Experiment No: 8
Experiment Name: : Implementation Of Decimation Process

Objective: The objective of this lab report is to implement and study the decimation process, which reduces a signal's sampling rate through selective downsampling. The lab focuses on understanding decimation principles, its impact on signal bandwidth and resolution, and its role in enhancing efficiency in digital signal processing systems. Practical implementation will demonstrate how decimation reduces computational load while addressing trade-offs like aliasing, exploring methods to minimize its effects.

Software:

- MATLAB

Theory: Decimation is the process of reducing a signal's sampling rate, while downsampling specifically refers to discarding samples without applying a low-pass filter. Decimation is often performed to match the output sampling rate of one system with the input requirements of another operating at a lower rate. More commonly, it is used to minimize processing costs, as the computational and memory demands of a DSP system are typically proportional to the sampling rate. By lowering the sampling rate, decimation enables more cost-effective implementations.

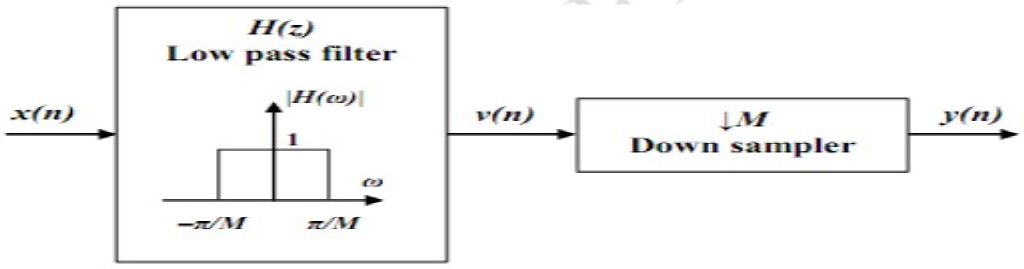
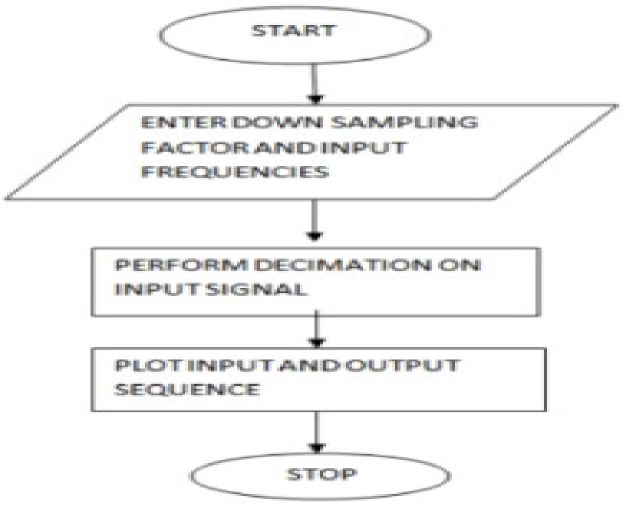


Figure 1:

Algorithm:

1. Define the downsampling factor and input frequencies f1f_1 and f2f_2.
2. Represent the input signal with frequencies f1f_1 and f2f_2.
3. Apply decimation on the input signal using the MATLAB command decimate.
4. Plot the input signal and the decimated output signal.

Flow Chart:



START
ENTER DOWN SAMPLING FACTOR AND INPUT FREQUENCIES
PERFORM DECIMATION ON INPUT SIGNAL
PLOT INPUT AND OUTPUT SEQUENCE
STOP

Code:

```
clc; clear all; close all;
D = input('Enter the downsampling factor: ');
L = input('Enter the length of the input signal: ');
f1 = input('Enter the frequency of the first sinusoidal: ');
f2 = input('Enter the frequency of the second sinusoidal: ');
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');
```

Input:

```
Enter the downsampling factor: 10
Enter the length of the input signal: 3000
Enter the frequency of the first sinusoidal: 0.1
Enter the frequency of the second sinusoidal: 0.07
```

Output:

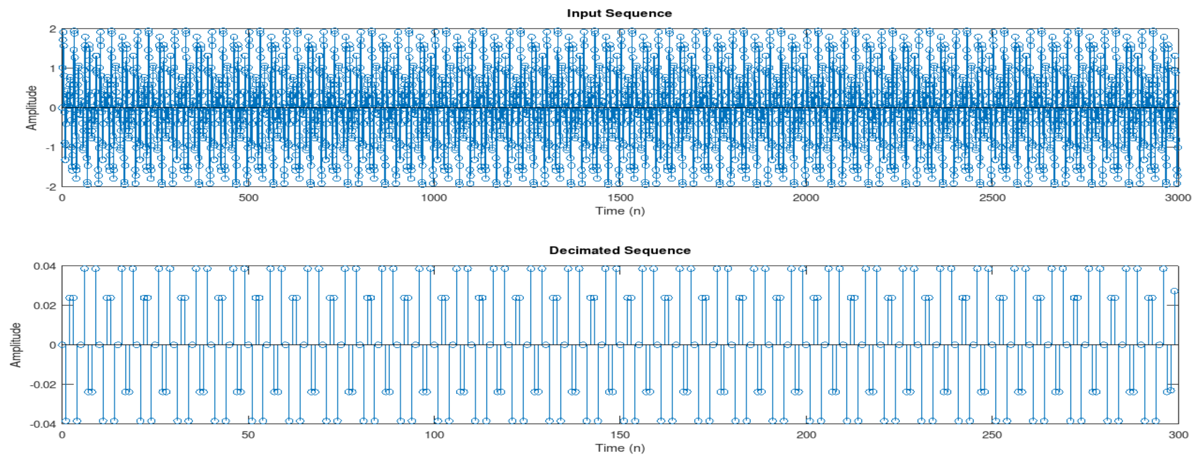


Figure: Output

Discussion: The lab focused on implementing the decimation process, where the signal's sampling rate was reduced by a factor of DD. Using MATLAB, an input signal with two sinusoidal components was generated and processed with the decimate function, which applies low-pass filtering to prevent aliasing. Both the original and decimated signals were visualized, showing the impact of downsampling. This exercise reinforced key signal processing concepts, such as the Nyquist criterion and the necessity of pre-filtering to avoid aliasing in the decimation process.