Experiment no: 01

Name of the Experiment: Sampling of a sinusoidal signal and reconstruction of analog signal.

Objectives:

1. To convert a continuous-time signal into a discrete-time signal.
2. reconstructing an analog signal from a sampled signal is to obtain an analog signal that closely approximates the original analog signal.

Theory:

Sampling of a sinusoidal signal and reconstruction of an analog signal are fundamental concepts in signal processing. The process involves converting a continuous-time analog signal into a discrete-time digital signal, and then reconstructing the analog signal from the digital signal. The following is a brief overview of the theory behind these processes:

Sampling:

Sampling is the process of converting a continuous-time analog signal into a discrete-time digital signal. This is done by taking regular samples of the analog signal at fixed intervals of time. The rate at which the samples are taken is called the sampling rate, and is typically measured in samples per second or Hertz (Hz).

The Nyquist-Shannon Sampling Theorem states that in order to accurately represent a continuous-time signal using a digital signal, the sampling rate must be at least twice the highest frequency present in the analog signal. This is known as the Nyquist rate.

Reconstruction:

Reconstruction is the process of converting a discrete-time digital signal back into a continuous-time analog signal. This is done by using an analog reconstruction filter to smooth out the stair-step waveform that results from the digital samples. The analog filter removes high-frequency components above the Nyquist frequency, and interpolates between the discrete-time samples to reconstruct the original analog signal.

One common reconstruction method is the zero-order hold (ZOH) method, which holds the value of each sample until the next sample is taken. Another common method is the first-order hold (FOH) method, which linearly interpolates between adjacent samples.

Applications:

Sampling and reconstruction are used in many applications, including digital audio, digital image processing, and digital communication systems. In digital audio, sampling and reconstruction are used to convert analog audio signals into digital audio signals, which can be stored and manipulated using digital signal processing techniques. In digital image processing, sampling and reconstruction are used to convert analog images into digital images, which can be manipulated using image processing algorithms. In digital communication systems, sampling and reconstruction are used to transmit and receive digital signals over communication channels.

Source Code:

```
% Define the parameters of the signal
f = 10;             % Frequency of the sinusoid (in Hz)
fs = 200;           % Sampling rate (in Hz)
t = 0:1/fs:1;       % Time vector

% Generate the sinusoidal signal
x = sin(2*pi*f*t);

% Plot the original signal
%figure(1)
subplot(311);
plot(t,x);
xlabel('Time (s)');
ylabel('Amplitude');
title('Original Signal');

% Sample the signal
Ts = 1/fs;          % Sampling interval (in seconds)
n = 0:Ts:1;         % Sampling instants
xn = sin(2*pi*f*n);  % Sampled signal

% Plot the sampled signal
%figure(2)
subplot(312);
stem(n,xn);
xlabel('Time (s)');
ylabel('Amplitude');
```

```
title('Sampled Signal');

% Reconstruct the analog signal using ideal reconstruction
xr = zeros(size(t));  % Initialize the reconstructed signal
for i = 1:length(n)
    xr = xr + xn(i)*sinc((t-(i-1)*Ts)/Ts);
end

% Plot the reconstructed signal
%figure(3)
subplot(313);
plot(t,xr);
xlabel('Time (s)');
ylabel('Amplitude');
title('Reconstructed Signal');
```

Experiment no:02

Name of the experiment: Z-transform of a discrete time function, Inverse Z-transform, pole-zeros diagram and root of a system .

Objectives:

1. To convert a discrete-time signal into a representation in the Z-domain.
2. To convert a function in the Z-domain back into the time-domain representation.
3. to provide a graphical representation of the poles and zeros of a system in the Z-domain.

Theory:

The Z-transform of a discrete time function is a mathematical technique used to transform a sequence of discrete data into a function of a complex variable z. The inverse Z-transform is then used to transform the function back into the original sequence.

To find the Z-transform of a discrete time function, you can use the formula:

$X(z) = \Sigma x[n]z^{(-n)}$

where x[n] is the sequence of discrete data and z is the complex variable.

To find the inverse Z-transform, you can use the formula:

$x[n] = (1/2\pi j)\Sigma X(z)z^{(n-1)}dz$

where X(z) is the function of the complex variable z.

A pole-zero diagram is a graphical representation of the poles and zeros of a system. Poles are the values of z that cause the transfer function to go to infinity, while zeros are the values of z that cause the transfer function to be zero. The pole-zero diagram can help in analyzing the stability and behavior of a system.

The root of a system refers to the values of z that make the transfer function of the system equal to zero. These values can be found by setting the transfer function equal to zero and solving for z. The roots can also be represented on the pole-zero diagram.

In the context of control systems theory, the pole-zero diagram and root analysis are important tools for understanding the behavior of a system and designing control algorithms to ensure the system is stable and performs as desired.

Source Code:

```
syms z n

a=1/16^n;

ZTrans=ztrans(a);       %Z transform

disp(ZTrans);
```

```
InvrZ=iztrans(a);       %InverseZtransform
disp(InvrZ);
B=[0 1 1];
A=[1 -2 3];
roots(A);
roots(B);
zplane(B,A);
syms Z n
I=iztrans(3*Z/(Z+1));
disp(I);
clc;clear;
z=roots([1,0,2]);
p=roots([1,2,-1,1]);
clc;clear;
z=roots([1,0,0,1]);
p=roots([1,0,2,0,1]);
clc;clear;
ZZ=roots([4,8,10]);
PP=roots([2,8,18,20]);
clc;clear;
num=[1,0,0,1];
den=[1,0,2,0,1];
systf=tf(num,den);
pzmap(systf);
```

Experiment no:03

Name of the experiment: The Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT).

Objectives:

1. To analyze the frequency content of a signal.
2. To implement digital filters.
3. To compress and encode data.

Theory:

The Discrete Fourier Transform (DFT) and Fast Fourier Transform (FFT) are mathematical algorithms used to analyze signals and data in the frequency domain. The DFT and FFT can be used to determine the frequency content of a signal and are commonly used in digital signal processing applications.

The DFT is a mathematical operation that transforms a sequence of time-domain data into a sequence of frequency-domain data. The DFT can be computed using the following equation:

$X[k] = \Sigma x[n]e^{(-2\pi ikn/N)}$

where X[k] is the frequency-domain representation of the signal, x[n] is the time-domain sequence, and N is the length of the sequence.

The FFT is a more efficient algorithm for computing the DFT. The FFT reduces the number of computations required to compute the DFT by exploiting the symmetries and periodicity of the

complex exponential functions used in the DFT. The FFT is commonly used in digital signal processing applications due to its speed and computational efficiency.

In practice, the DFT and FFT are used to analyze signals and data in the frequency domain. The frequency-domain representation of a signal can reveal information about the frequency content of the signal, such as the presence of specific frequency components or the frequency spectrum of noise in the signal. The DFT and FFT are commonly used in a wide range of applications, including audio processing, image processing, and data compression.

In summary, the DFT and FFT are mathematical algorithms used to analyze signals and data in the frequency domain. The DFT transforms a sequence of time-domain data into a sequence of frequency-domain data, while the FFT is a more efficient algorithm for computing the DFT. The DFT and FFT are commonly used in digital signal processing applications to analyze signals and extract information about their frequency content.

Source Code:

```
n=-1:3;
x=1:5;
k=0:500;
w=(pi/500)*k;
X=x*(exp(-j*pi/500))*(n'*k);
magX=abs(X);
angX=angle(X);
realX=real(X);
imagX=imag(X);
subplot(221);
plot(k/500,magX);
grid;
xlabel('Frequency in pi units ');
title('Magnitude part');
subplot(222);
plot(k/500,angX/pi);
grid;
```

```
xlabel('Frequency in pi units ');
title('Angle part');
subplot(223);
plot(k/500,realX);
grid;
xlabel('Frequency in pi units ');
title('Real part');
subplot(224);
plot(k/500,imagX);
grid;
xlabel('Frequency in pi units ');
title('Imaginary part');
                    %%%%%%          FFT PART          %%%%%%%%%%%%%%
N=256;
T=1/128;
k=0:N-1;
time=k*T;
f=0.25+2*sin(2*pi*5*k*T)+1*sin(2*pi*12.5*k*T)+1.5*sin(2*pi*20*k*T)+0.5*sin(2*pi*35*k*
T);
subplot(211);
plot(time,f);
title('Signal sampled at 128Hz');
F=fft(f);
magF=abs([F(1)/N,F(2:N/2)/(N/2)]);
hertz=k(1:N/2)*(1/(N*T));
subplot(212);
stem(hertz,magF);
title('Frequency Components');
```

Experiment no: 04

Name of the experiment:

Objectives:

1. to provide a stable and linear-phase filter response that can be easily designed and implemented.
2. to provide a more efficient and flexible filter response that can be designed to have specific frequency response characteristics.
3. to filter out unwanted noise or frequencies from a signal, while preserving the signal's essential characteristics.
4. To design a filter response with specific frequency characteristics

Theory:

FIR (Finite Impulse Response) and IIR (Infinite Impulse Response) filters are two types of digital filters used in signal processing applications. Both types of filters have different characteristics and are used for different purposes.

FIR filters are digital filters with a finite impulse response. They are characterized by their linear phase response, which means that all frequencies in the input signal are delayed by the same amount of time. FIR filters have a stable response and are used for applications where phase distortion needs to be avoided, such as in audio processing.

The impulse response of an FIR filter can be expressed as a finite sequence of coefficients, which are used to compute the output of the filter. The coefficients can be adjusted to change the frequency response of the filter, making it possible to design FIR filters with specific frequency characteristics.

IIR filters, on the other hand, have an infinite impulse response. They are characterized by their non-linear phase response, which means that different frequencies in the input signal are delayed by different amounts of time. IIR filters have a more complex response than FIR filters and can introduce phase distortion, but they have a more efficient design and require fewer coefficients to achieve a specific frequency response.

The impulse response of an IIR filter can be expressed as a recursive equation that depends on both the current input and previous outputs. IIR filters can be designed using different methods, such as Butterworth, Chebyshev, and elliptic filter designs.

In summary, FIR and IIR filters are two types of digital filters used in signal processing applications. FIR filters have a finite impulse response, linear phase response, and a stable response, while IIR filters have an infinite impulse response, non-linear phase response, and a more efficient design. FIR filters are used for applications where phase distortion needs to be avoided, while IIR filters are used for applications where efficiency is more important than phase distortion.

Source Code:

```
% FIR Now our target is to pass all frequencies above 1200 Hz

fs=8000;

n=50;

w=1200/ (fs/2); b=fir1(n,w,'high');

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);


%Suppose out target is to pass all frequencies below 1200 Hz
```

```
fs=8000; % sampling frequency
n=50; % order of the filter
w=1200/ (fs/2);
b=fir1(n,w,'low'); % Zeros of the filter
freqz(b,1,128,8000); % Magnitude and Phase Plot of the filter
figure(2)
[h,w]=freqz(b,1,128,8000);
plot(w,abs(h)); % Normalized Magnitude Plot
grid
figure(3)
zplane(b,1);

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);

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);

%We will consider same filter but our target now is to pass all frequencies above 1200 Hz
[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)
%Suppose our target is to design a filter to pass all frequencies below 1200 Hz with pass band
%ripples = 1 dB and minimum stop band attenuation of 50 dB at 1500 Hz. The sampling frequency
%for the filter is 8000 Hz;
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

%with pass band ripples = 1 dB and minimum stop band attenuation of 50 dB. The
%sampling frequency for the filter is 8000 Hz;
[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)
```

```
%%bandstop pass
[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);
```