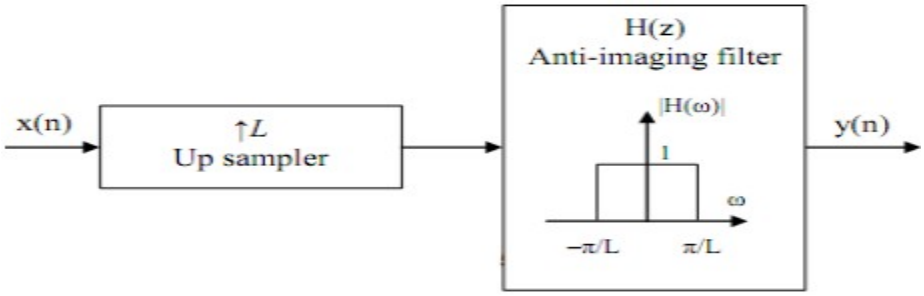
Experiment No: 9
Experiment Name: : Implementation Of Interpolation Process

Objective: The objective of this lab is to understand and implement interpolation techniques to estimate values between known data points. This experiment focuses on applying various interpolation methods, such as linear and polynomial interpolation, to derive approximate values from a set of discrete data. The goal is to evaluate the accuracy, efficiency, and practical applications of these methods in solving real-world problems where data points are sparse or incomplete. By the end of the experiment, students will gain hands-on experience in implementing and comparing different interpolation algorithms.

Software:

- MATLAB

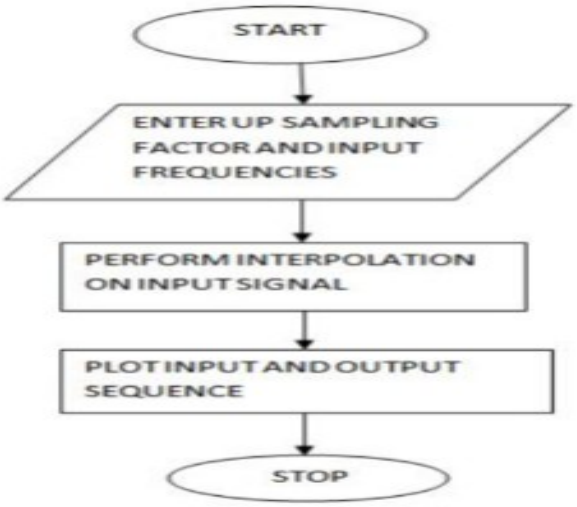
Theory: "Upsampling" refers to the process of inserting zero-valued samples between the original samples to increase the sampling rate, often called "zero-stuffing." "Interpolation" involves upsampling followed by filtering, where the filtering step removes the undesired spectral images introduced by zero-stuffing. The main purpose of interpolation is to increase the sampling rate of a signal so that it can be processed by a system operating at a higher sampling rate, ensuring compatibility between different systems.



Algorithm:

1. Define the upsampling factor and input frequencies f1f_1 and f2f_2.
2. Represent the input signal with frequencies f1f_1 and f2f_2.
3. Apply interpolation on the input signal using the MATLAB command interp.
4. Plot both the input signal and the interpolated output signal/sequence.

Flow Chart:



Code:

```
clc;
clear all;
close all;
L = input('Enter the upsampling factor: ');
N = input('Enter the length of the input signal: '); % Length should be greater than 8
f1 = input('Enter the frequency of the first sinusoidal: ');
f2 = input('Enter the frequency of the second sinusoidal: ');
n = 0:N-1;
x = sin(2 * pi * f1 * n) + sin(2 * pi * f2 * n);
y = interp(x, L);
subplot(2, 1, 1);
stem(n, x(1:N));
title('Input Sequence');
xlabel('Time (n)');
ylabel('Amplitude');
subplot(2, 1, 2);
m = 0:N*L-1;
stem(m, y(1:N*L));
title('Output Sequence');
xlabel('Time (n)');
ylabel('Amplitude');
```

Input:

Enter the upsampling factor: 10
Enter the length of the input signal: 220
Enter the frequency of the first sinusoidal: 1
Enter the frequency of the second sinusoidal: 1.2

Output:

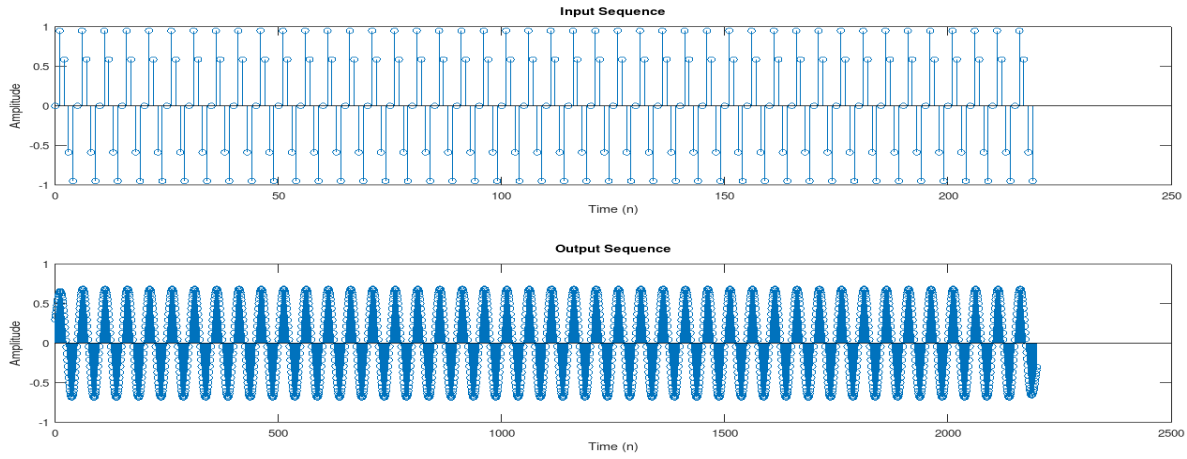


Figure: Output

Discussion: The implementation of the interpolation process involves increasing the sampling rate of a signal by inserting additional samples between existing ones. In this lab, MATLAB demonstrated interpolation, which includes upsampling followed by low-pass filtering to prevent aliasing. This process was applied to a discrete signal, showing how interpolation smooths the signal and reconstructs it at a higher resolution. Visual comparisons between the original and interpolated signals highlighted the improved sampling density. This experiment reinforced key concepts of signal reconstruction, filtering, and their importance in digital signal processing applications.