EXPERIMENT 1

AIM: TO STUDY DECIMATION AND INTERPOLATION OF SIGNAL

SOFTWARE: MATLAB

THOERY:

**Decimation** reduces the sampling rate of a signal by a factor of DDD. It involves low-pass filtering to remove high-frequency components and downsampling, where every Dth sample is retained. This is used for data compression and efficient processing.

**Interpolation** increases the sampling rate by a factor of LLL. It consists of upsampling, where L−1L-1L−1 zeros are inserted between samples, followed by low-pass filtering to smooth the signal. It is used in signal reconstruction and sample rate conversion.

Both processes help in optimizing signal processing for communication and multimedia applications.

CODES:

DECIMATION:

clc;

clear all;

close all;

D=input("Enter the Downsapmling Factor");

L=input("Enter the Length of the input signal");

f1=input("Enter the frequency of first sinusoidal signal");

f2=input("Enter the frequency of second sinusoidal signal");

n=0:L-1;

x=sin(2\*pi\*f1\*n)+sin(2\*pi\*f2\*n);

y=decimate(x,D,'fir');

subplot(2,1,1);

stem(n,x(1:L));

title('input sequence');

xlabel('time(n)');

ylabel('amplitude');

subplot(2,1,2);

m=0:(L/D)-1;

stem(m,y(1:L/D));

title('Decimated sequence');

xlabel('time(n)');

ylabel('amplitude');

INTERPOLATION:

D = input('Enter the Downsampling Factor: ');

L = input('Enter the Length of the input signal: ');

f1 = input('Enter the frequency of the first sinusoidal signal: ');

f2 = input('Enter the frequency of the second sinusoidal signal: ');

n = 0:L-1;

% Create the input signal as the sum of two sinusoidal signals

x = sin(2\*pi\*f1\*n) + sin(2\*pi\*f2\*n);

% Decimate (downsample) the signal

y = decimate(x, D, 'fir');

% Upsample (interpolate) the decimated signal by a factor of I

I = input('Enter the Interpolation Factor: '); % Interpolation factor

y\_interpolated = upsample(y, I);

% Plot the original signal

subplot(3, 1, 1);

stem(n, x(1:L)); % Display original signal

title('Input Sequence');

xlabel('Time (n)');

ylabel('Amplitude');

% Plot the decimated signal

subplot(3, 1, 2);

m = 0:(L/D)-1;

stem(m, y(1:L/D)); % Display decimated signal

title('Decimated Sequence');

xlabel('Time (n)');

ylabel('Amplitude');

% Plot the interpolated signal

subplot(3, 1, 3);

m\_interpolated = 0:(L/D)\*I-1; % Adjust the time vector for interpolated signal

stem(m\_interpolated, y\_interpolated(1:(L/D)\*I)); % Display interpolated signal

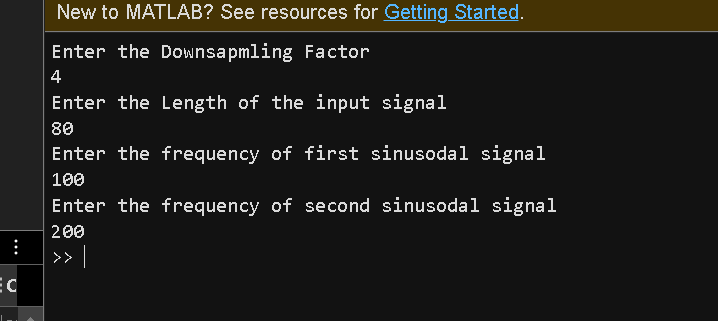
title('Interpolated Sequence');

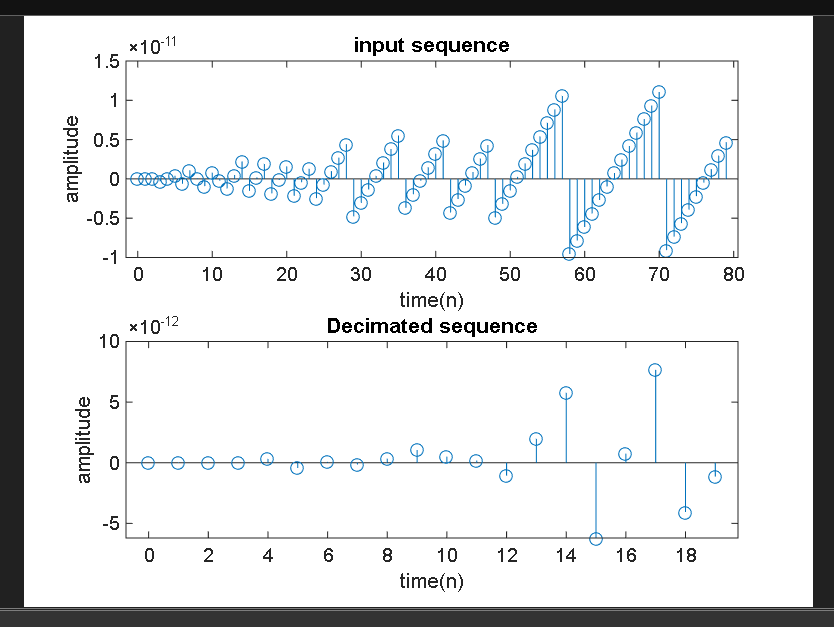
xlabel('Time (n)');

ylabel('Amplitude');

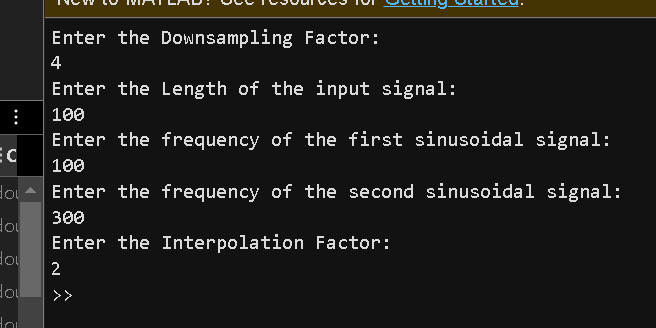
OUTPUT:

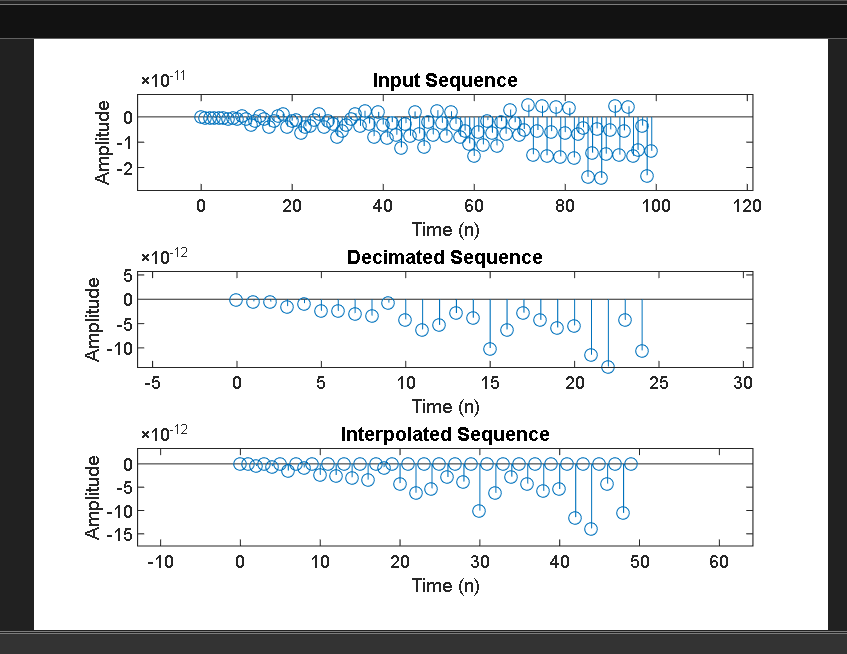
DECIMATION:





INTERPOLATION:





CONCLUTION:

Decimation and interpolation are fundamental techniques for modifying the sampling rate of a signal. Decimation reduces redundancy and optimizes processing, while interpolation reconstructs signals for higher resolution. Proper filtering is essential in both processes to prevent aliasing and distortion, ensuring efficient and accurate signal manipulation in digital communication and multimedia applications.