling frequency in Hz.
T=1/f #time period of passband signal in s.
Ts=1/fs #Sampling Time period in s.
```

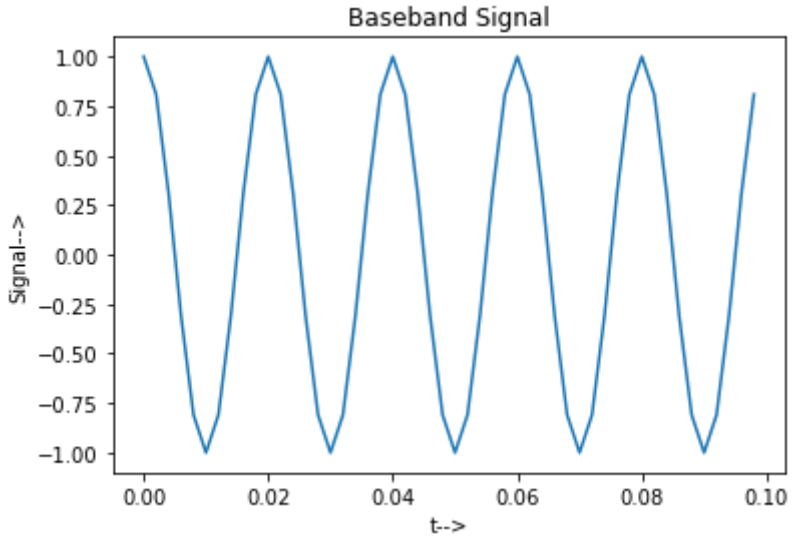
```
t=np.arange(0,2*T,1*Ts) #time of passband signal in s.
ts=np.arange(0,2*T,1*Ts) #sampling time of passband signal in s.

x=A*np.cos(2*np.pi*f*t) #passband signal construction.
xc=Ac*np.cos(2*np.pi*fc*t) #carrier signal construction.
X=np.multiply(x,xc) #Resultant passband signal. (DSBSC)

mplt.plot(t,X) #passband signal plotting.
mplt.title("Passband Signal")
mplt.xlabel("t-->")
mplt.ylabel("Signal-->")
mplt.show()

xs=A*np.cos(2*np.pi*f*ts) #sampled passband signal construction.
xc_s=Ac*np.cos(2*np.pi*fc*ts) #sampled carrier signal construction.
Xs=np.multiply(xs,xc) #Resultant sampled passband signal.

mplt.stem(ts,Xs) #sampled passband signal plotting.
mplt.title("Sampled Passband Signal")
mplt.xlabel("nTs-->")
mplt.ylabel("Signal-->")
mplt.show()

X_f=fft(X) #fft of passband signal.

mplt.plot(abs(X)) #plot of fft of passband signal.
mplt.title("FFT/Magnitude Spectrum Of Passband Signal")
mplt.show()

wp=(fs/2)/fs #pass-band frequency
ws=(fs-10)/fs #stop-band frequency
apb=3 #pass-band attenuation in dB
```

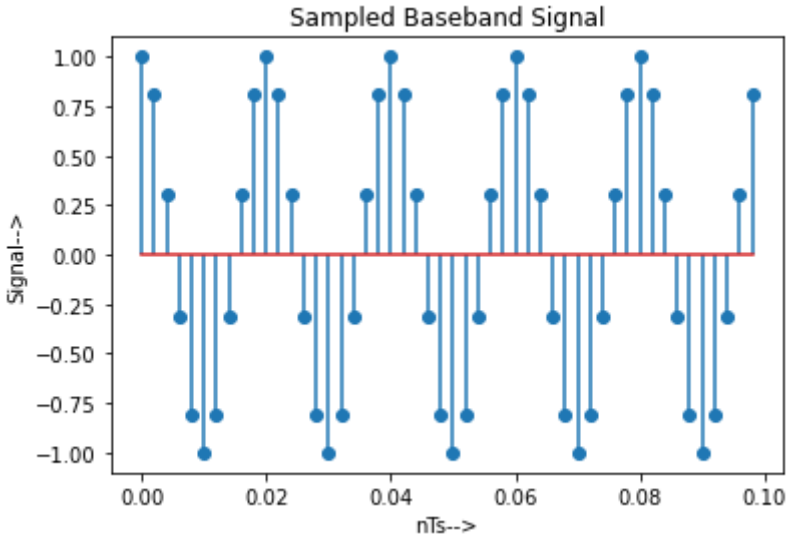
```
asb=60 #stop-band attenuation in dB
[N, Wn]=signal.buttord(wp,ws,apb,asb) #determining order and cut-off frequency of lpf.
[b, a]=signal.butter(N,Wn,'low') #finding constants for lpf.

Xr=signal.lfilter(b,a,Xs) #passband signal reconstruction.

mplt.plot(t,Xr) #reconstructed passband signal plotting.
mplt.title("Reconstructed Passband Signal")
mplt.xlabel("t-->")
mplt.ylabel("Signal-->")
mplt.show()

Xr_f=fft(Xr) #fft of reconstructed passband signal.

mplt.plot(abs(Xr_f)) #plot of fft of reconstructed passband signal.
mplt.title("FFT/Magnitude Spectrum Of Reconstructed Passband Signal")
mplt.show()
```

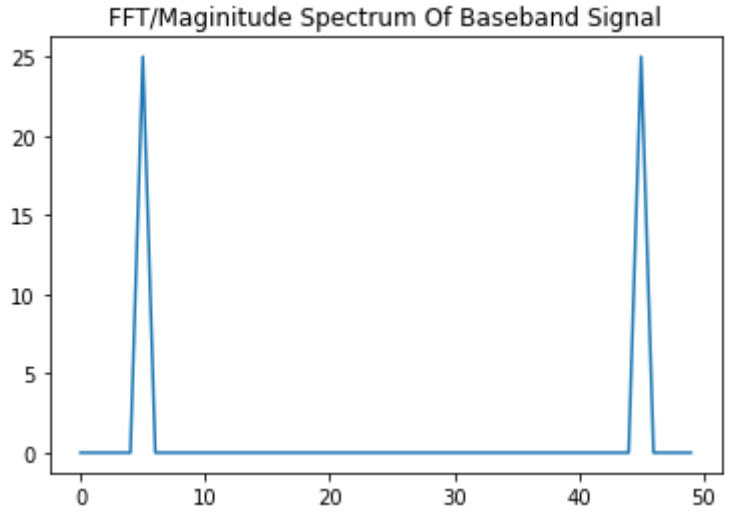
Graphs :-

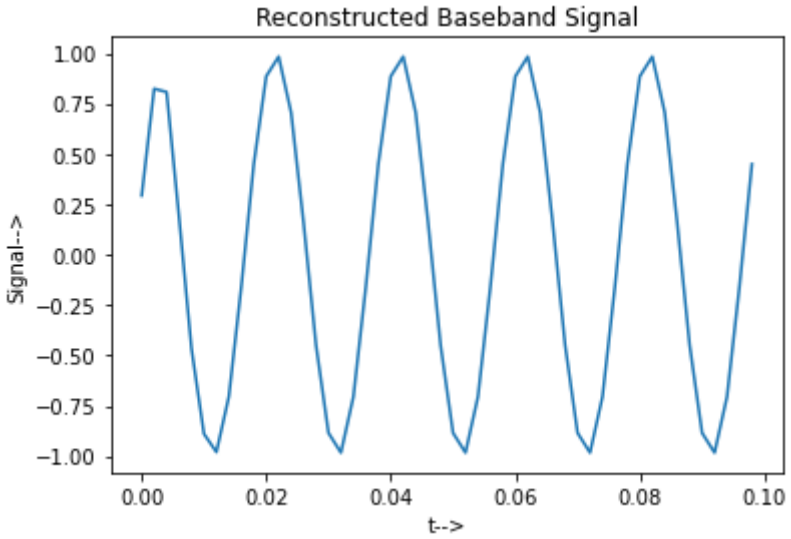
(i) Bandlimited/Bandpass Signal :

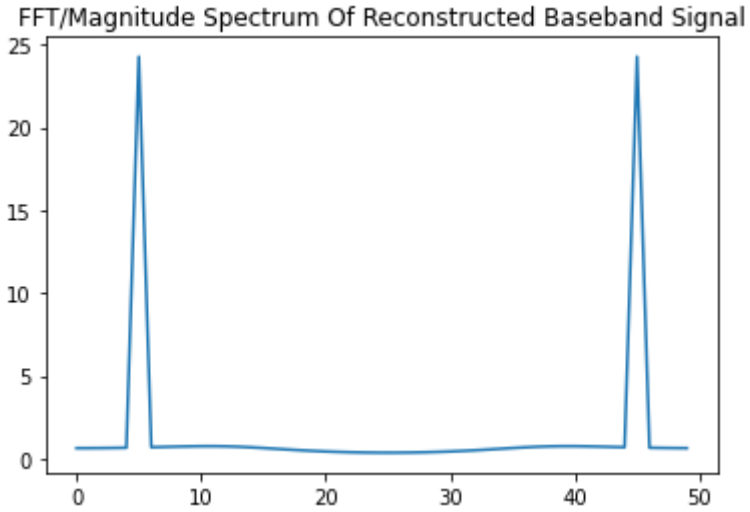
Oversampling (M=10)




Sampled Baseband Signal
Signal-->
nTs-->
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.00
0.02
0.04
0.06
0.08
0.10

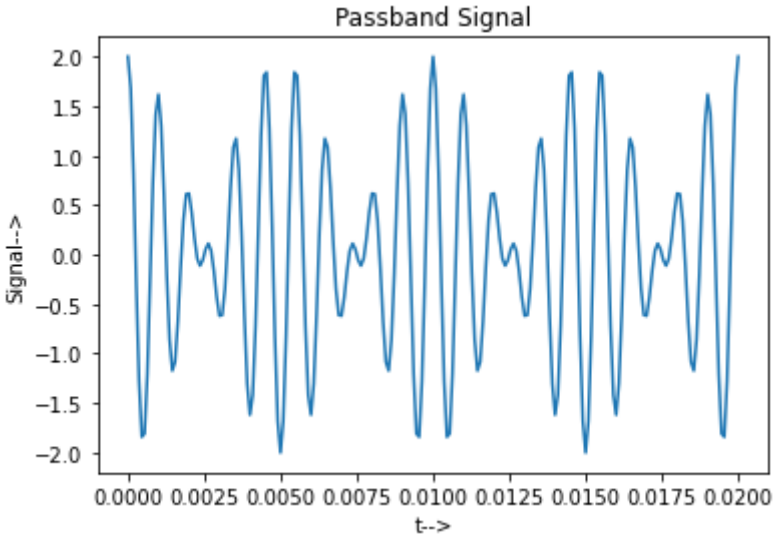

FFT/Maginitude Spectrum Of Baseband Signal
25
20
15
10
5
0
0
10
20
30
40
50

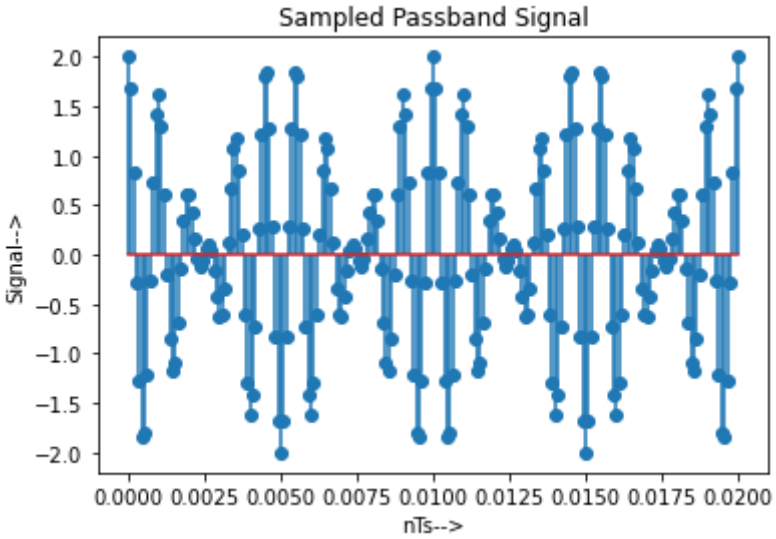

Reconstructed Baseband Signal
Signal-->
t-->
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.00
0.02
0.04
0.06
0.08
0.10

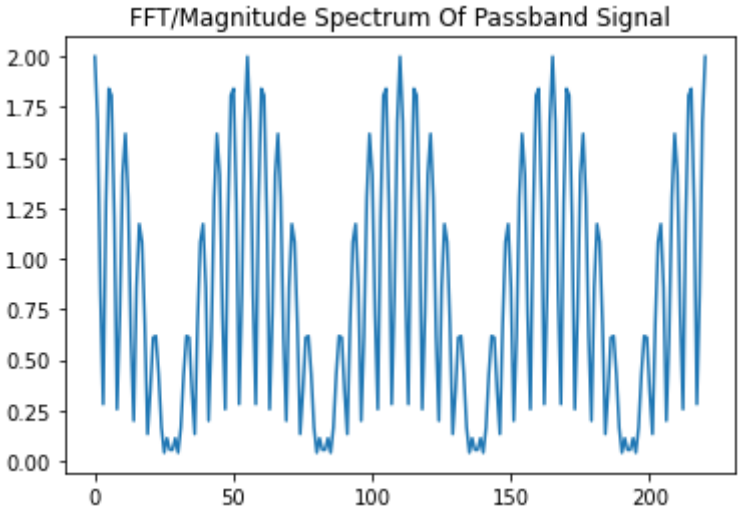


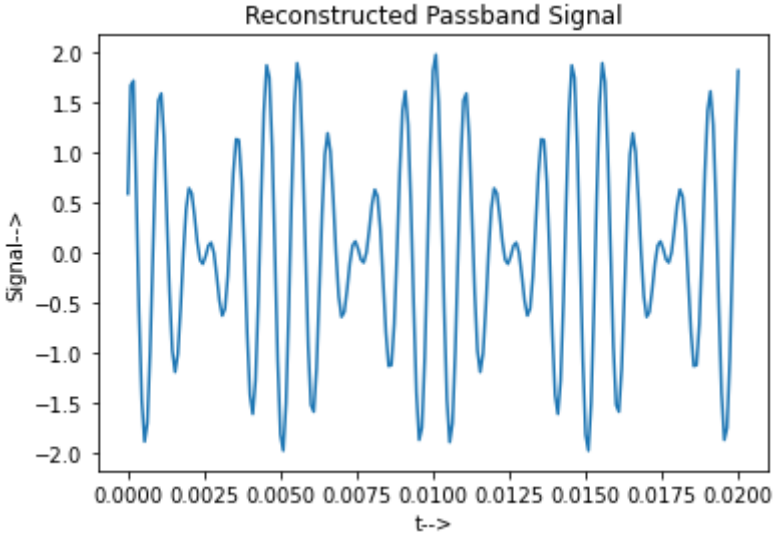
(ii) Passband Signal :

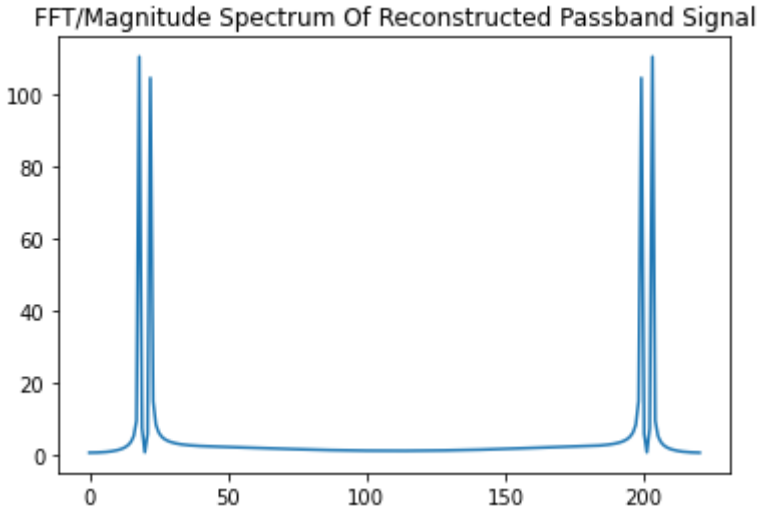
Case 1 : Oversampling (M=10)



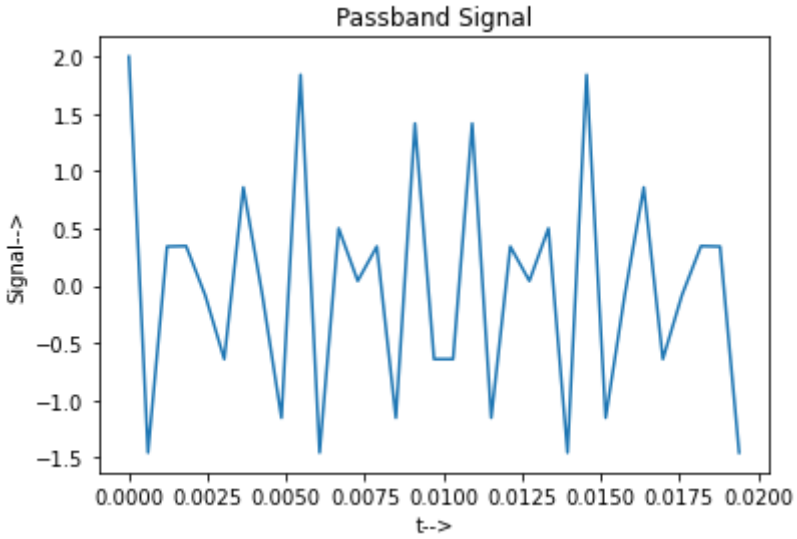

Sampled Passband Signal
Signal-->
nTs-->
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200

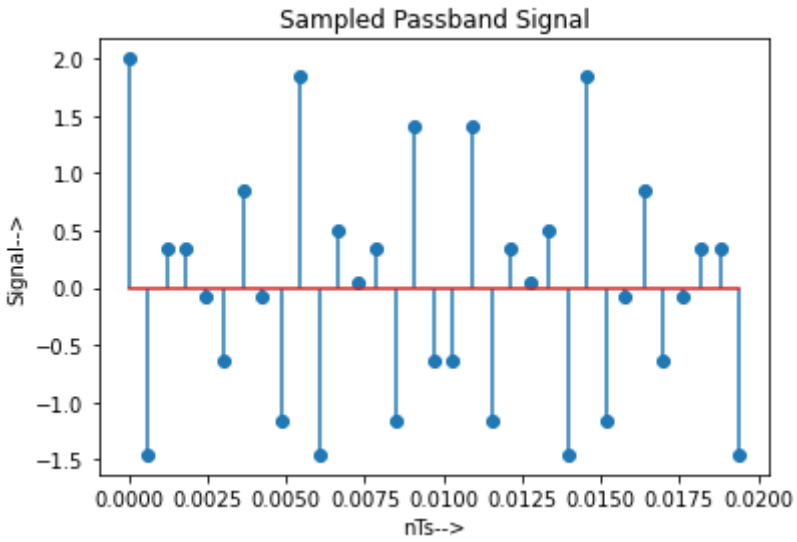

FFT/Magnitude Spectrum Of Passband Signal
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
50
100
150
200

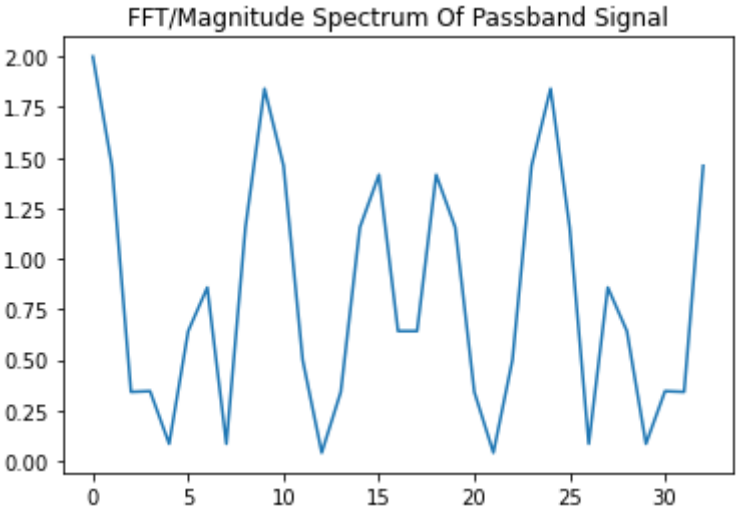

Reconstructed Passband Signal
Signal-->
t-->
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200

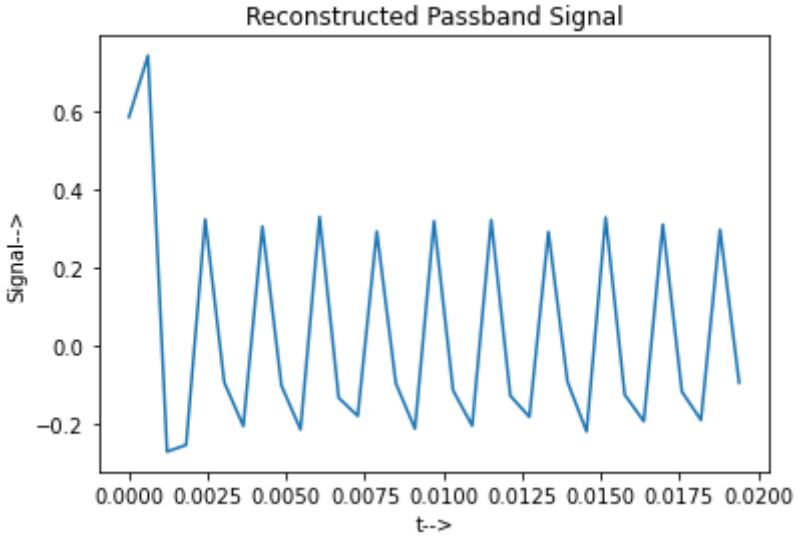


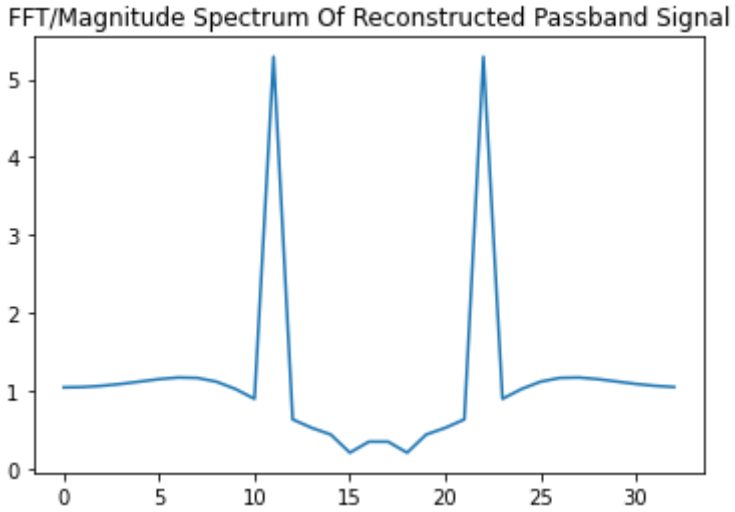
Case 2 : Undersampling (M=1.5)



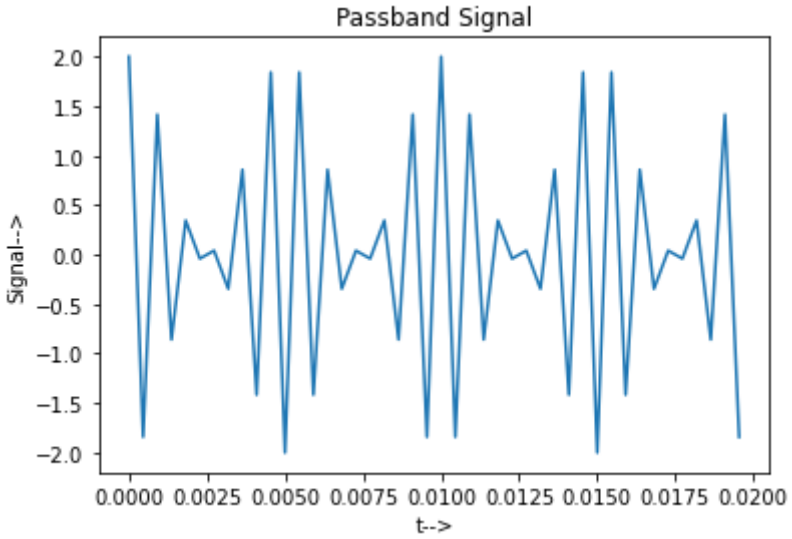


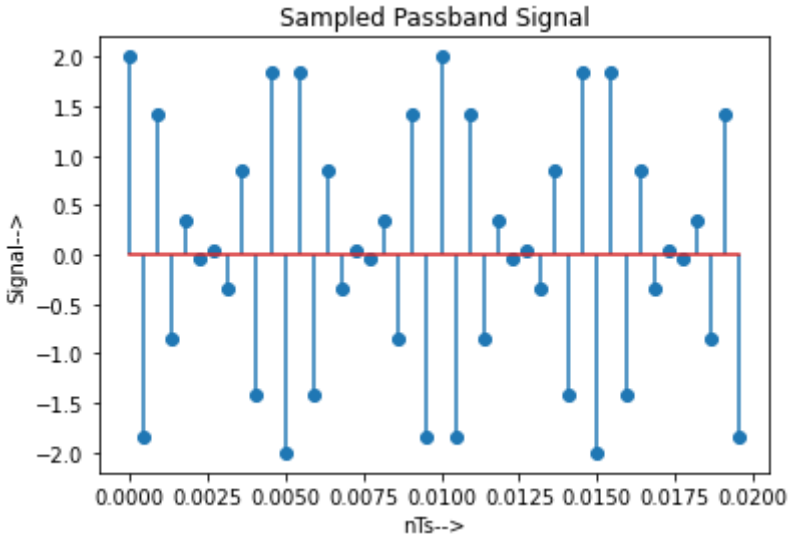


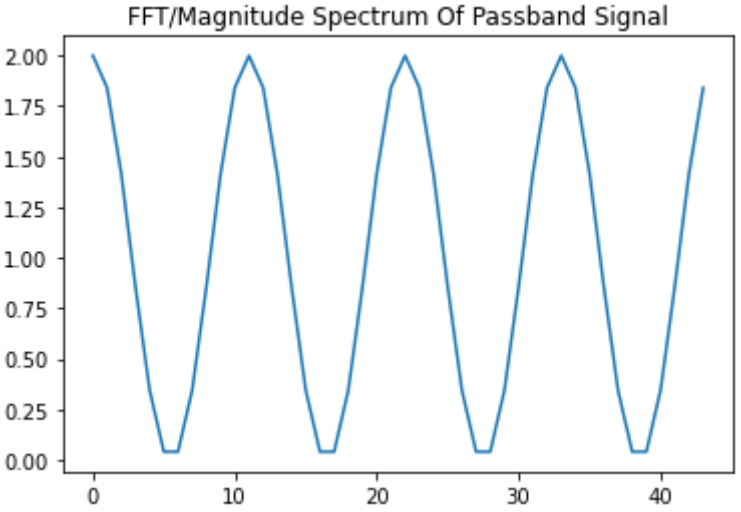


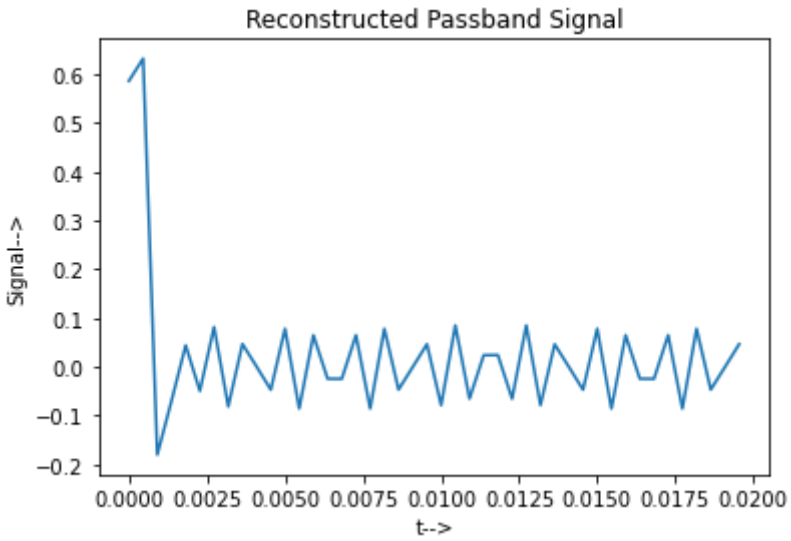


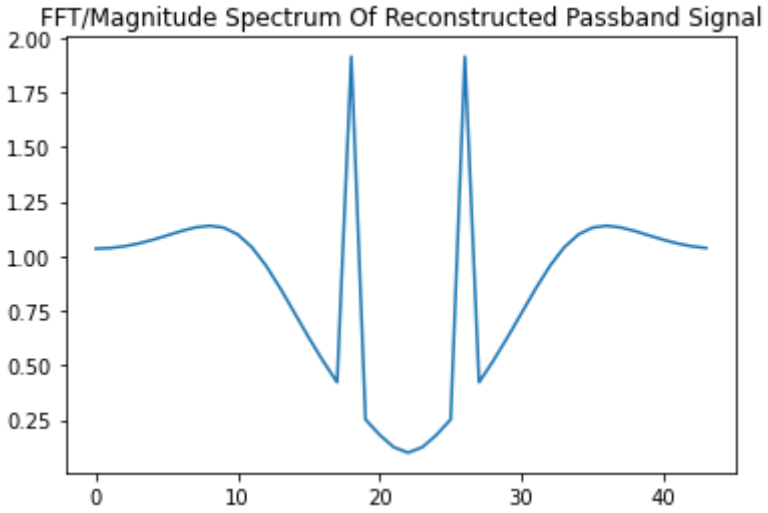
Case 3 : Perfect Sampling (M=2)


Passband Signal
Signal-->
t-->
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200


Sampled Passband Signal
Signal-->
nTs-->
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200


FFT/Magnitude Spectrum Of Passband Signal
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
30
40





Observations :-

With respect to the simulation and output graphs that were received, the following observations could be made regarding the sampling and reconstruction of Baseband/Bandlimited signals and Passband signals,

(i) if the sampling frequency fs is greater than 2*maximum frequency of signal (fs>2*fmax), then, over sampling happens and we can retrieve or reconstruct the signal back properly and with minimum error.

(ii) if the sampling frequency is equal to 2*maximum frequency of signal (fs=2*fmax), then, perfect sampling happens and retrieval or reconstruction of signal is a bit difficult and possibility of error may occur.

(iii) if the sampling frequency is less than 2*maximum frequency of signal (fs<2*fmax), then, under sampling happens and we cannot retrieve or reconstruct the signal back and there is a high possibility of getting very high error.

Results :-

In this experiment, we write a Python program to plot a signal, its sampled version, the fourier transform of the signal, the reconstructed signal and the fourier transform of the reconstructed signal. The graphs were obtained by using basic python inbuilt functions like “stem”, “plot”,”buttord”,”butter”,”lfilter” and so on, from the python modules, “numpy”,”matplotlib.pyplot”, “scipy”,etc. The simulation starts by the creation of a signal, then plotting the signal, then sampling the signal, plotting the the sampled signal, finding the fourier transform of the signal, then, reconstructing the sampled signal through filtration, and finally finding the plot for fourier transform of the reconstructed signal.

Conclusion :-

A Python program was successfully written and implemented in Anaconda’s Spyder Software Simulator to generate a signal, and get the plots of the signal, the sampled version of the signal, the fourier transform of the signal, the reconstructed signal and finally, the fourier transform of the reconstructed signal.