DSP LAB - Experiment 5

Design of FIR Filters

Ajay Krishnan K
EE22BTECH11003

February 24, 2024

Aim

To design a low-pass and a band-pass FIR filter using the windowing method and to verify the design using MATLAB and C.

Theory

FIR Filters

A Finite Impulse Response (FIR) filter is a type of digital filter. It is characterized by the fact that its impulse response is of finite duration. The impulse response of an LTI system is the output of the system when the input is an impulse. The impulse response of an FIR filter is always of finite duration because the output of the filter is a weighted sum of the input samples, and the weights are always of finite duration.

Windowing Method

The windowing method is a simple and effective technique for designing FIR filters. The basic idea is to design an ideal filter and then truncate and window the impulse response to obtain a practical filter. The ideal filter is designed by specifying the desired frequency response and then taking the inverse Fourier transform to obtain the impulse response. The impulse response is then truncated and windowed to obtain the practical filter.

Low-pass Filter

A low-pass filter is a filter that passes signals with a frequency lower than a certain cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. The ideal low-pass filter has a frequency response that is unity for frequencies less than the cutoff frequency and zero for frequencies greater than the cutoff frequency.

Band-pass Filter

A band-pass filter is a filter that passes signals with a frequency within a certain range and attenuates signals with frequencies outside the range. The ideal band-pass filter has a frequency response that is zero for frequencies less than the lower cutoff frequency, unity for frequencies between the lower and upper cutoff frequencies, and zero for frequencies greater than the upper cutoff frequency.

Experiment

Low-pass Filter

Design

The design of a low-pass filter involves the following steps:

1. Specify the filter order N and the cutoff frequency $\omega_c = \frac{2\pi f_c}{f_s}$.
2. Design the ideal low-pass filter impulse response $h_d[n]$ using the formula

$$h_d[n] = \begin{cases} \frac{\sin(\omega_c n)}{\pi n}, & \text{if } n \neq 0 \\ \frac{\omega_c}{\pi}, & \text{if } n = 0 \end{cases}, \quad \frac{-(N-1)}{2} \leq n \leq \frac{N-1}{2}$$

3. Window the impulse response using a window function $w(n)$ to obtain the final impulse response $h(n)$.

MATLAB Implementation

The following MATLAB code designs a low-pass filter using the windowing method and plots the frequency response of the filters.

```matlab
fs = 1600;
fc = 400;
N = 39;

[h_n] = LPF(fc, fs, N);

n = (-N + 1)/2 : (N - 1)/2;
stem(n, h_n, 'b', 'Marker', 'o', 'LineWidth', 1.5);
title('LPF Impulse Response with Hamming Window');
xlabel('n');
ylabel('Amplitude');
grid on;
legend('h[n]');

fvtool(h_n, 1, 'Fs', fs);
```

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

The corresponding frequency response of the filter is shown in Figure 1.

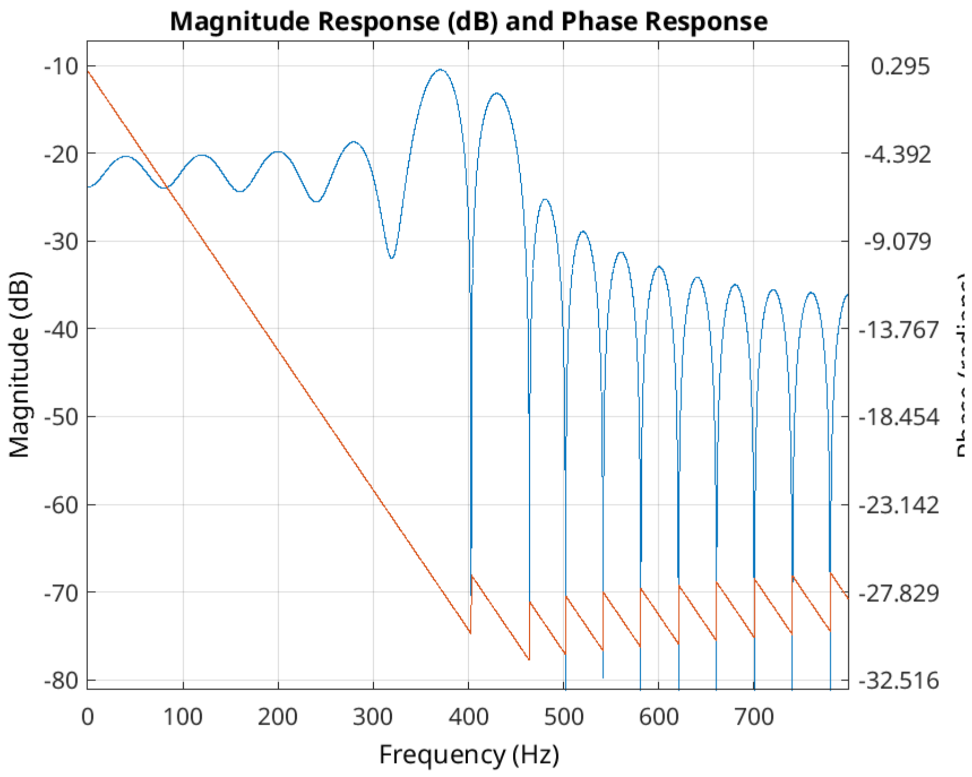

Figure 1: Frequency response of low-pass filter designed using MATLAB

The corresponding impulse response of the filter is shown in Figure 2.

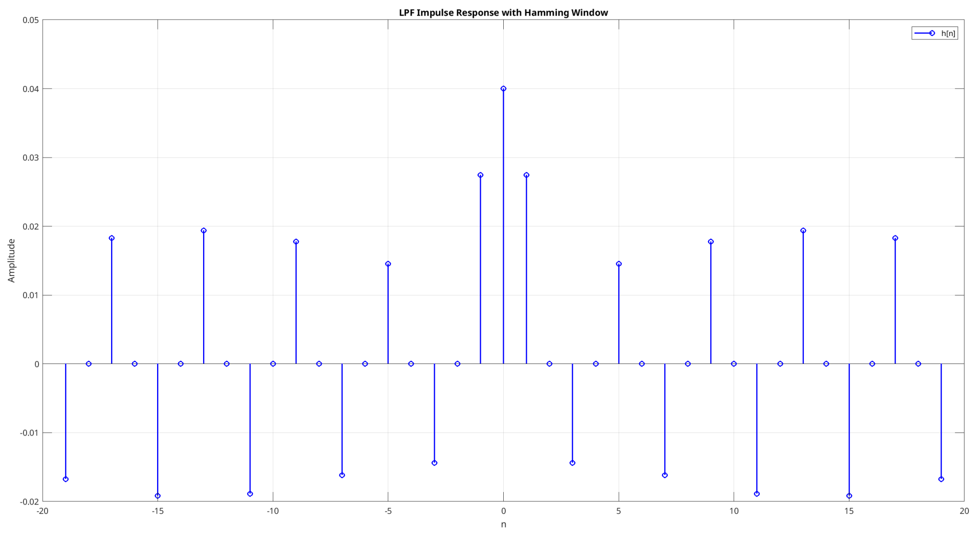

Figure 2: Impulse response of low-pass filter designed using MATLAB

C Implementation

The following C code designs a low-pass filter using the windowing method.

```c
#include <stdio.h>
#include <math.h>

#define N 39
#define PI 3.14159265358979323846

void hammingWindow(float h[]) {
    for (int n = 0; n < N; n++) {
        float window = 0.54 - 0.46 * cos(2 * PI * n / (N - 1));
        h[n] *= window;
    }
}

void LPF(float fc, float fs, float h[]) {
    float omega_c = 2 * PI * fc / fs;

    for (int n = -N / 2; n <= N / 2; n++) {
        if (n == 0) {
            h[n + N / 2] = omega_c / PI;
        } else {
            h[n + N / 2] = sin(omega_c * n) / (PI * n);
        }
    }
    hammingWindow(h);
}

int main() {
    float fs = 6000;
    float fc = 400;
    float h[N];

    LPF(fc, fs, h);

    printf("Impulse Response after Hamming Window:\n");
    for (int i = 0; i < N; i++) {
        printf("%f\n", h[i]);
    }

    return 0;
}
```

The corresponding impulse response of the filter is output as shown :

```
    Impulse Response after Hamming Window:
    0.001333
    0.001451
    0.001460
    0.001096
    0.000000
    -0.002113
    -0.005248
    -0.008961
    -0.012291
    -0.013839
    -0.012016
    -0.005401
    0.006852
    0.024684
    0.046948
    0.071465
    0.095316
    0.115327
    0.128656
    0.133333
    0.128656
    0.115327
    0.095316
    0.071465
    0.046948
    0.024684
    0.006852
    -0.005401
    -0.012016
    -0.013839
    -0.012291
    -0.008961
    -0.005248
    -0.002113
    0.000000
    0.001096
    0.001460
    0.001451
    0.001333
```

Band-pass Filter

Design

The design of a band-pass filter involves the following steps:

1. Specify the filter order N and the lower and upper cutoff frequencies $\omega_{c1} = \frac{2\pi f_{c1}}{f_s}$ and $\omega_{c2} = \frac{2\pi f_{c2}}{f_s}$.
2. Design the ideal band-pass filter impulse response $h_d[n]$ using the formula

$$h_d[n] = \begin{cases} \frac{\sin(\omega_{c2}n)-\sin(\omega_{c1}n)}{\pi n}, & \text{if } n \neq 0 \\ \frac{\omega_{c2}-\omega_{c1}}{\pi}, & \text{if } n = 0 \end{cases}, \quad \frac{-(N-1)}{2} \leq n \leq \frac{N-1}{2}$$

3. Window the impulse response using a window function $w(n)$ to obtain the final impulse response $h(n)$.

MATLAB Implementation

The following MATLAB code designs a band-pass filter using the windowing method and plots the frequency response of the filters.

```
fs = 6000;
fc1 = 500;
fc2 = 1200;
N = 39;

h_n = BPF(fc1, fc2, fs, N);


n = (-N + 1)/2 : (N - 1)/2;
stem(n, h_n, 'b', 'Marker', 'o', 'LineWidth', 1.5);
title('BPF Impulse Response with Hamming Window');
xlabel('n');
ylabel('Amplitude');
grid on;
legend('h[n]');

fvtool(h_n, 1, 'Fs', fs);

function [h_n] = BPF(fc1, fc2, fs, N)
n = (-N + 1)/2 : (N - 1)/2;
omega_c1 = 2 * pi * fc1 / fs;
```

```
omega_c2 = 2 * pi * fc2 / fs;
hd_n = (sin(omega_c2 * n) - sin(omega_c1 * n)) ./ (pi * n);
hd_n(N - (N - 1)/2) = (omega_c2 - omega_c1) / pi;
w_n = 0.54 - 0.46 * cos(2 * pi * n / (N - 1));
h_n = hd_n .* w_n;
end
```

The corresponding frequency response of the filter is shown in Figure 3.

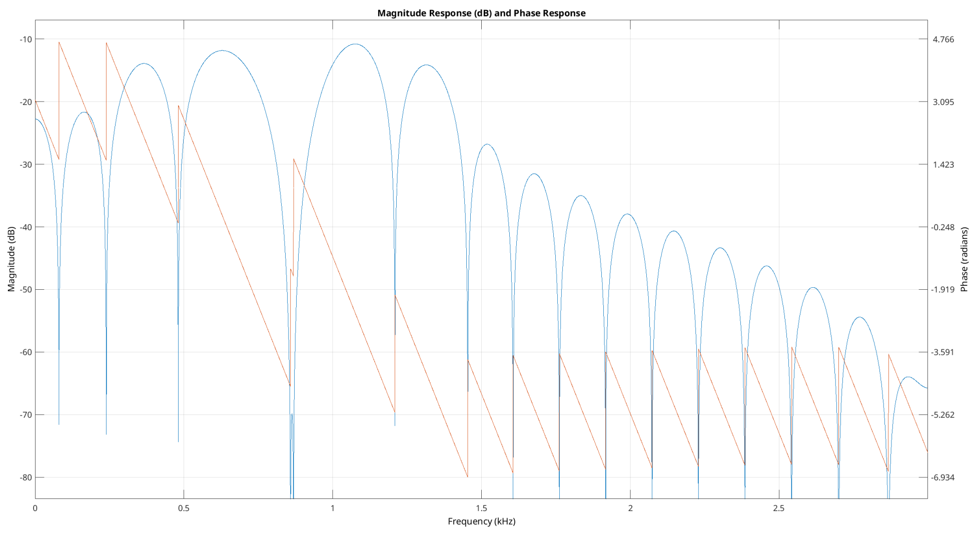

Figure 3: Frequency response of band-pass filter designed using MATLAB

The corresponding impulse response of the filter is shown in Figure 4.

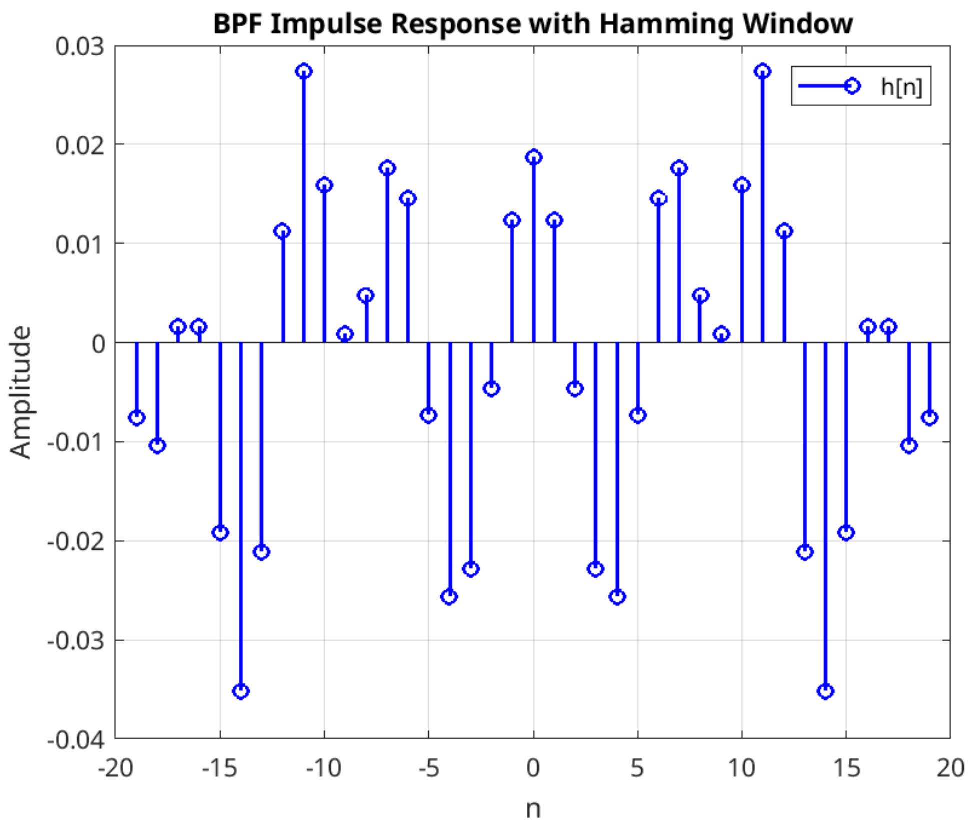

Figure 4: Impulse response of band-pass filter designed using MATLAB

C Implementation

The following C code designs a band-pass filter using the windowing method.

```c
#include <stdio.h>
#include <math.h>

#define N 39
#define PI 3.14159265358979323846

void hammingWindow(float h[]) {
    for (int n = 0; n < N; n++) {
        float window = 0.54 - 0.46 * cos(2 * PI * n / (N - 1));
        h[n] *= window;
    }
}

void BPF(float fc1, float fc2, float fs, float h[]) {
    float omega_c1 = 2 * PI * fc1 / fs;
    float omega_c2 = 2 * PI * fc2 / fs;

    for (int n = -N / 2; n <= N / 2; n++) {
        if (n == 0) {
            h[n + N / 2] = (omega_c2 - omega_c1) / PI;
        } else {
            h[n + N / 2] = (sin(omega_c2 * n) - sin(omega_c1 * n)
               ) / (PI * n);
        }
    }

    hammingWindow(h);
}

int main() {
    float fs = 6000;
    float fc1 = 500;
    float fc2 = 1200;
    float h[N];
    BPF(fc1, fc2, fs, h);

    printf("Impulse Response after Hamming Window:\n");
    for (int i = 0; i < N; i++) {
        printf("%f\n", h[i]);
    }

    return 0;
}
```

The corresponding impulse response of the filter is output as shown :

```
    Impulse Response after Hamming Window:
    -0.000605
    -0.000897
    0.000172
    0.000229
    -0.003756
    -0.009438
    -0.007682
    0.005538
    0.017933
    0.013839
    0.001001
    0.007228
    0.035851
    0.039940
    -0.027106
    -0.130573
    -0.159129
    -0.043180
    0.142675
    0.233333
    0.142675
    -0.043180
    -0.159129
    -0.130573
    -0.027106
    0.039940
    0.035851
    0.007228
    0.001001
    0.013839
    0.017933
    0.005538
    -0.007682
    -0.009438
    -0.003756
    0.000229
    0.000172
    -0.000897
    -0.000605
```

Observations

Low-pass Filter

The frequency response of the low-pass filter designed using MATLAB is shown in Figure 1. TH cutoff frequency of the filter and the transition band can be observed from the plot. Based on the given fs and fc, the filter was designed to have higher amount of ripples in the passband and stopband. This can be reduced by increasing the filter order. By the same way, the transition band can be reduced by increasing the filter order.

The impulse response of the low-pass filter designed using MATLAB is shown in Figure 2. The impulse response of the filter is of finite duration and the windowing method has been used to obtain the practical filter. The use of windowing method can be observed from the impulse response.

Band-pass Filter

The frequency response of the band-pass filter designed using MATLAB is shown in Figure 3. The lower and upper cutoff frequencies of the filter and the transition band can be observed from the plot. Based on the given fs and fc1, fc2, the filter was designed to have higher amount of ripples in the passband and stopband. This can be reduced by increasing the filter order. By the same way, the transition band can be reduced by increasing the filter order.

The impulse response of the band-pass filter designed using MATLAB is shown in Figure 4. The impulse response of the filter is of finite duration and the windowing method has been used to obtain the practical filter. The use of windowing method can be observed from the impulse response.

Conclusion

The low-pass and band-pass FIR filters have been designed using the windowing method and the design has been verified using MATLAB and C. The frequency response and impulse response of the filters have been plotted and the observations have been made.

From the observations, it can be concluded that the windowing method is an effective technique for designing FIR filters. The stopband and passband ripples can be reduced by increasing the filter order and the transition band can be reduced by increasing the filter order. Using effective windowing functions can also help in reducing the ripples in the passband and stopband. Optimising the filter order and the windowing function can help in obtaining the desired frequency response.