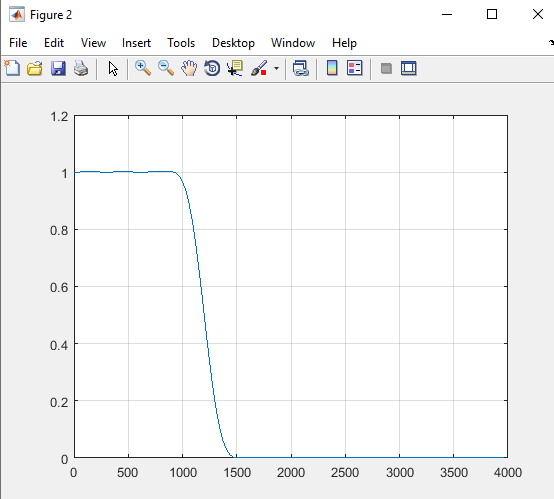
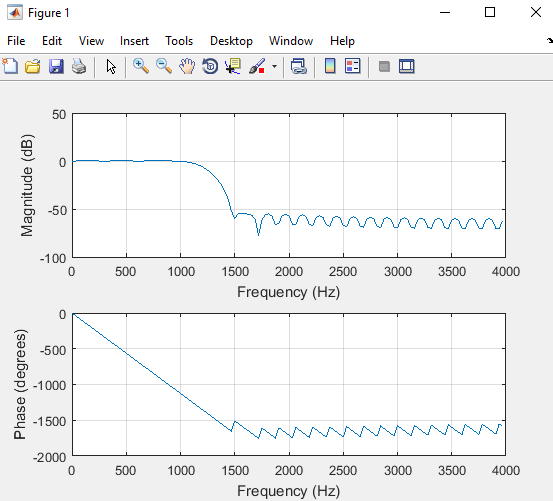
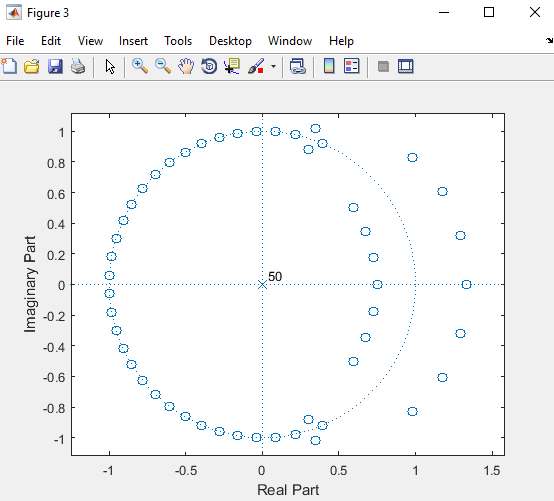
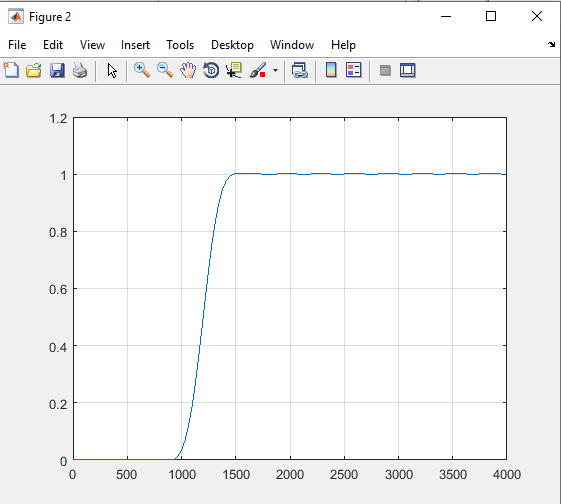
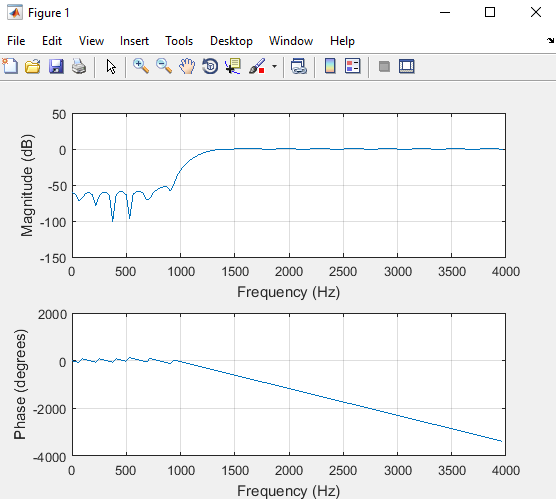
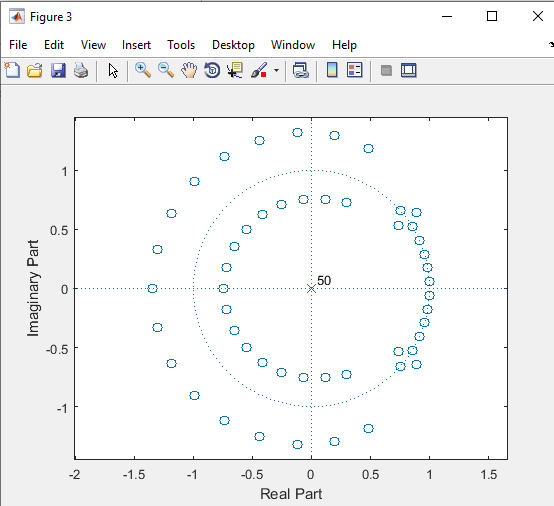
**Output:**





**Output:**





**Experiment No: 03**

**Experiment Name: Write a program to designing a Band Pass Filter.**

**Objectivese:**

1. To allow signals of a specific frequency range to pass through while attenuating signals at other frequencies.
2. To eliminate unwanted noise and interference from a signal.
3. To provide better signal-to-noise ratio (SNR) and improve the quality of the signal.
4. To prepare a filter with a specific bandwidth, gain, and frequency response for a specific application.

**Theory:**

A bandpass filter is a circuit that allows a specific range of frequencies to pass through while attenuating signals at frequencies outside that range. The filter consists of a combination of passive components such as resistors, capacitors, and inductors, and active components such as operational amplifiers. The design of a bandpass filter depends on the desired frequency range, gain, and bandwidth.

The frequency range of a bandpass filter is determined by the center frequency and the bandwidth. The center frequency is the frequency at which the filter has maximum gain, and the bandwidth is the range of frequencies that the filter allows to pass through. The bandwidth is typically defined as the difference between the upper and lower cutoff frequencies, which are the frequencies at which the filter's gain is 3 dB lower than the maximum gain.

There are two types of bandpass filters: passive and active. Passive bandpass filters consist of only passive components and do not require an external power source, while active bandpass filters use operational amplifiers and require a power supply.

**Matlab Code:**

fs=8000;

n=40;

b=fir1(n,[1200/4000 1800/4000],’bandpass’);

freqz(b,1,128,8000)

figure(2)

[h,w]=freqz(b,1,128,8000);

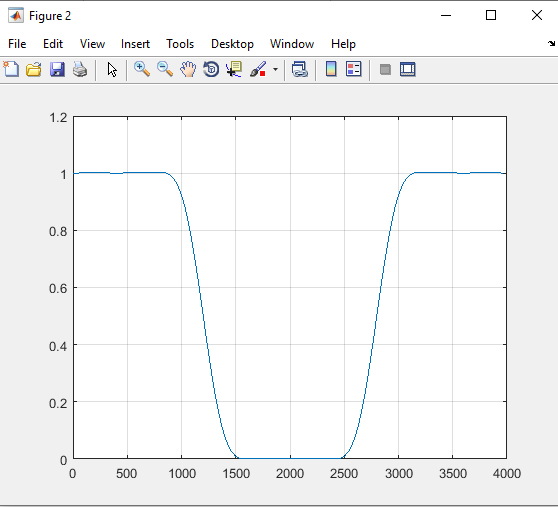
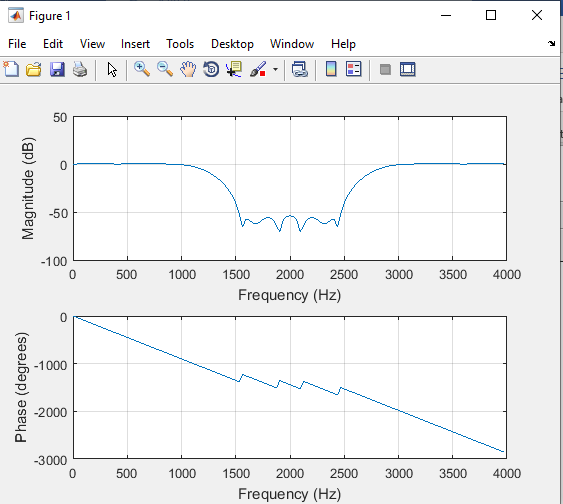
plot(w,abs(h)); % Normalized Magnitude Plot

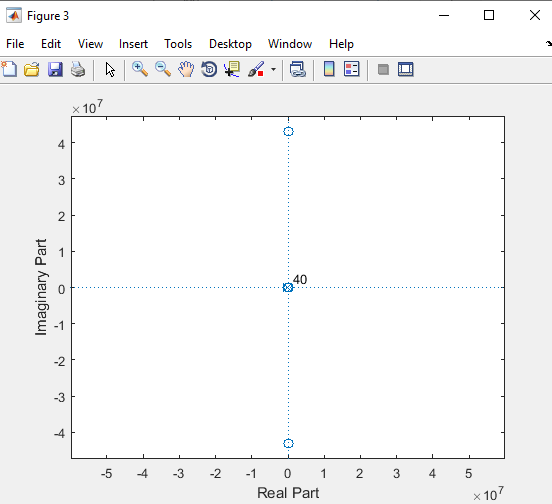
grid

figure(3)

zplane(b,1);

**Output:**





**Experiment No: 04**

**Experiment Name: Write a program to designing a Notch Pass Filter.**

**Objectivese:**

1. To attenuate or eliminate a specific frequency range from a signal while allowing other frequencies to pass through.
2. To reduce or eliminate unwanted noise and interference from a signal.
3. To improve the signal-to-noise ratio (SNR) and overall quality of the signal.
4. To prepare a filter with a specific bandwidth, gain, and frequency response for a particular application.

**Theory:**

A notch filter is a type of filter that attenuates a specific frequency range while allowing other frequencies to pass through. It is essentially the opposite of a bandpass filter. The notch filter can be designed using passive components such as resistors, capacitors, and inductors or active components such as operational amplifiers.

The frequency range that is attenuated by the notch filter is known as the stopband, and the frequencies that are allowed to pass through are known as the passband. The stopband can be either a narrow or wide range of frequencies, depending on the application. The notch filter's performance is determined by the center frequency of the stopband and the bandwidth of the stopband.

Passive notch filters are usually designed as second-order filters, which can be achieved using a combination of a capacitor and an inductor or a capacitor and a resistor. The second-order notch filter has a roll-off rate of -12 dB/octave, which means that the attenuation of the stopband increases by 12 dB for every octave increase in frequency beyond the cutoff frequency.

**Matlab code:**

fs=8000;

n=40;

b=fir1(n,[1500/4000 1550/4000],'stop');

freqz(b,1,128,8000)

figure(2)

[h,w]=freqz(b,1,128,8000);

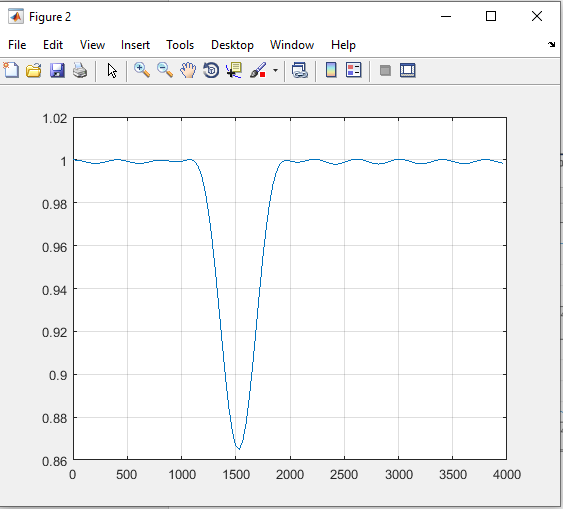
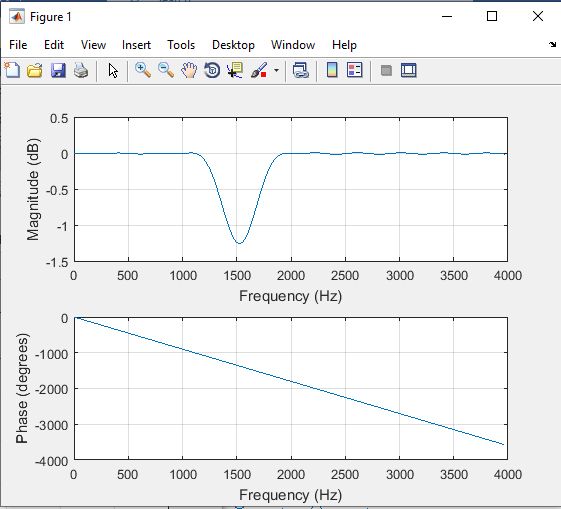
plot(w,abs(h)); % Normalized Magnitude Plot

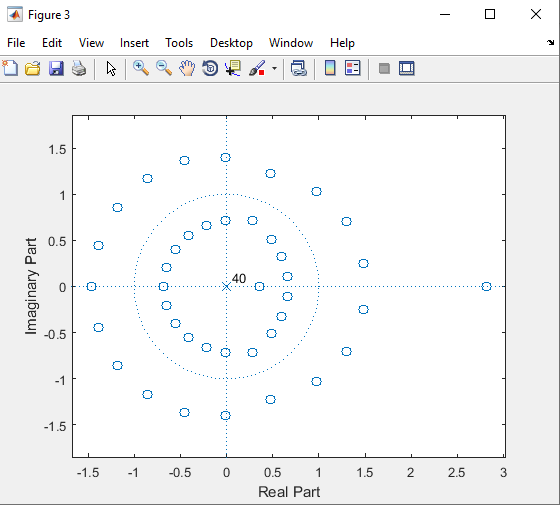
grid

figure(3)

zplane(b,1);

**Output:**





**Experiment No: 05**

**Experiment Name: Write a program to designing a Multi Pass Filter.**

**Objectives:**

1. To provide multiple stages of filtering to achieve a specific frequency response and gain.
2. To attenuate or eliminate unwanted frequencies and noise from a signal while allowing desired frequencies to pass through.
3. To improve the signal-to-noise ratio (SNR) and overall quality of the signal.
4. To prepare a filter with a specific bandwidth, gain, and frequency response for a particular application.

**Theory:**

A multipass filter is a type of filter that consists of multiple stages of filtering, each stage consisting of one or more filter elements. The filter elements can be passive components such as resistors, capacitors, and inductors or active components such as operational amplifiers. The design of a multipass filter involves selecting the appropriate filter elements and configuration to achieve the desired frequency response and gain.

The advantage of using a multipass filter is that it can achieve a higher order filter response, which means that it can provide steeper roll-off rates and a narrower transition band than a single-stage filter. The order of the filter is determined by the number of stages in the filter, with each stage contributing a certain amount of roll-off to the overall filter response.

There are two types of multipass filters: cascaded and parallel. Cascaded filters are connected in series, with the output of one stage connected to the input of the next stage. The advantage of cascaded filters is that they can achieve a higher order filter response than a single-stage filter. The disadvantage is that they can introduce additional phase shift and attenuation of the signal.

**Matlab code:**

n=50;

w=[0.2 0.4 0.6];

b=fir1(n,w);

freqz(b,1,128,8000)

figure(2)

[h,w]=freqz(b,1,128,8000);

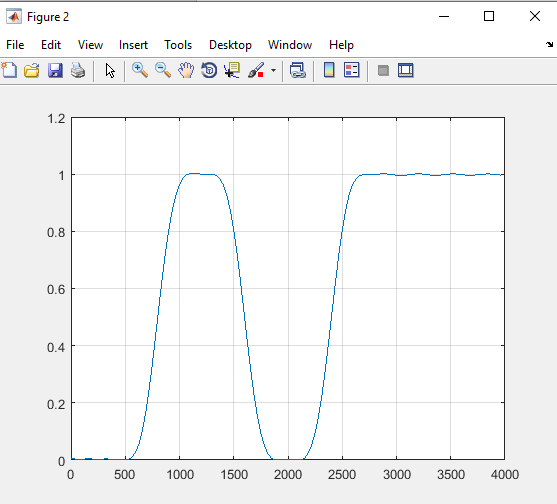
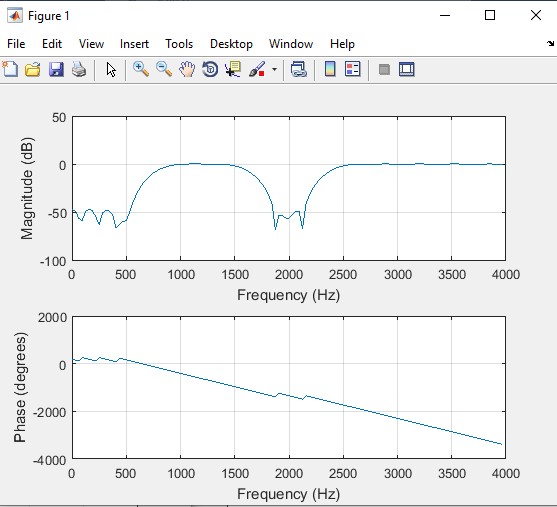
plot(w,abs(h)); % Normalized Magnitude Plot

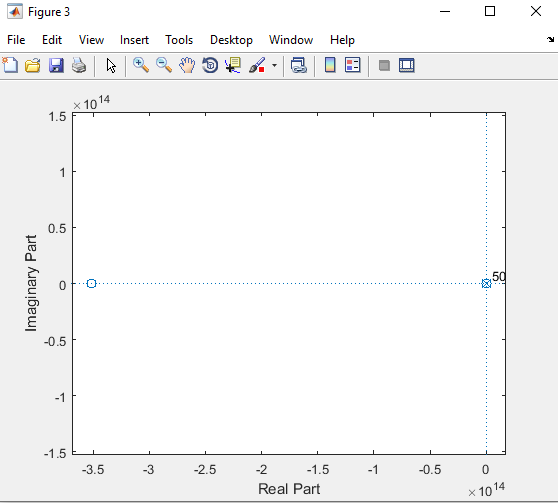
grid

figure(3)

zplane(b,1);

**Output:**





**Experiment No: 06**

**Experiment Name: Write a program to designing a IIR Low Pass Filter.**

**Objectives:**

1. To achieve a specific frequency response and gain for a particular application.
2. To improve the signal-to-noise ratio (SNR) and overall quality of the signal.
3. To design a filter with a sharp roll-off rate and minimal phase distortion.

**Theory:**

An IIR low-pass filter is a type of filter that uses feedback to achieve a specific frequency response. The filter is called "infinite impulse response" because the impulse response of the filter does not decay to zero, unlike a finite impulse response (FIR) filter.

The IIR low-pass filter consists of a feedback loop that contains one or more poles and zeros, which determine the frequency response and gain of the filter. The poles and zeros are located in the complex plane, and their locations determine the frequency response and stability of the filter.

The transfer function of the IIR low-pass filter can be expressed as a ratio of two polynomials in the complex variable "s", which represents the Laplace transform of the time-domain signal. The transfer function can be derived using techniques such as the bilinear transform, which maps the complex "s" plane to the unit circle in the complex "z" plane.

**Matlab code:**

fs=8000;

[n,w]=buttord(1200/4000,1500/4000,1,50); % finding the order of the filter

[b,a]=butter(n,w); % finding zeros and poles for filter

figure(1)

freqz(b,a,512,8000);

figure(2)

[h,q] = freqz(b,a,512,8000);

plot(q,abs(h)); % Normalized Magnitude plot

grid

figure(3)

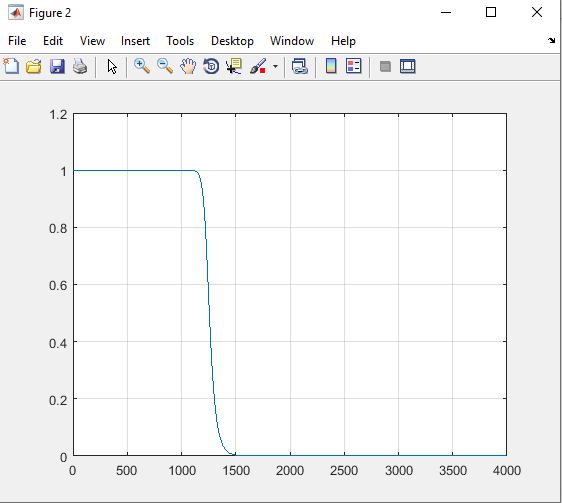
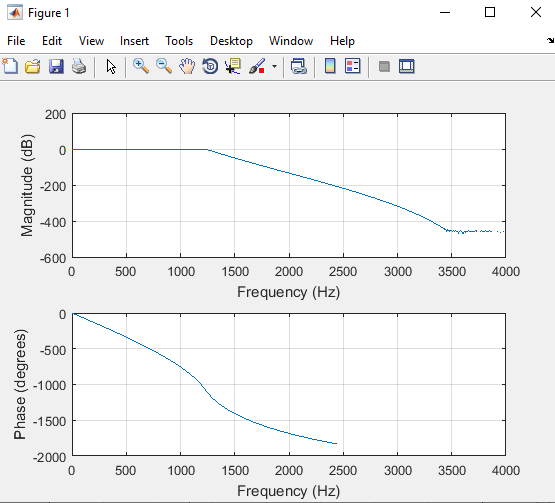
f=1200:2:1500;

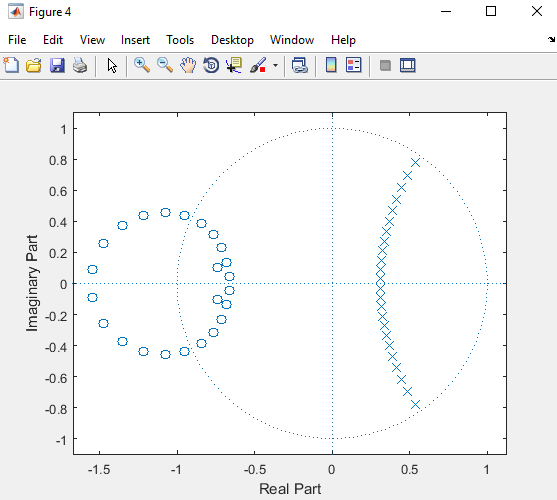
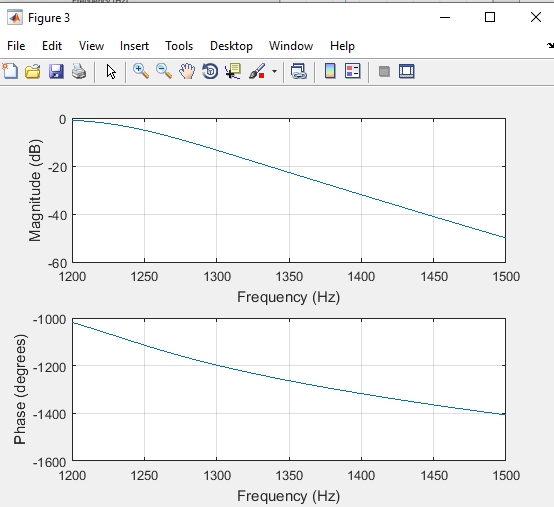
freqz(b,a,f,8000) % plotting the Transition band

figure(4)

zplane(b,a) % pole zero constellation diagram

**Output:**





**Experiment No: 07**

**Experiment Name: Write a program to designing a IIR High Pass Filter.**

**Objectives:**

1. Determine the filter specifications, such as the desired cutoff frequency and passband ripple.
2. Choose an appropriate filter type and design method based on the specifications.
3. Implement the filter design using a suitable software tool.
4. Analyze the filter performance using simulation and/or measurement techniques.
5. Evaluate the filter performance based on the design specifications and make any necessary adjustments.

**Theory:**

An IIR (Infinite Impulse Response) filter is a type of digital filter that uses feedback to achieve its desired frequency response. In contrast to FIR (Finite Impulse Response) filters, which only use feedforward paths, IIR filters can achieve a sharper cutoff and a more compact design.

The general transfer function of an IIR filter can be expressed as:

H(z) = Y(z) / X(z) = b0 + b1z^-1 + ... + bMz^-M / 1 + a1z^-1 + ... + aNz^-N

Where Y(z) is the output, X(z) is the input, M is the order of the numerator (b), and N is the order of the denominator (a).

To design an IIR high pass filter, we need to first determine the desired cutoff frequency and passband ripple. The cutoff frequency is the frequency below which the filter attenuates the input signal, and the passband ripple is the maximum deviation of the filter's gain from unity in the passband region.

**Matlab Code:**

[n,w]=buttord(1200/5000,1500/5000,1,50);

[b,a]=butter(n,w,'high');

figure(1)

freqz(b,a,512,10000);

figure(2)

[h,q] = freqz(b,a,512,8000);

plot(q,abs(h)); % Normalized Magnitude plot

grid

figure(3)

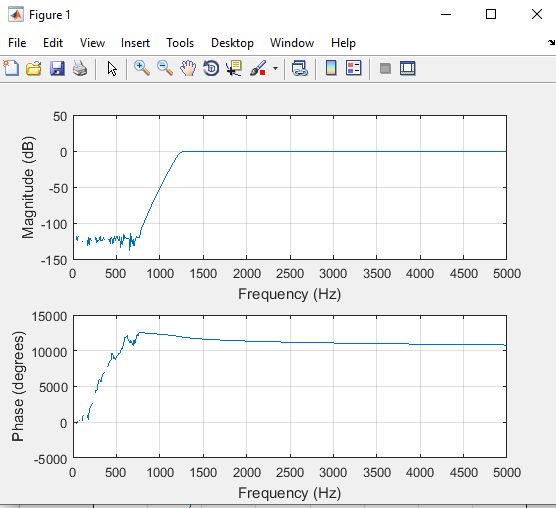
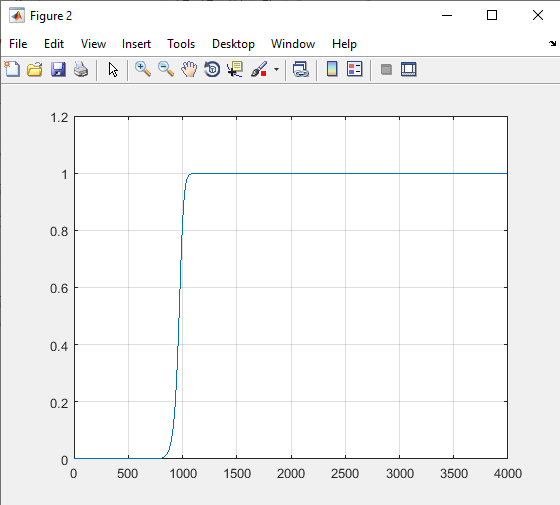
f=1200:2:1500;

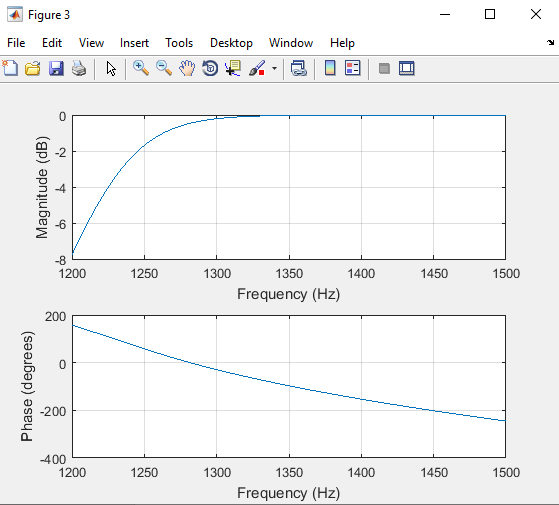
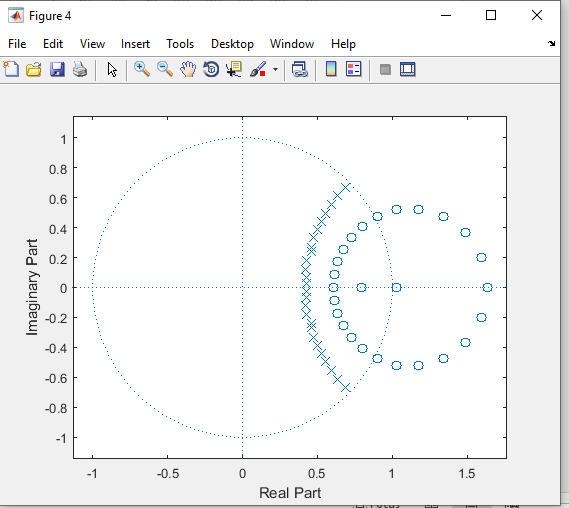
freqz(b,a,f,10000)

figure(4)

zplane(b,a)

**Output:**

**Experiment No: 08**

**Experiment Name: Write a program to designing a IIR Band Pass Filter.**

**Objectives:**

1. Determine the filter specifications, such as the center frequency, passband width, and stopband attenuation.
2. Choose an appropriate filter type and design method based on the specifications.
3. Implement the filter design using a suitable software tool.
4. Analyze the filter performance using simulation and/or measurement techniques.
5. Evaluate the filter performance based on the design specifications and make any necessary adjustments.

**Theory:**

An IIR (Infinite Impulse Response) bandpass filter is a type of digital filter that allows a specific frequency band to pass through while attenuating frequencies outside of that band. This is achieved by combining a high-pass filter and a low-pass filter.

The general transfer function of an IIR bandpass filter can be expressed as:

H(z) = Y(z) / X(z) = b0 + b1z^-1 + ... + bMz^-M / 1 + a1z^-1 + ... + aNz^-N

where Y(z) is the output, X(z) is the input, M is the order of the numerator (b), and N is the order of the denominator (a).

To design an IIR bandpass filter, we need to first determine the center frequency, passband width, and stopband attenuation. The center frequency is the frequency at which the passband is centered, and the passband width is the range of frequencies that are allowed to pass through. The stopband attenuation is the degree to which frequencies outside of the passband are attenuated.

**Matlab Code:**

[n,w]=buttord([1200/4000,2800/4000],[400/4000, 3200/4000],1,50);

[b,a]=butter(n,w,'bandpass');

figure(1)

freqz(b,a,128,8000)

figure(2)

[h,w]=freqz(b,a,128,8000);

plot(w,abs(h))

grid

figure(3)

f=600:2:1200;

freqz(b,a,f,8000); % Transition Band

figure(4)

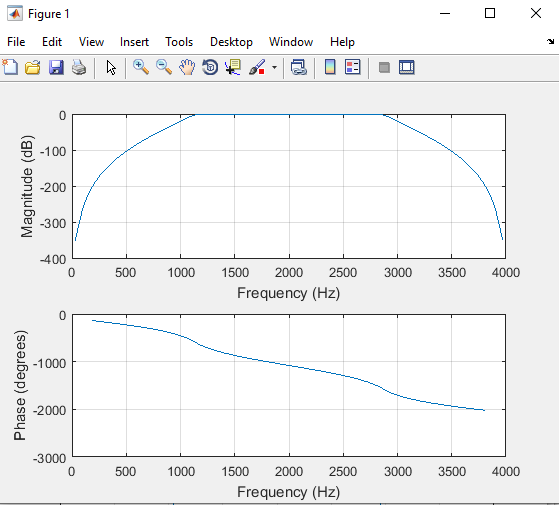
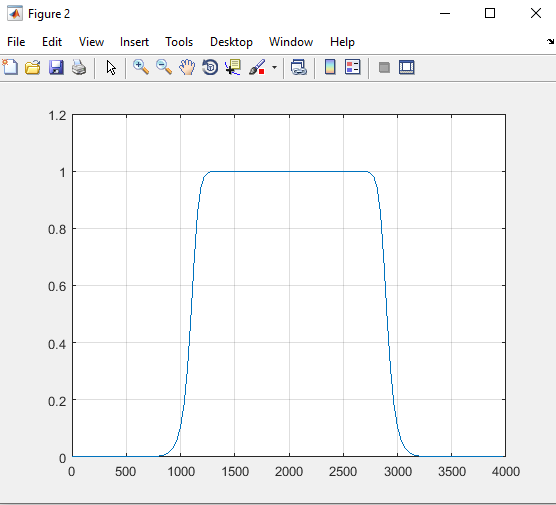
f=2800:2:3200;

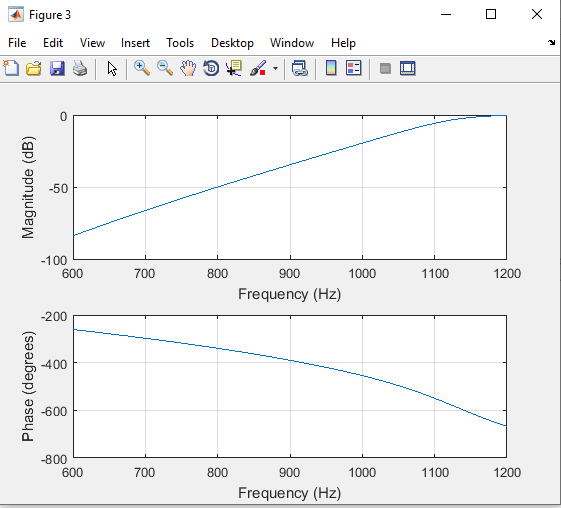
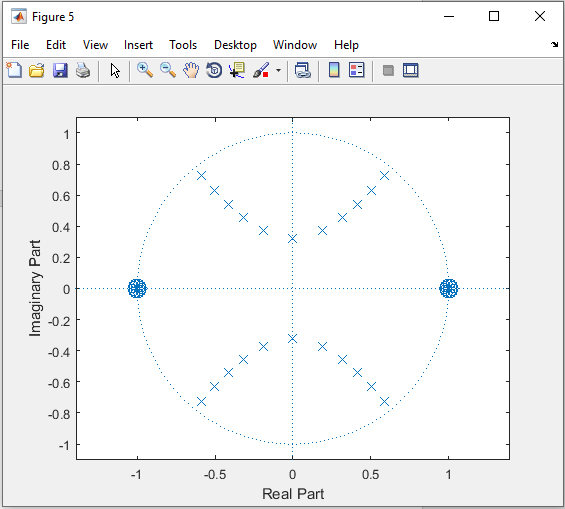
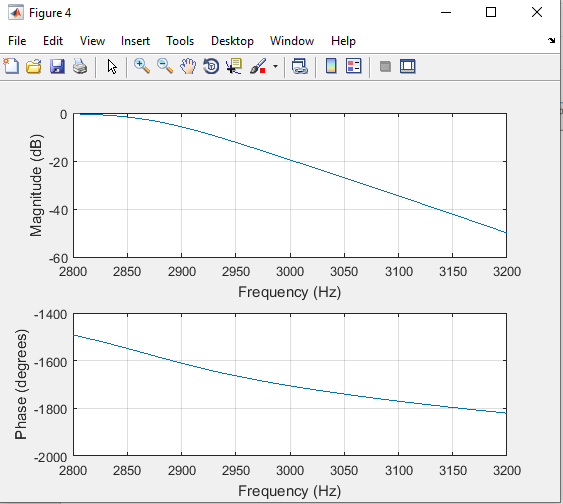
freqz(b,a,f,8000); % Transition Band

figure(5)

zplane(b,a)

**Output:**

**Experiment No: 09**

**Experiment Name: Write a program to designing a IIR Band stop Filter.**

**Objectives:**

1. Determine the filter specifications, such as the center frequency, passband width, and stopband attenuation.
2. Choose an appropriate filter type and design method based on the specifications.
3. Implement the filter design using a suitable software tool.
4. Analyze the filter performance using simulation and/or measurement techniques.
5. Evaluate the filter performance based on the design specifications and make any necessary adjustments.

**Theory:**

An IIR (Infinite Impulse Response) bandpass filter is a type of digital filter that allows a specific frequency band to pass through while attenuating frequencies outside of that band. This is achieved by combining a high-pass filter and a low-pass filter.

The general transfer function of an IIR bandpass filter can be expressed as:

H(z) = Y(z) / X(z) = b0 + b1z^-1 + ... + bMz^-M / 1 + a1z^-1 + ... + aNz^-N

where Y(z) is the output, X(z) is the input, M is the order of the numerator (b), and N is the order of the denominator (a).

To design an IIR bandpass filter, we need to first determine the center frequency, passband width, and stopband attenuation. The center frequency is the frequency at which the passband is centered, and the passband width is the range of frequencies that are allowed to pass through. The stopband attenuation is the degree to which frequencies outside of the passband are attenuated.

There are several types of IIR filters that can be used for bandpass filtering, including Butterworth, Chebyshev, and Elliptic filters. Each filter type has its own design method and trade-offs between passband width, stopband attenuation, and ripple.

Once the filter type and design method have been chosen, we can use software tools such as MATLAB or Python to implement the filter design. The software will typically provide functions or tools for designing the filter coefficients (b and a) based on the desired specifications.

**Matlab code:**

[n,w]=buttord([1200/4000,2800/4000],[400/4000, 3200/4000],1,50);

[b,a]=butter(n,w,'stop');

figure(1)

freqz(b,a,128,8000)

[h,w]=freqz(b,a,128,8000);

figure(2)

plot(w,abs(h));

grid

figure(3)

f=600:2:1200;

freqz(b,a,f,8000); % Transition Band

figure(4)

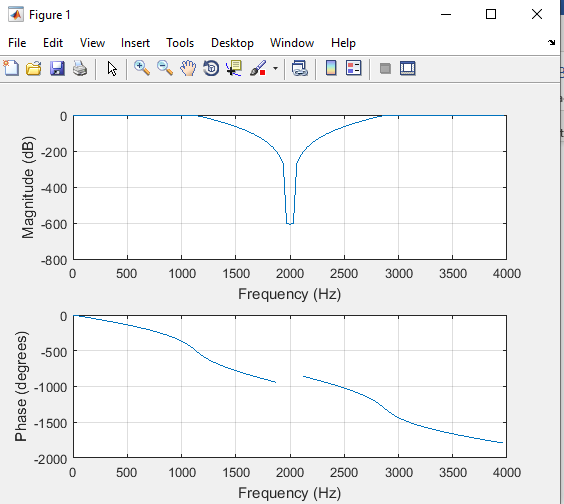
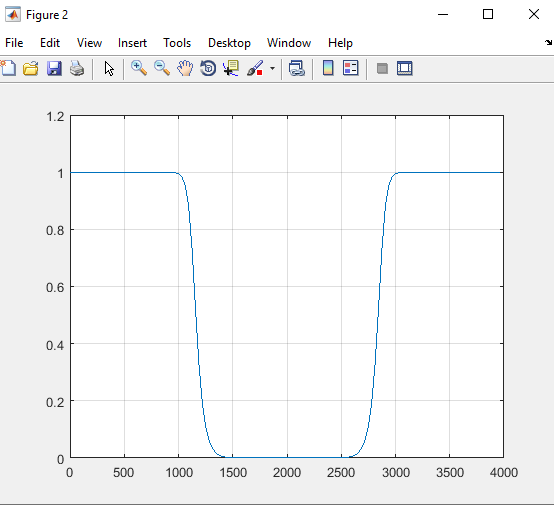
f=2800:2:3200;

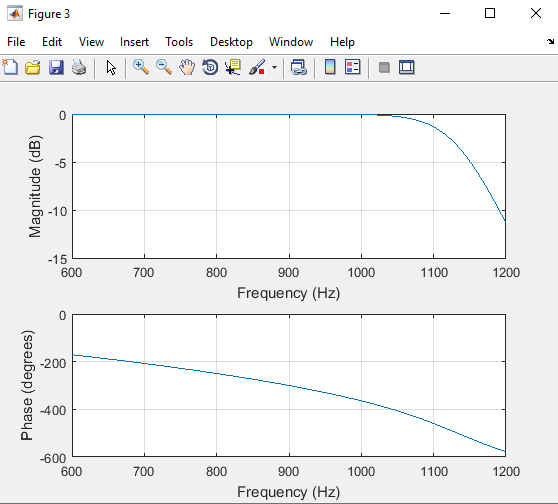
freqz(b,a,f,8000); % Transition Band

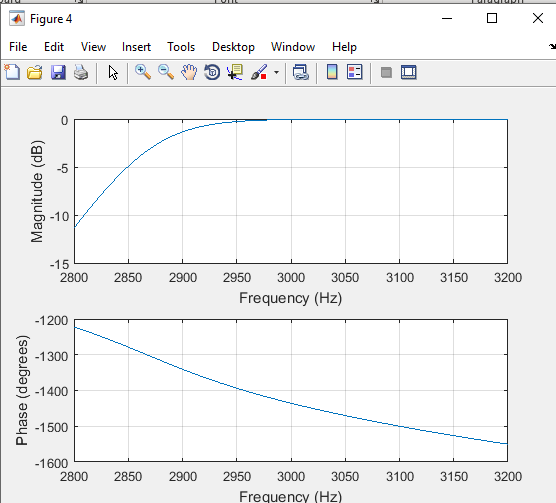
figure(5)

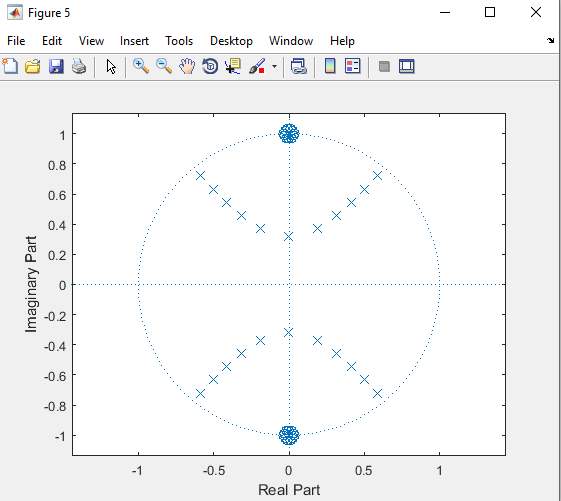
zplane(b,a);

**Output:**







**Experiment No: 10**

**Experiment Name: Design an IIR filter to suppress frequencies of 5 Hz and 30 Hz from given signal.**

**Objectives:**

1. Determine the filter specifications, such as the type of filter and order of the filter.
2. Design the filter using an appropriate method and software tool.
3. Implement the filter design and apply it to the given signal.
4. Evaluate the filter's performance in suppressing the specified frequencies.
5. Analyze the filtered signal and compare it with the original signal to assess the effectiveness of the filter.

**Theory:**

An IIR (Infinite Impulse Response) filter is a type of digital filter that can be designed to suppress certain frequencies in a signal. The filter can be designed using various methods, including Butterworth, Chebyshev, and Elliptic designs. Each method has its own trade-offs between passband ripple, stopband attenuation, and filter order.

To design an IIR filter to suppress frequencies of 5 Hz and 30 Hz from a given signal, we need to determine the filter specifications, including the type of filter and order of the filter. For this specific application, a bandstop filter should be used to suppress the specified frequencies. The order of the filter should be chosen to balance the required stopband attenuation and the complexity of the filter.

Once the filter specifications have been determined, we can design the filter using an appropriate method and software tool. For example, MATLAB or Python can be used to design the filter coefficients based on the specifications.

**Matlab Code:**

fs=100;

t=(1:100)/fs;

s=sin(2\*pi\*t\*5)+sin(2\*pi\*t\*15)+sin(2\*pi\*t\*30);

plot(t,s)

grid

wp1 = 10/50;

wp2 = 20/50;

ws1 = 5/50;

ws2 = 25/50;

wp = [Wp1 Wp2];

ws = [Ws1 Ws2];

rp = 0.1;

rs = 40;

[n,wn] = ellipord(wp,ws,rp,rs);

[b,a] = ellip(n,.1,40,w);

freqz(b,a,128,100)

[h,w]=freqz(b,a,128,100);

plot(w,abs(h));

grid

title(‘Normalized Magnitude Response’);

axis([0 50 0 1.2]);

figure(4)

sf=filter(b,a,s); % Time domain Response of the Filter

plot(t,sf)

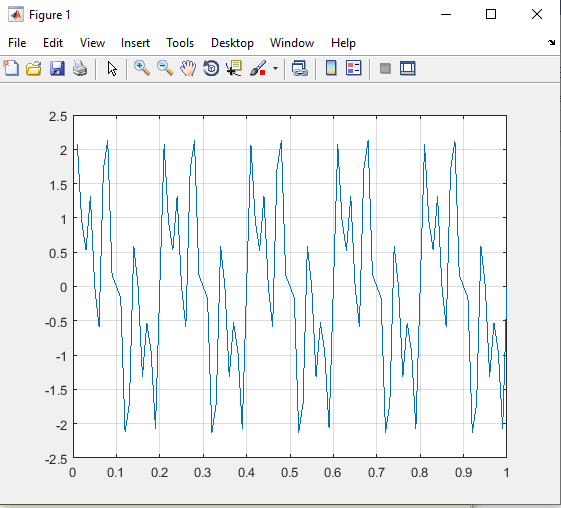
grid

xlabel('Time (seconds)');

ylabel('Signal Amplitude');

title('Filtered Signal only 15 Hz frequency');

**Output:**



Experiment no: 14

Name of The Experiment : Write a program to Compute the DTFT using the FFT algorithm.

Objectives:

The objective of this lab report is to explain the process of computing the Discrete Time Fourier Transform (DTFT) using the Fast Fourier Transform (FFT) algorithm. The lab report will cover the theoretical background of the FFT algorithm and provide a step-by-step guide on how to use the FFT algorithm to compute the DTFT of a given sequence. The lab report will also provide examples of how to apply the FFT algorithm to compute the DTFT of various types of signals.

Theory:

The Discrete Time Fourier Transform (DTFT) is a mathematical tool used to analyze the frequency content of a discrete-time signal. The DTFT of a sequence x[n] is defined as:

X(e^(jw)) = ∑\_(n=∞)^∞ x[n]e^(-jwn)

where w is the normalized frequency, and j is the imaginary unit. The DTFT of a sequence x[n] is a continuous function of w.

The Fast Fourier Transform (FFT) algorithm is an efficient way of computing the DTFT of a sequence. The FFT algorithm computes the DTFT of a sequence by dividing the sequence into smaller sub-sequences, computing their DTFTs, and then combining them to obtain the DTFT of the original sequence.

The FFT algorithm uses the Cooley-Tukey algorithm to compute the DTFT. The Cooley-Tukey algorithm is a divide-and-conquer algorithm that recursively divides the sequence into smaller sub-sequences and computes their DTFTs. The sub-sequences are combined using a butterfly structure, which is a series of complex multiplications and additions.

The FFT algorithm has a computational complexity of O(NlogN), where N is the length of the sequence. This makes it much faster than the direct computation of the DTFT, which has a computational complexity of O(N^2).

To compute the DTFT of a sequence using the FFT algorithm, we follow these steps:

Choose the sequence x[n] of length N.

Compute the FFT of x[n] using the FFT algorithm.

Scale the FFT result by 1/N to obtain the DTFT of x[n].

Python code:

import numpy as np

import matplotlib.pyplot as plt

r = np.array([1, 2, 3, 4, 5])

N = len(r)

R = np.fft.fft(r)

fs = 1 # Sampling frequency

f = np.arange(0, N) \* fs / N

plt.figure()

plt.subplot(2, 1, 1)

plt.plot(f, np.abs(R))

plt.xlabel('Frequency (Hz)')

plt.ylabel('Magnitude')

plt.title('Magnitude Spectrum')

plt.grid()

plt.subplot(2, 1, 2)

plt.plot(f, np.angle(R))

plt.xlabel('Frequency (Hz)')

plt.ylabel('Phase (rad)')

plt.title('Phase Spectrum')

plt.grid()

plt.show()

Output:

