EE2801 - DSP LAB

Experiment – 8: Interpolation and Decimation

Name: Anupama Kulshreshtha

Roll no.: EE22BTECH11009

Aim of the Experiment:

To perform Interpolation and Decimation operations on a given input signal, and then calculate and display the mean of absolute error between the input signal and the obtained output signal.

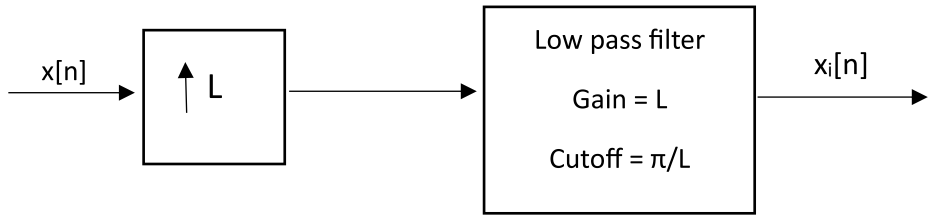
Theory of Interpolator and Decimator:

Interpolation and decimation are essential processes in digital signal processing (DSP), often used in applications such as digital communications, audio processing, and image processing.

Interpolator:

Interpolation is the process of increasing the sampling rate of a discrete-time signal. It involves upsampling the signal followed by applying a low pass filter. The purpose of interpolation is to reconstruct a smoother version of the original signal at a higher sampling rate, allowing for finer resolution in time or frequency domain analysis.

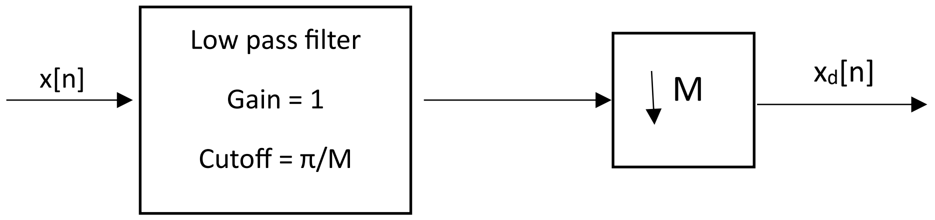
The block diagram for an interpolator is given by:



Decimator:

Decimation is the process of reducing the sampling rate of a discrete-time signal. It involves applying a low pass filter to the input signal and then downsampling it. It is often used in applications where high sampling rates are unnecessary or impractical, such as reducing computational complexity or conserving memory.

The block diagram for a decimator is given by:



How filter cutoff frequencies are decided in Interpolation and Decimation:

Decimation involves downsampling of the obtained signal. Therefore, if the appropriate cutoff frequency conditions are not taken care of, it results in aliasing, which makes it difficult to reconstruct the original signal.

Hence, for decimation, input signal bandwidth must be **less than 2π/M**, to ensure no aliasing is occurring.

In the given experiment, the input signal of decimator is the output signal of the interpolator. Hence, the cutoff frequencies of the low pass filters used in interpolator and decimator are chosen to be around π/2, which is less than π, so that no aliasing takes place.

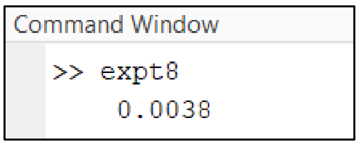
Matlab Code and Mean of absolute error:

```
L = 2;
M = 2;

f1 = 500;
f2 = 1000;
f3 = 700;
fs = 5000;
t = 0:1/fs:1;
x = sin(2*pi*f1*t) + 2*sin(2*pi*f2*t) + 1.5*sin(2*pi*f3*t);

xi = myInterpolator(x, L);
x_hat = myDecimator(xi, M);

error = mean(abs(x-x_hat));
disp(error)

function xi = myInterpolator(x, L)
    wc = pi/L;
    N = 21;
    fs_filter = 5000;
    fc = (wc * fs_filter) / (2 * pi);

    x1 = upsample(x,L);
    h1 = LPF(fc,fs_filter,N, L);
    xi = myConvolution(x1, h1);
    xi = xi(11:end);
    xi = xi(1:end-10);
end

function xd = myDecimator(x, M)
    wc = pi/M;
    N = 21;
    fs_filter = 5000;
    fc = (wc * fs_filter) / (2 * pi);
    h2 = LPF(fc, fs_filter, N, 1);
    x2 = myConvolution(x, h2);
    x2 = x2(11:end);
    x2 = x2(1:end-10);
    xd = downsample(x2,M);
end

function result = downsample(x, m)
    % Lengths of the signal
    L = length(x);
```

```matlab
    k = round(L/m);
    % Initialize the result signal
    result = zeros(1, k);
    result(1) = x(1);

    for i = 2:k
        if m*(i-1)+1 <= L
        result(i) = x(m*(i-1)+1);
        end
    end
end

function result = upsample(x,m)
    % Lengths of the signal
    L = length(x);
    k = m*L;
    % Initialize the result signal
    result = zeros(1, k);
    result(1) = x(1);

    for i = 2:k
        if mod(i-1,m) == 0
            result(i) = x(((i-1)/m)+1);
        else
            result(i) = 0;
        end
    end
end

function h = LPF(fc, fs, N, gain)
    wc = 2 * pi * fc / fs;
    hd = zeros(1,N);
    % Calculating impulse response
    for k = 1:N
        n = k - (N+1)/2;
        if n == 0
            hd(k) = wc/pi;
        else
            hd(k) = sin(wc*n)/(pi*n);
        end
    end
    % Define Hamming window
    n1 = 0:N-1;
    WH = zeros(1, N);
    l = (n1 >= 0) & (n1 <= N-1);
    WH(l) = 0.54 - 0.46 * cos(2 * pi * n1(l) / (N-1));

    % Apply window to filter coefficients to get practical impulse response
    h = hd .* WH;
    h = gain * h;
end

function result = myConvolution(x,h)
    % Lengths of the signals
    M = length(x);
    N = length(h);

    % Length of the result signal
    L = M + N - 1;

    % Initialize the result signal
    result = zeros(1, L);
    % Perform convolution
    for n = 1:L
        for k = max(1, n-N+1):min(n, M)
```

```
            result(n) = result(n) + x(k) * h(n-k+1);
        end
    end
end
```

The mean of absolute error value obtained on execution of this code is given by:



```
>> expt8
     0.0038
```

Observations and Conclusions:

- The error calculated using the mean absolute difference between the input and output signals was around 0.0038, which indicates the discrepancy between the input signal and the output signal after passing through the interpolator and decimator.
- The discrepancy between the input and output signals can be attributed to various sources of error, including numerical precision limitations, noise among several others.
- Techniques such as optimization of interpolation and decimation filters, using higher precision arithmetic, and employing advanced signal processing algorithms may help minimize errors in future implementations.
- Despite the observed error, the experiment demonstrates the practical application of interpolation and decimation in altering the sampling rate of signals.
- Understanding the sources of error and optimizing the implementation can lead to improved performance and accuracy in signal processing applications.