DSP LAB - Experiment 8

Interpolation and Decimation

Ajay Krishnan K
EE22BTECH11003

April 11, 2024

Aim

To implement interpolation and decimation of a given signal and calculate the mean of absolute error between the original signal and the decimated signal.

Theory

Interpolation

Interpolation is the process of increasing the sampling rate of a signal. This is done by inserting zeros between the samples of the signal and then applying a low-pass filter to remove the high frequency components introduced by the zero padding.

Decimation

Decimation is the process of decreasing the sampling rate of a signal. This is done by initially passing the signal through a low-pass filter to remove the high frequency components and then removing samples from the signal.

Procedure

1. Generate a signal $x[n] = \sin(2\pi f_1 n) + \sin(2\pi f_2 n) + \sin(2\pi f_3 n)$ where $f_1 = 500$, $f_2 = 1000$ and $f_3 = 700$.
2. Interpolate the signal by a factor of $L = 2$.
3. Decimate the interpolated signal by a factor of $M = 2$.
4. Observe the effect of interpolation and decimation on the signal.

Design

Interpolation

The interpolation of a signal is done by inserting zeros between the samples of the signal. The interpolated signal is then passed through a low-pass filter to remove the high frequency components introduced by the zero padding.

The cutoff frequency of the low-pass filter is given by

$$w_c = \frac{\pi}{L}$$

where f_s is the sampling frequency of the signal and L is the interpolation factor.

Since the signal is being interpolated by a factor of L, the sampling frequency of the signal is also increased by a factor of L. Thus, the equation.

A gain of L is applied to the output of the low-pass filter to compensate for the loss of energy due to the zero padding.

Decimation

The decimation of a signal is done by removing samples from the signal. The decimated signal is then passed through a low-pass filter to remove the high frequency components introduced by the decimation.

The cutoff frequency of the low-pass filter is given by

$$w_c = \frac{\pi}{M}$$

where f_s is the sampling frequency of the signal and M is the decimation factor.

Since the signal is being decimated by a factor of M, the sampling frequency of the signal is also decreased by a factor of M. Thus, the equation.

Matlab Code

Code for the LPF is given below.

```
function [h_n] = LPF(fc,fs,N,A)
    n = (-N + 1)/2 : (N - 1)/2;
    omega_c = 2 * pi * fc / fs;
    hd_n = sin(omega_c * n) ./ (pi * n);
    hd_n(N - (N - 1)/2) = omega_c / pi;
    w_n = zeros(1, N);
    for i = 1:N
        if i >=1 && i <= N
            w_n(i) = 0.54 - 0.46 * cos(2 * pi * (i - 1) / (N - 1)
               );
        end
    end
    h_n = hd_n .* w_n;
    h_n = h_n * A;
    end
```

Code for the downSampler is given below.

```
function y = downSample(x, n)
    y = x(1:n:end);
    end
```

Code for the upSampler is given below.

```
function y = upSample(x, n)
    N = length(x);
    y = zeros(1, N * n);
    y(1:n:end) = x;
    end
```

Code for Convolution is given below.

```
function [conv] = convol(signal, impulse)
    L = length(signal) + length(impulse) - 1;
    conv = zeros(1, L);

    for n = 1:L
        for k = 1:length(impulse)
            m = n - k + 1;
            if 1 <= m && m <= length(signal)
                conv(n) = conv(n) + signal(m) * impulse(k);
            end
        end
    end
    end
```

Code for Interpolation and Decimation is given below.

```
fs = 5000;
duration = 1;
t = linspace(0, duration, fs*duration);
N = 39;

% Create the input signal
f1 = 500;
f2 = 1000;
f3 = 700;

x1 = sin(2*pi*f1*t);
x2 = sin(2*pi*f2*t);
x3 = sin(2*pi*f3*t);

x = x1 + 2*x2 + 1.5*x3;
```

```
% Interpolate the signal
L = 2;
interPolated = interpolate(x, L, fs, N);

% Decimate the signal
M = 2;
decimated = decimate(interPolated, M, fs, N);

% Calculate the mean absolute error
meanError = mean(abs(x - decimated));
disp(meanError);

function interpolated = interpolate(x, L, fs, N)
    upSampled = upSample(x, L);
    LPF_fc = 1250;
    LPF_A = L;
    LPF_V = LPF(LPF_fc,fs,N,LPF_A);
    fullConv = convol(upSampled, LPF_V);
    interpolated = fullConv((length(LPF_V)-1)/2 + 1 : end - (
       length(LPF_V)-1)/2);
end

function decimated = decimate(x, M, fs, N)
    LPF_fc = 1250;
    LPF_A = 1;
    LPF_V = LPF(LPF_fc,fs,N,LPF_A);
    fullConv = convol(x, LPF_V);
    decimated = fullConv((length(LPF_V)-1)/2 + 1 : end - (length(
       LPF_V)-1)/2);
    decimated = downSample(decimated, M);
end
```

Observations

- The mean of absolute error between the original signal and the decimated signal is **0.0044**.
- This shows that the decimated signal is an almost perfect representation of the original signal.
- The interpolated signal has a higher sampling rate than the original signal.
- The decimated signal has a lower sampling rate than the original signal.
- The Low Pass Filter used for interpolation and decimation removes the high frequency components introduced by the zero padding and the decimation respectively.
- Gain in the LPF compensates for the loss of energy due to zero padding.

Conclusion

Interpolation increases signal sampling rates for improved fidelity, while decimation reduces data rates to conserve bandwidth. Even though the mean absolute error between the original signal and the decimated signal is small, the decimated signal is an almost perfect representation of the original signal. They are used for enhancing audio quality to optimizing data transmission in wireless networks, emphasizing the importance of these techniques in modern signal processing.