Name : Sahil Yadav
Roll No. : 21ECE1039
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,

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

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

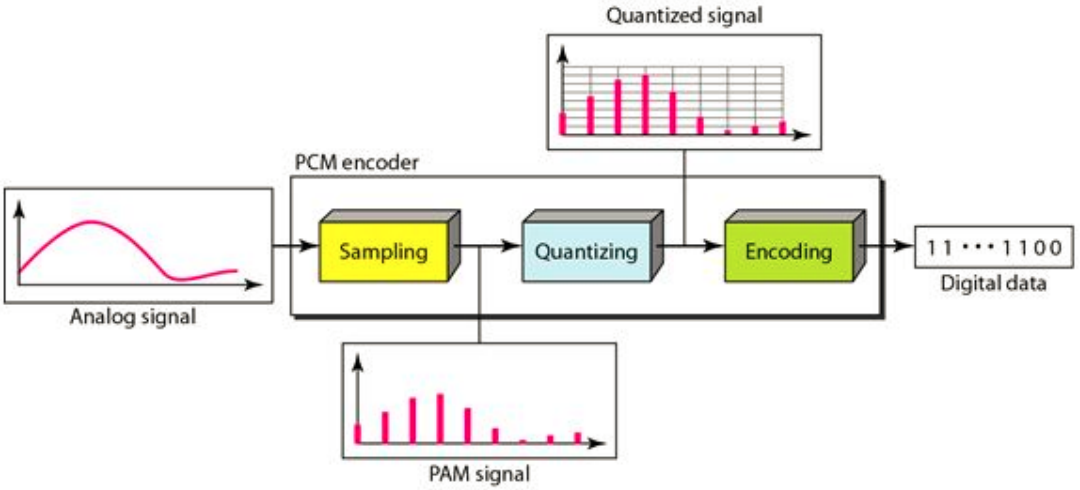
(t)

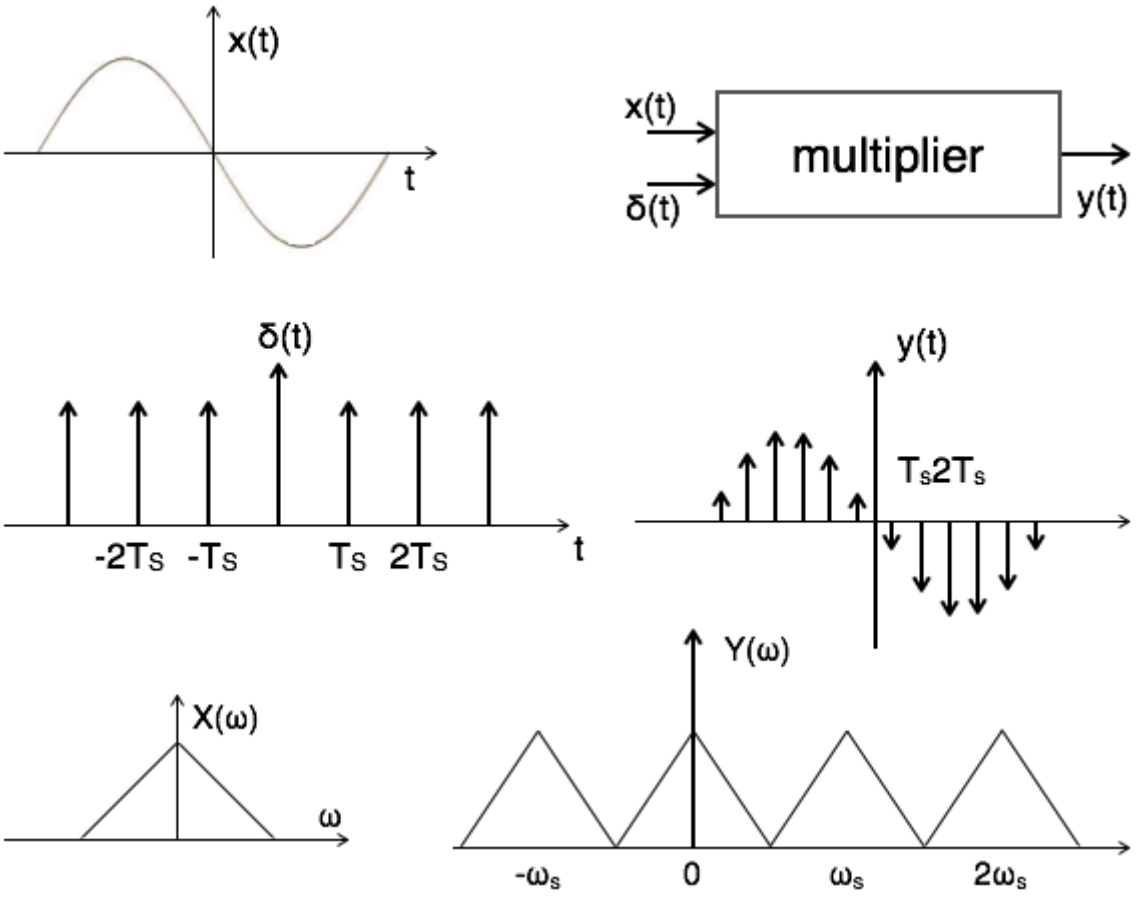
its reciprocal $1/T = F$ is called the sampling rate. The desired relationship between

s

the spectrum X(F) of the discrete-time signal and the spectrum X is given by,

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


Quantized signal
PCM encoder
Sampling
Quantizing
Encoding
11 • • • 1100
Digital data
Analog signal
PAM signal


x(t)
t
x(t)
δ(t)
multiplier
y(t)
δ(t)
-2Ts -Ts Ts 2Ts
t
y(t)
Ts2Ts
Y(ω)
X(ω)
ω
-ωs 0 ωs 2ωs

Derivation :

The mathematical expression of sampled signal can be given as,

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

The unit impulse function δ(t) can be represented in terms of Fourier Series as

$$\delta(t) = a0 + \Sigma\infty n = 1(ancosn\omega st + bnsinn\omega st)\ldots\ldots (2)$$

Where $a0 = 1Ts\int T2 - T2\delta(t)dt = 1Ts\delta(0) = 1Ts$

$$an = 2Ts\int T2 - T2\delta(t)cosn\omega sdt = 2T2\delta(0)cosn\omega s0 = 2T$$

$$bn = 2Ts\int T2 - T2\delta(t)sinn\omega stdt = 2Ts\delta(0)sinn\omega s0 = 0$$

Substitute above values in equation 2.

$$\therefore\delta(t) = 1Ts + \Sigma\infty n = 1(2Tscosn\omega st + 0)$$

Substitute δ(t) in equation 1.

$$\begin{aligned}\rightarrow y(t) &= x(t).\delta(t)\\ &= x(t)[1Ts + \Sigma\infty n = 1(2Tscosn\omega st)]\\ &= 1Ts[x(t) + 2\Sigma\infty n = 1(cosn\omega st)x(t)]\end{aligned}$$

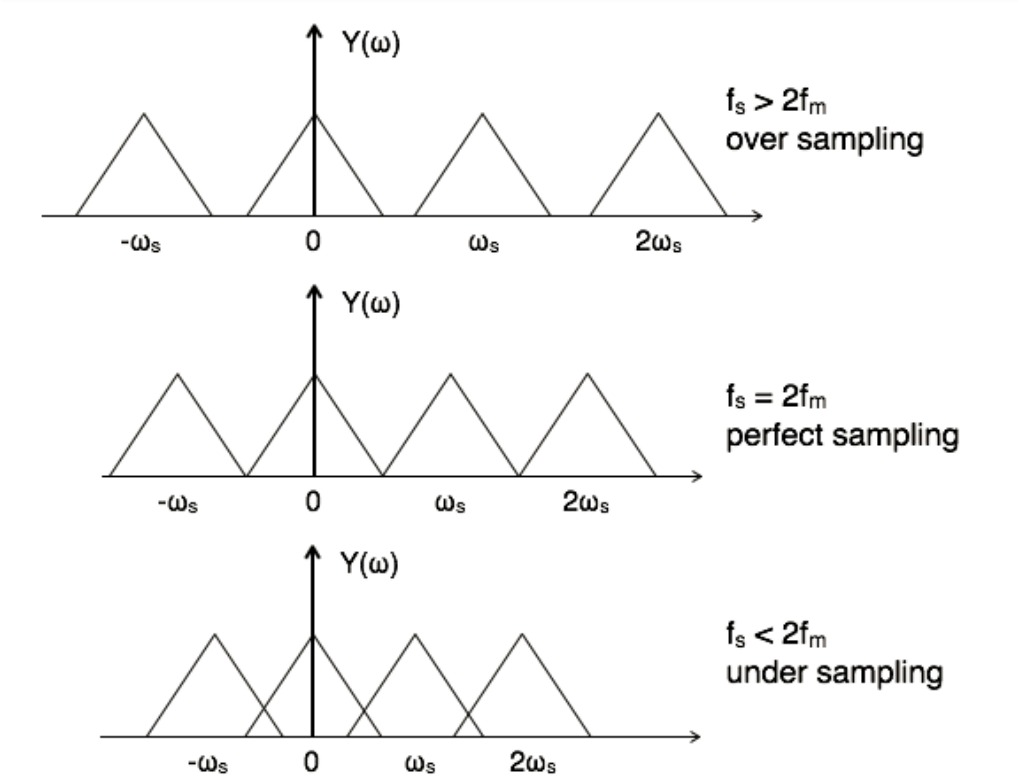
$$y(t) = 1Ts[x(t) + 2cos\omega st.x(t) + 2cos2\omega st.x(t) + 2cos3\omega st.x(t)\ldots\ldots]$$

Take Fourier transform on both sides

$$Y(\omega) = 1Ts[X(\omega) + X(\omega - \omega s) + X(\omega + \omega s) + X(\omega - 2\omega s) + X(\omega + 2\omega s) + \ldots]$$

$$\therefore Y(\omega) = 1Ts\Sigma\infty n = -\infty X(\omega - n\omega s)wheren = 0, \pm 1, \pm 2,\ldots$$

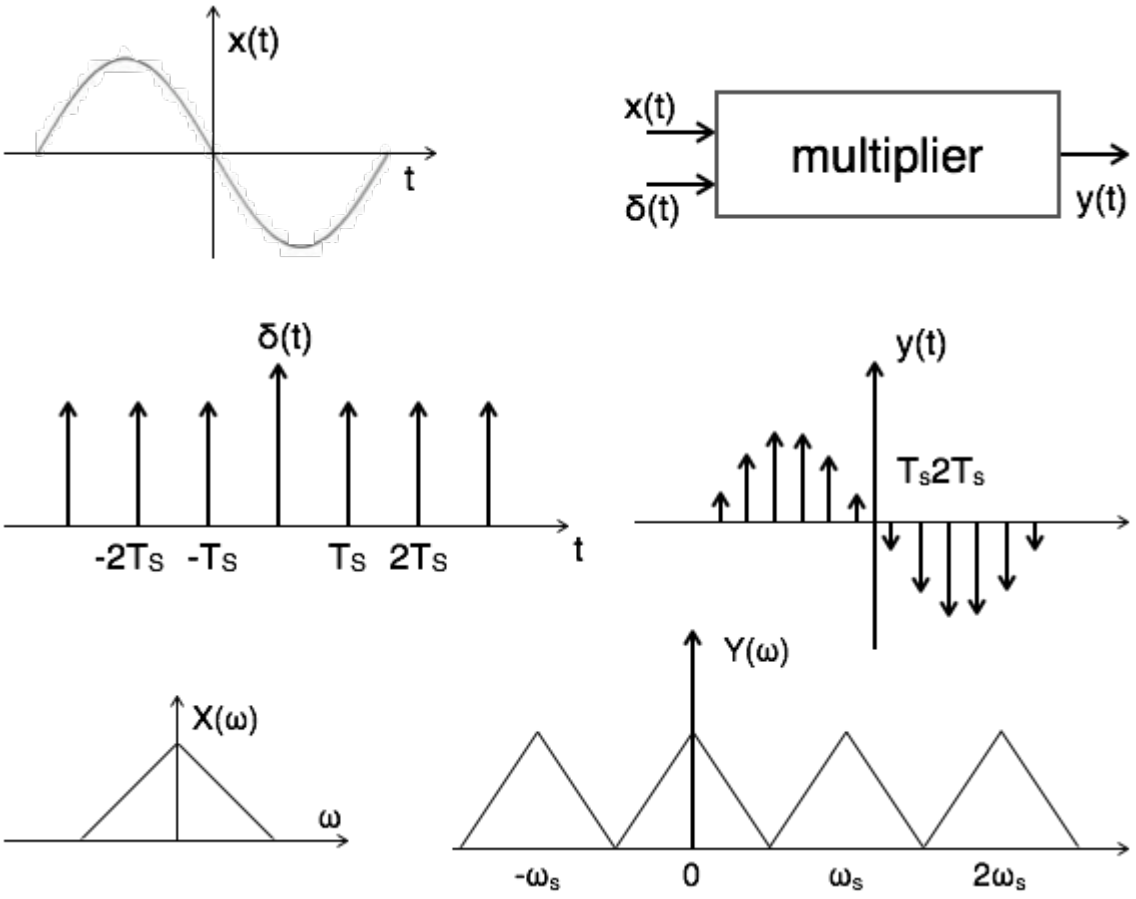
Possibility of sampled frequency spectrum with different conditions is given by the following diagrams:



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

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 has its frequency content in the two frequency bands defined by



Aliasing Effect

The overlapped region in case of under sampling represents aliasing effect, which can be removed by

- considering fs >2fm
- By using anti aliasing filters.

Signals Sampling Techniques

There are three types of sampling techniques:

- Impulse sampling.
- Natural sampling.
- Flat Top sampling.

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 xS(n) is the sampled signal
3. Determine the DFT X s (K) of the sampled signal xS(n).
4. Pass the sampled signal through a low pass filter of cut-off frequency fS/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 :-

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

# Bandlimited/Bandpass 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.

# Time of baseband signal
t = np.arange(0, 5 * T, 0.1 * T)

# Sampling time of baseband signal
ts = np.arange(0, 5 * T, 1 / fs)

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

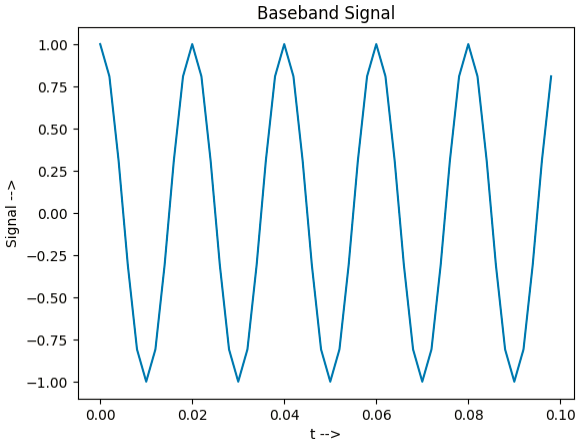
```
# Plotting baseband signal
plt.plot(t, x)
plt.title("Baseband Signal")
plt.xlabel("t -->")
plt.ylabel("Signal -->")
plt.show()

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

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

# FFT of baseband signal
x_f = fft(x)

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

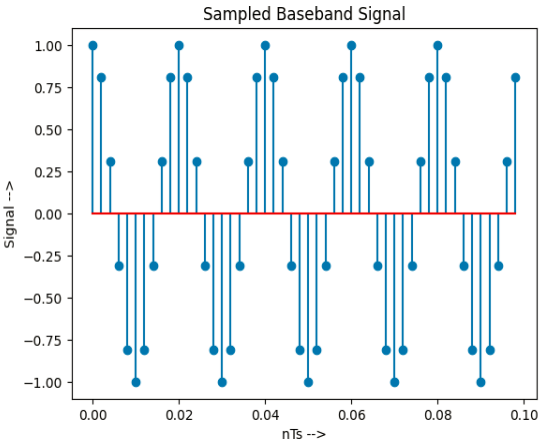
```
# Pass-band and stop-band frequencies
wp = (fs / 2) / fs
ws = (fs - 10) / fs

# Pass-band and stop-band attenuations
apb = 3  # pass-band attenuation in dB
asb = 60  # stop-band attenuation in dB

# Determining order and cut-off frequency of low-pass filter
[N, Wn] = signal.buttord(wp, ws, apb, asb)

# Finding constants for low-pass filter
b, a = signal.butter(N, Wn, 'low')

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

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

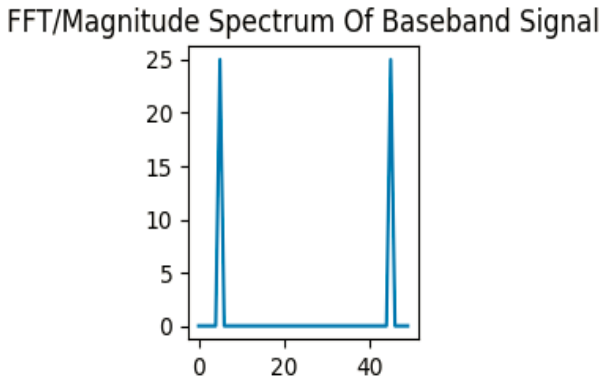
```
# FFT of reconstructed baseband signal
xr_f = fft(xr)

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

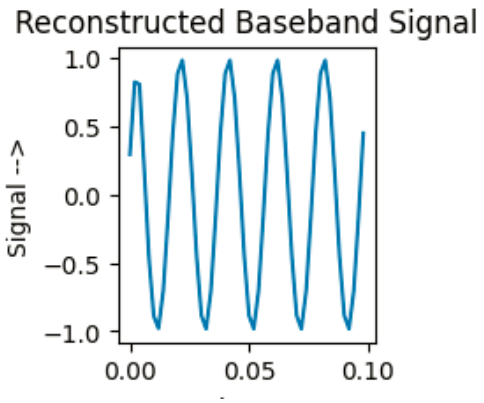
Graphs :-

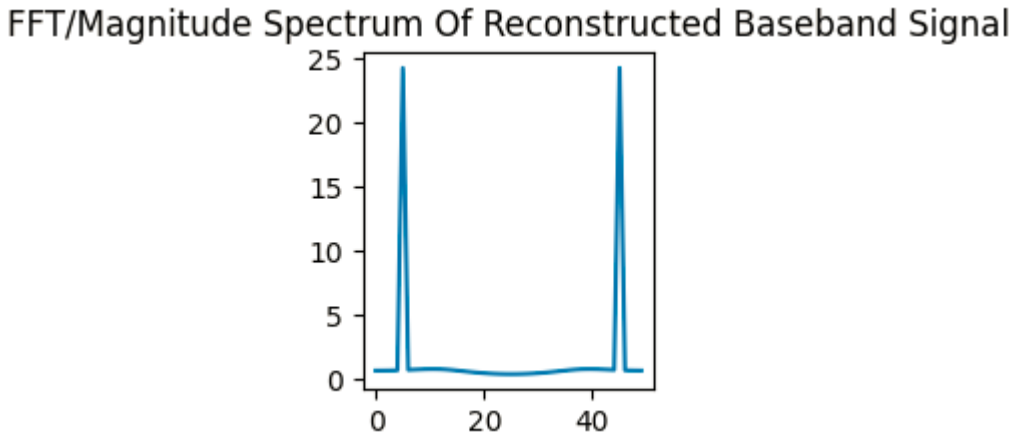
(i) Bandlimited/Bandpass Signal :











(ii) Passband Signal :

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

# Passband 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.

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

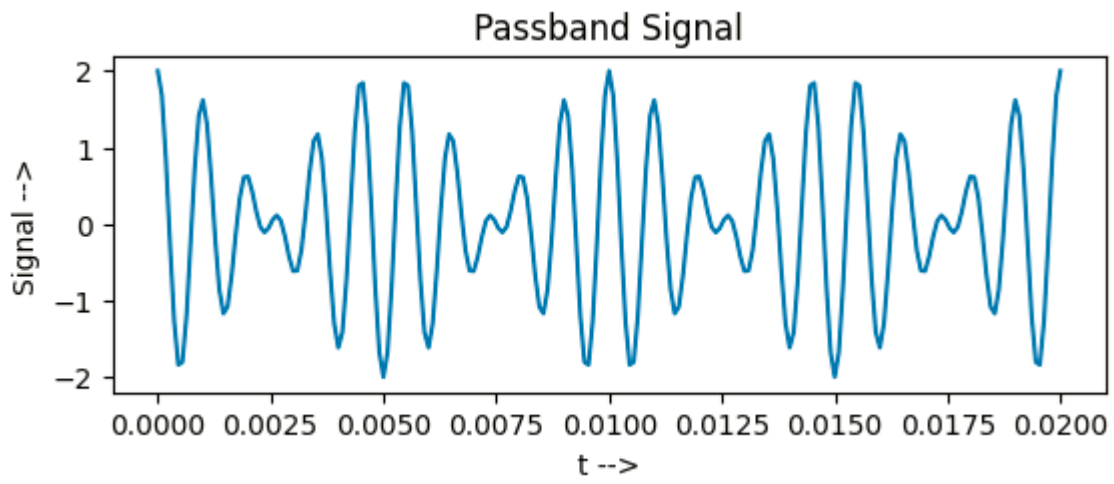
```
# Passband signal and carrier signal construction
x = A * np.cos(2 * np.pi * f * t)
xc = Ac * np.cos(2 * np.pi * fc * t)

# Resultant passband signal (DSBSC)
X = np.multiply(x, xc)

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

# Sampled passband signal construction
xs = A * np.cos(2 * np.pi * f * ts)
xc_s = Ac * np.cos(2 * np.pi * fc * ts)

# Resultant sampled passband signal
Xs = np.multiply(xs, xc)
```

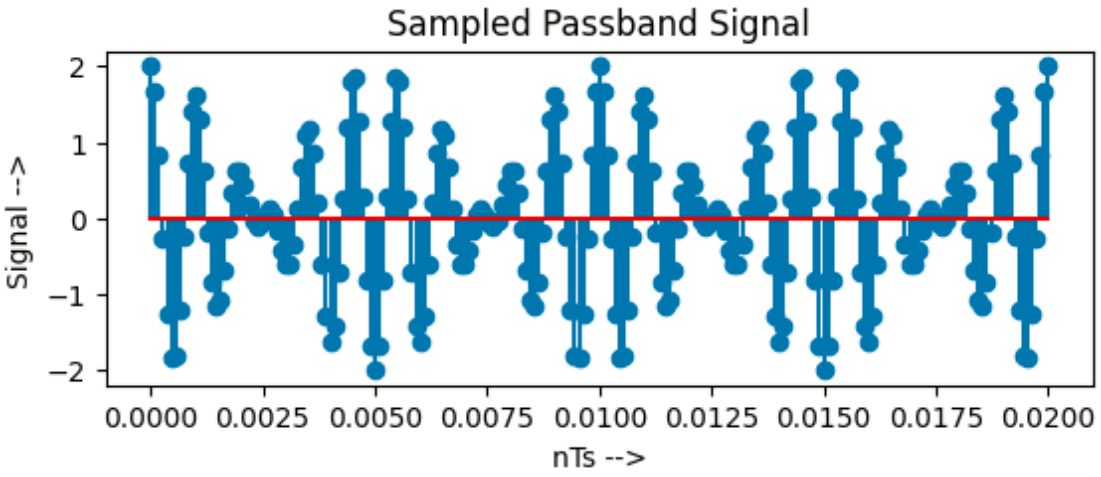
```
# Plotting sampled passband signal
plt.stem(ts, Xs)
plt.title("Sampled Passband Signal")
plt.xlabel("nTs -->")
plt.ylabel("Signal -->")
plt.show()

# FFT of passband signal
X_f = fft(X)

# Plotting FFT/Magnitude Spectrum of passband signal
plt.plot(abs(X_f))
plt.title("FFT/Magnitude Spectrum Of Passband Signal")
plt.show()

# Pass-band and stop-band frequencies
wp = (fs / 2) / fs
ws = (fs - 10) / fs

# Pass-band and stop-band attenuations
apb = 3  # pass-band attenuation in dB
asb = 60  # stop-band attenuation in dB

# Determining order and cut-off frequency of low-pass filter
[N, Wn] = signal.buttord(wp, ws, apb, asb)
```

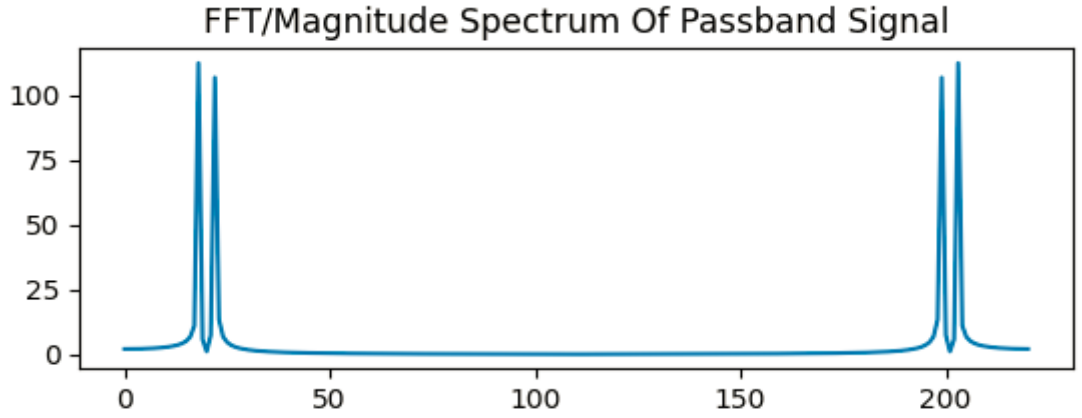
```
# Finding constants for low-pass filter
b, a = signal.butter(N, Wn, 'low')

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

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

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

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

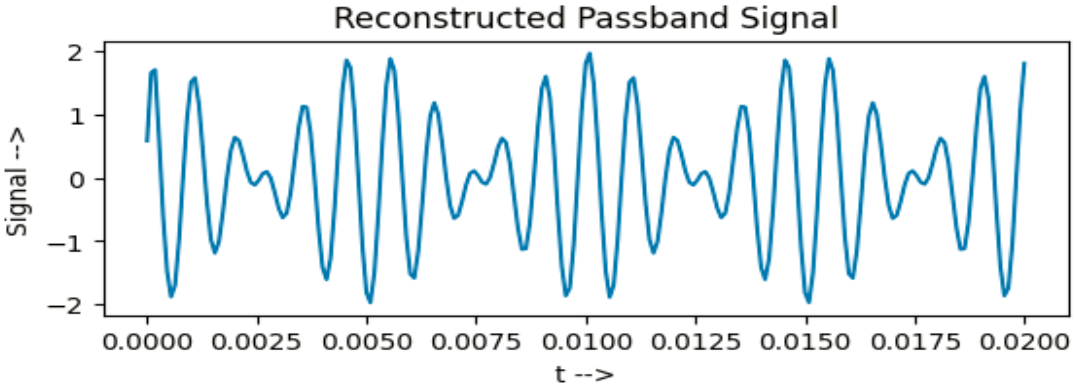
Graphs :-

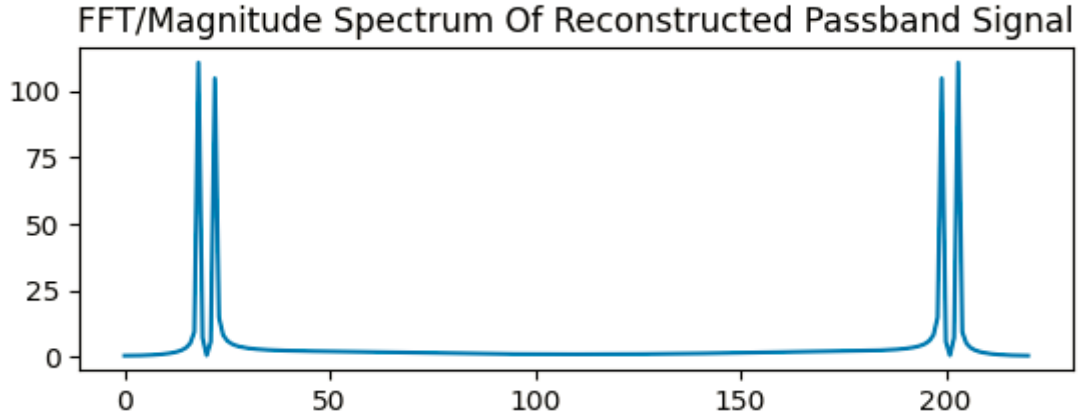
(ii) Passband Signal :




Sampled Passband Signal
Signal -->
nTs -->
2
1
0
−1
−2
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200


FFT/Magnitude Spectrum Of Passband Signal
100
75
50
25
0
0
50
100
150
200

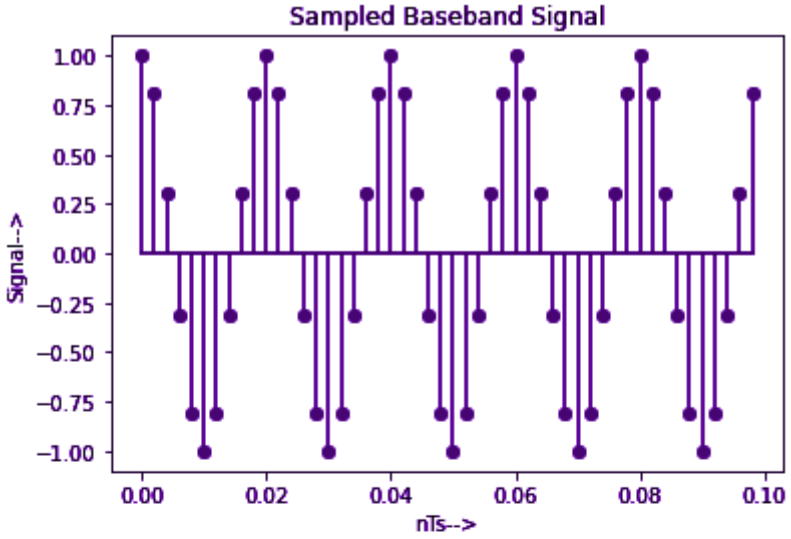

Reconstructed Passband Signal
Signal -->
t -->
2
1
0
−1
−2
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200

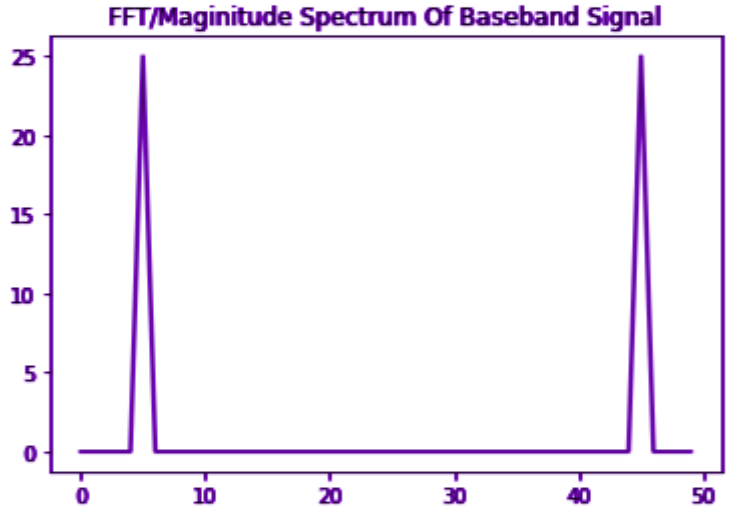

FFT/Magnitude Spectrum Of Reconstructed Passband Signal
100
75
50
25
0
0
50
100
150
200

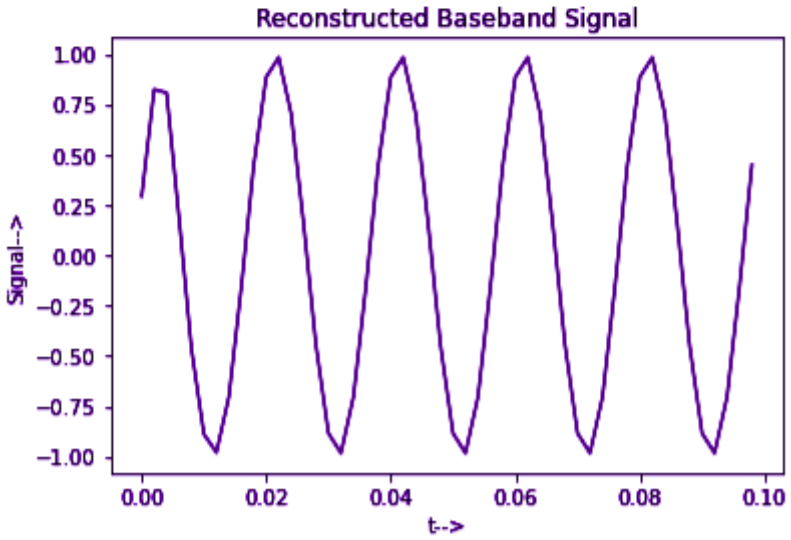
Graphs :-

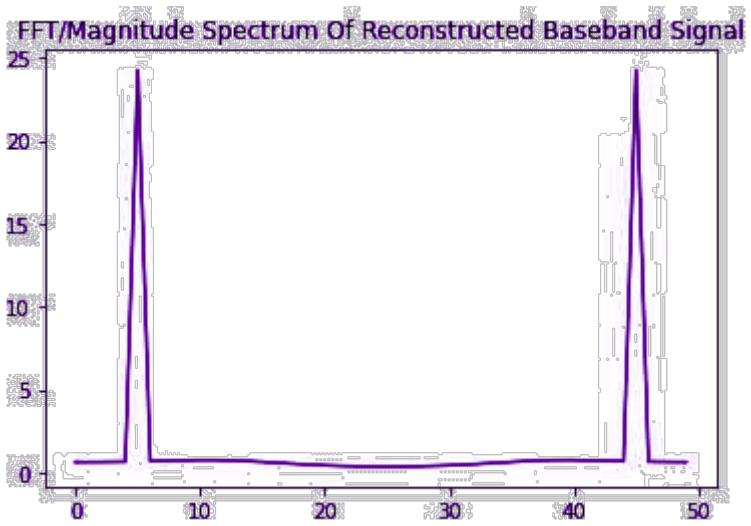
(i) Bandlimited/Bandpass Signal :

Oversampling (M=12)



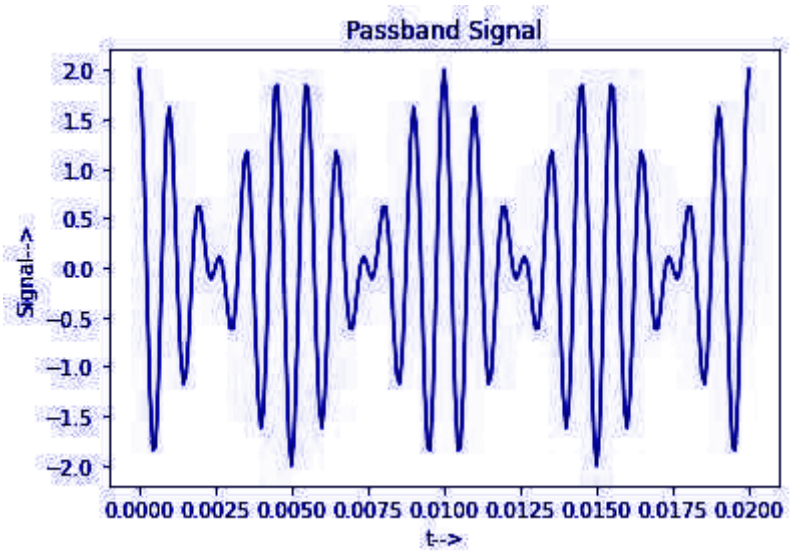


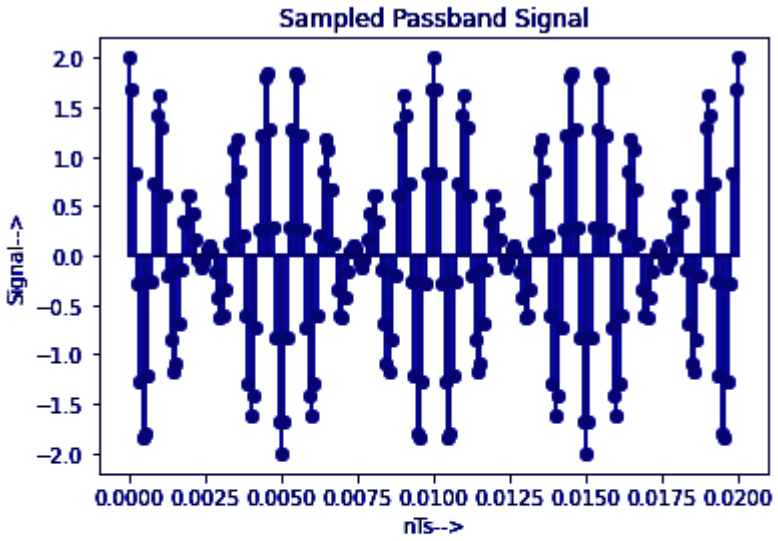


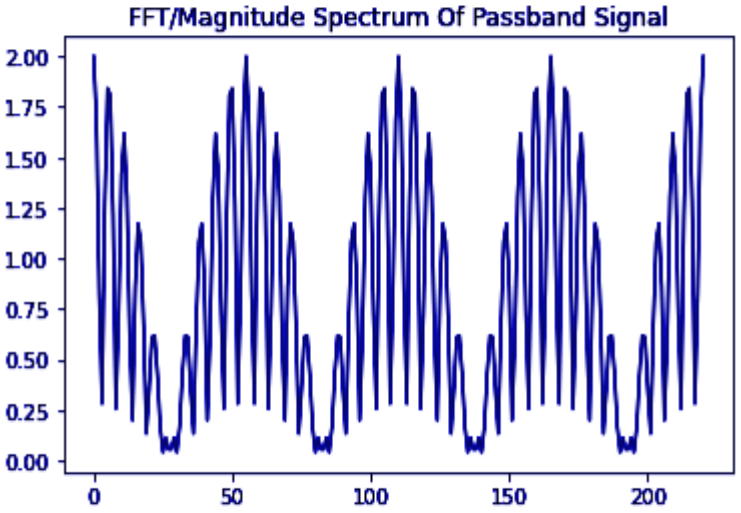


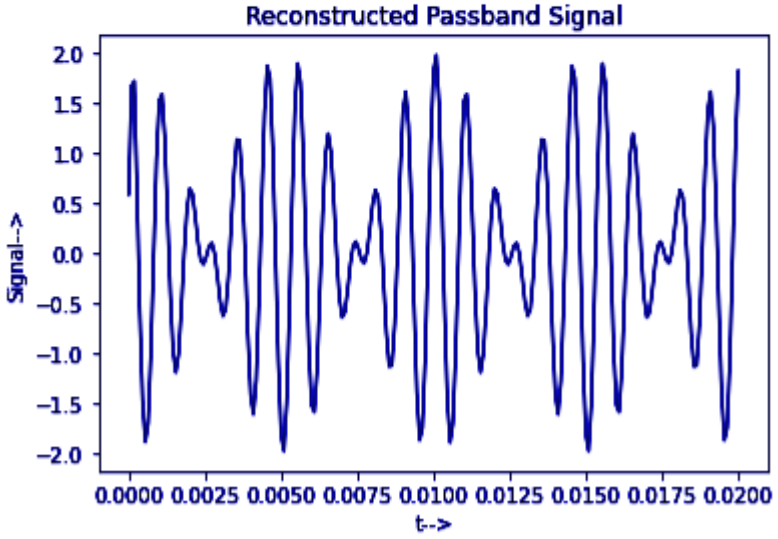
(ii) Passband Signal :

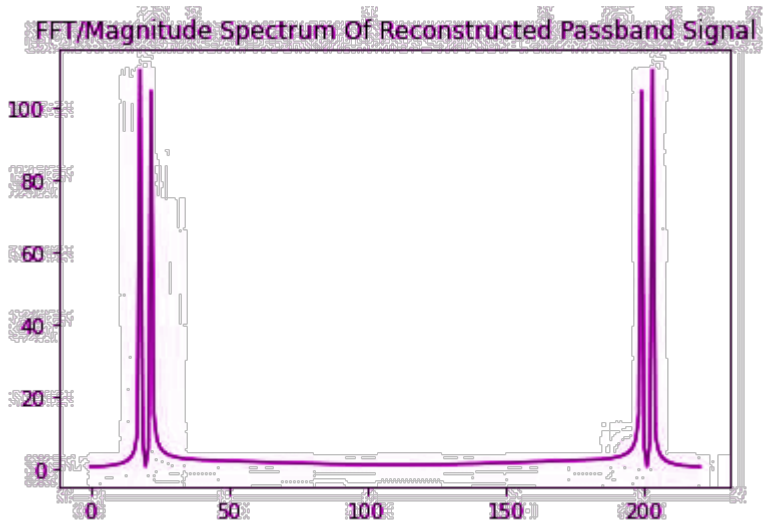
Case 1 : Oversampling (M=12)



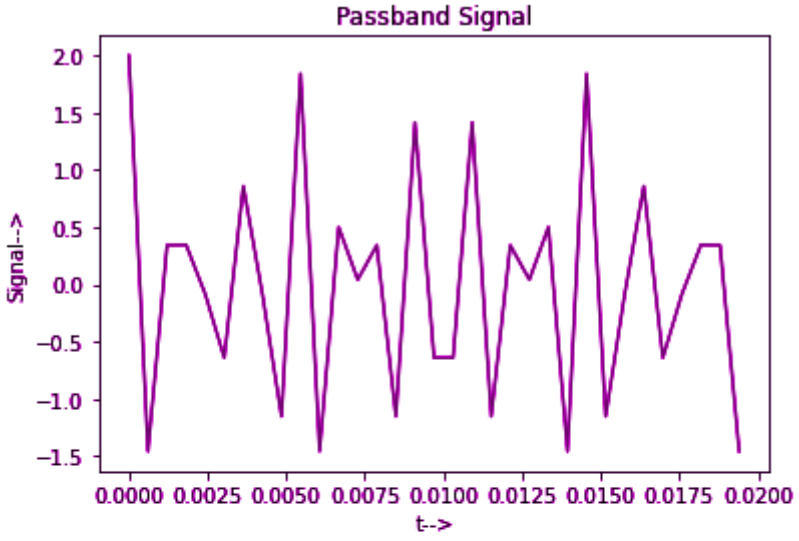

Sampled Passband Signal
Signal-->
nTs-->
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200

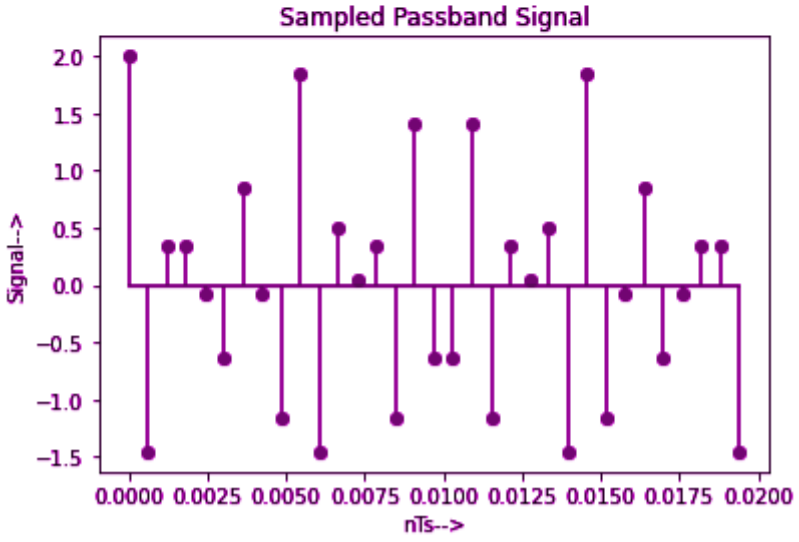

FFT/Magnitude Spectrum Of Passband Signal
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
50
100
150
200

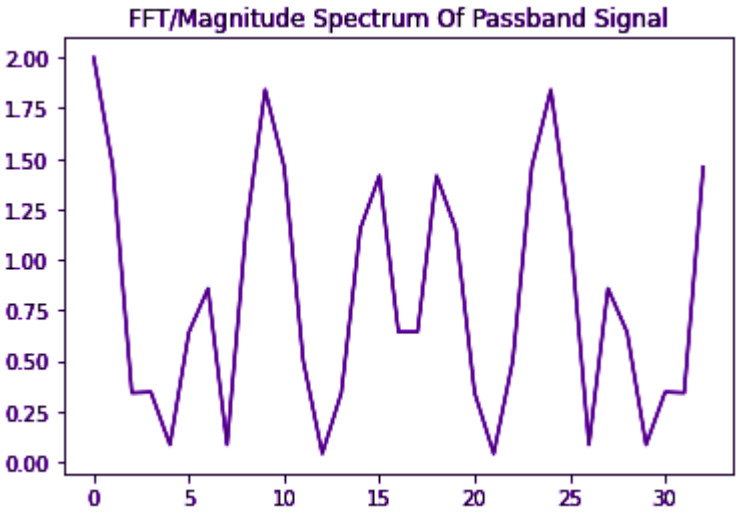

Reconstructed Passband Signal
Signal-->
t-->
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0000 0.0025 0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200

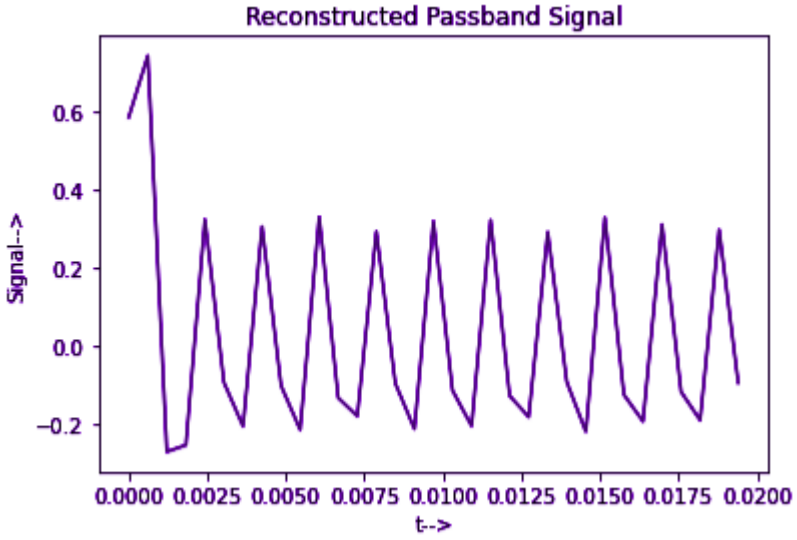


Case 2 : Undersampling (M=1.2)

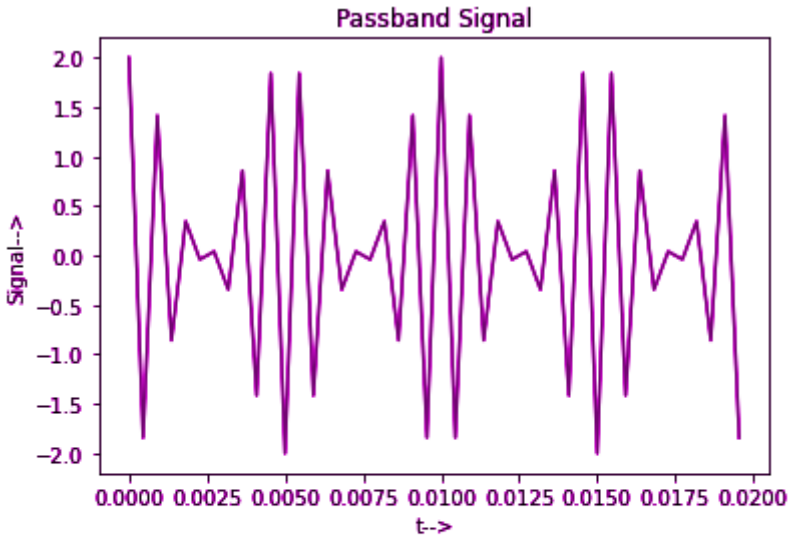


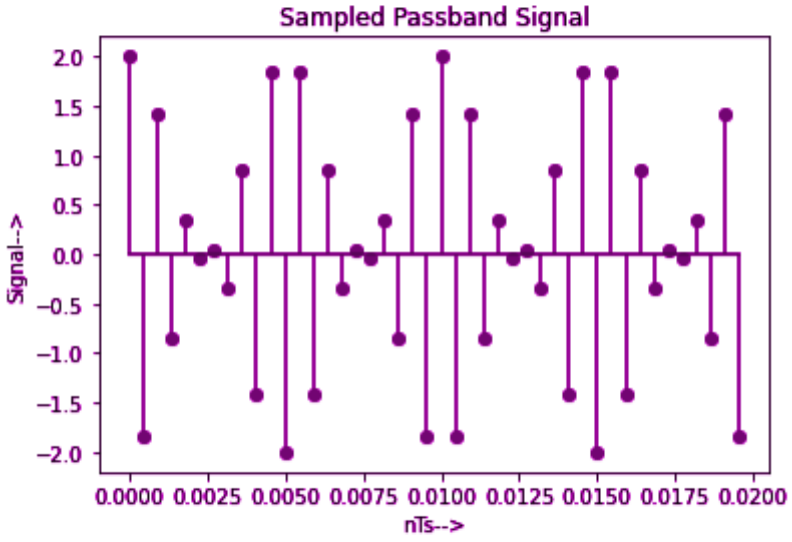


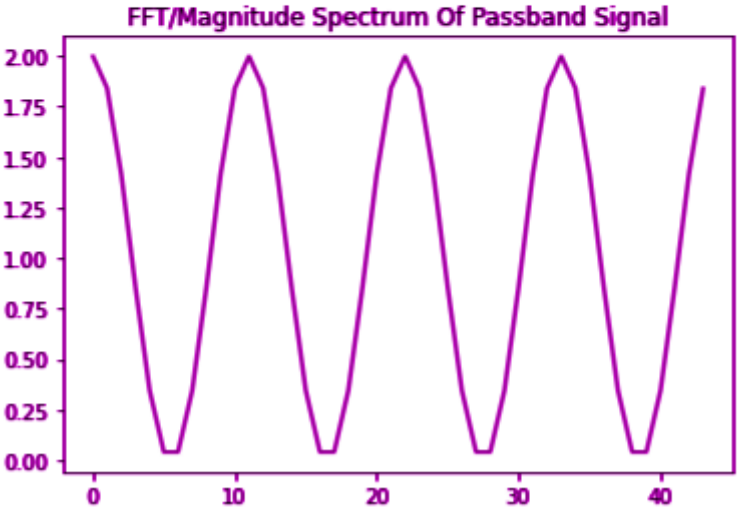


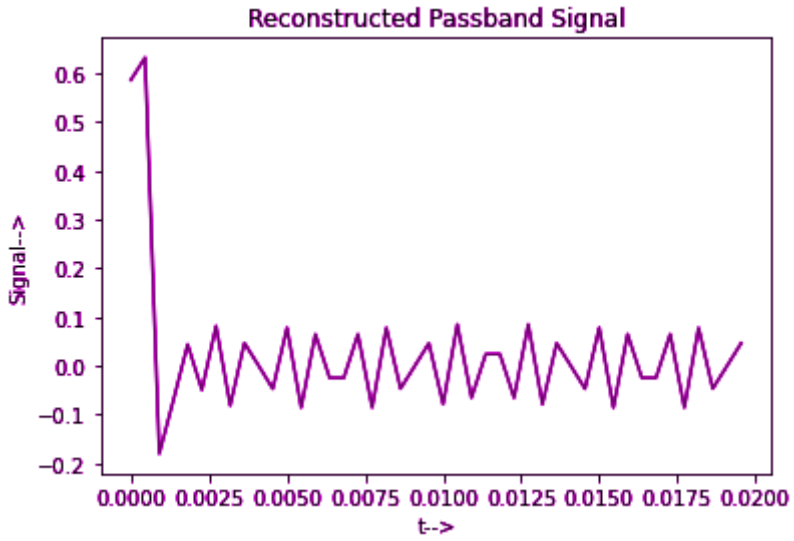


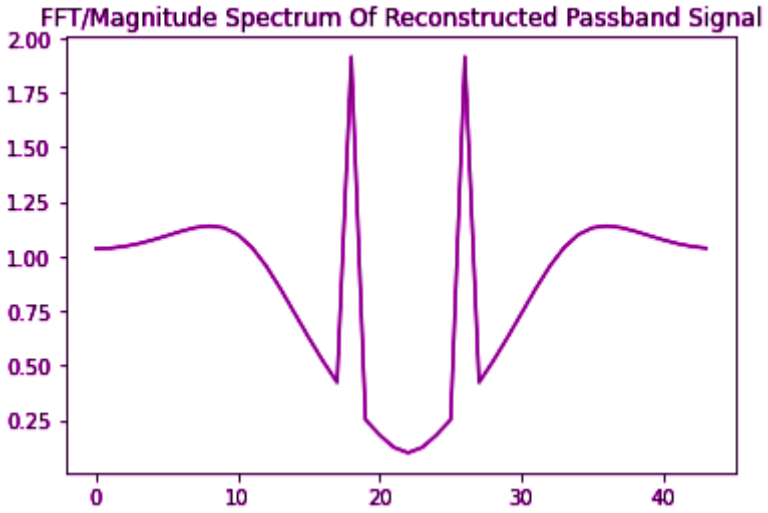
Case 3 : Perfect Sampling (M=2)


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 :-

In this simulation experiment the output graphs that were obtained, the following observations on the reconstruction and sampling of Baseband/Bandlimited signals and Passband signals may be made:

- Oversampling occurs when the sampling frequency (fs) exceeds the maximum frequency of the signal (fs>2*fmax), allowing for accurate and error-free signal reconstruction.
- Perfect sampling occurs when the sampling frequency equals the signal's maximum frequency (fs=2*fmax). This makes signal reconstruction or retrieval challenging and increases the chance of error.
- Undersampling occurs when the sampling frequency is less than the signal's 2*maximum frequency (fs<2*fmax), making it impossible to recover or reconstruct the signal and increasing the likelihood of extremely high error.

Results :-

In this experiment we found that in order to plot a signal, its sampled version, the signal's fourier transform, the reconstructed signal, and the fourier transform of the rebuilt signal, we developed a Python program for this experiment. The graphs were created using standard built-in Python functions from the python modules "numpy," "matplotlib.pyplot," "scipy," and so forth, such as "stem," "plot," "buttord," "butter," and so on. The simulation begins with the creation of a signal, which is then plotted, sampled, plotted again, and the fourier transform of the sampled signal is found. Next, the sampled signal is reconstructed through filtering, and the plot for the fourier transform of the reconstructed signal is found.

.

Conclusion :-

In conclusion , this experiment was done to learn how to generate a signal, obtain its plots, sampled version, fourier transform, reconstructed signal, and finally, fourier transform of the reconstructed signal, a Python program was successfully written and implemented in Anaconda's Spyder Software Simulator.