Experiment-5

EE:2801 DSP-Lab
Indian Institute of Technology, Hyderabad

Jay Vikrant
EE22BTECH11025

I. Aim of the experiment

Design of digital filters such as Low Pass Filter (LPF), Band Pass Filter(BPF).

II. Theory of Digital FIR

Finite Impulse Response (FIR) filters are fundamental in digital signal processing (DSP), extensively applied in audio processing, image processing, and communications. This technical overview delves into the core concepts, characteristics, design methodologies, and applications of FIR filters. An FIR filter operates on a finite number of input samples, producing an output sequence through a weighted sum of these samples. Mathematically, the output sequence $y[n]$ is given by the convolution of the input sequence $x[n]$ and a set of coefficients b_k:

$$y[n] = \sum_{k=0}^{N} a_k \cdot x[n-k]$$

Here, N is the order of the filter, determining the number of delay elements required. The coefficients b_k are chosen to achieve the desired frequency response.

III. Characteristics

- **Linear Phase:** FIR filters exhibit a linear phase response, ensuring that all frequency components experience the same delay.
- **Stability:** FIR filters are inherently stable, guaranteeing that the output signal remains bounded over time. This stability is vital in control systems.
- **Output Response:** The output response to an input impulse eventually decays to zero, making FIR filters suitable for applications with bounded response times, like radar signal processing.
- **Easy to Implement:** FIR filters can be implemented using basic arithmetic operations such as multiplication and addition.
- **Design Flexibility:** Adjusting the filter coefficients allows for a wide range of frequency responses.

IV. Designing FIR Filters

Designing FIR filters involves selecting coefficients to achieve a desired frequency response. There are Several methods to implent this one of which is:

- **Windowing Method:** Utilizes a window function to truncate the ideal impulse response, influencing characteristics like band width and stopband attenuation.

This method will be used in this experiment to demonstrate digital FIR.

V. Code , Output and Plot of the simulations

Matlab simulation,

```
function main5()

N = 39;

%lpf
wc = pi/2;
fc= 400;
fs_lpf = (2*pi*fc)/(wc);
h1 = lpf_func(fc,fs_lpf,N).*hw(N);
fvtool(h1);
disp('h1');
disp(h1);
x1 = 1 :N;
figure
stem(x1,h1)
xlabel('N−sample')
ylabel('lpf␣x␣wd␣(Impulse)')
grid on;

%bpf
fc1 = 500;
fc2 = 1200;
fs_bpf = 6000;
h2 = bpf_func(fc1,fc2,fs_bpf,N).*hw(N);
fvtool(h2);
disp('h2');
disp(h2);
x2 = 1:N;
figure
stem(x2,h2)
xlabel('N−sample')
ylabel('bpf␣x␣wd␣(Imppulse)')
grid on;

end

function y = lpf_func(fc,fs,N)

wc = (2*pi*fc)/(fs);
n = 1;
y = zeros(1,N−1);
for k = −(N−1)/2:(N−1)/2
    if k == 0
        y(n) = wc/pi;
    else
        y(n) = sin((wc*k))/(pi*k);
    end
    n = n + 1;
end
end

function y = bpf_func(fc1,fc2,fs,N) %bandpass

wc1 = (2*pi*fc1)/(fs);
wc2 = (2*pi*fc2)/(fs);
n = 1;
y = zeros(1,N-1);
for k = -(N-1)/2: (N-1)/2
    if k == 0
        y(n) = (wc2 - wc1)/pi;
    else
        y(n) = (sin((wc2*k))/(pi*k)) - (sin((wc1*k))/(pi*k));
    end
    n = n + 1;
end
end

function y = hw(N) %window function

n = 1;
y = zeros(1,N);
for k = 0 : (N-1)
    if k >= 0 && k <= (N-1)
        y(n) = 0.54 - 0.46*cos((2*pi*k)/(N-1));
    else
        y(n) = 0;
    end
    n = n + 1;
end
end
```

The following got computed in Matlab,

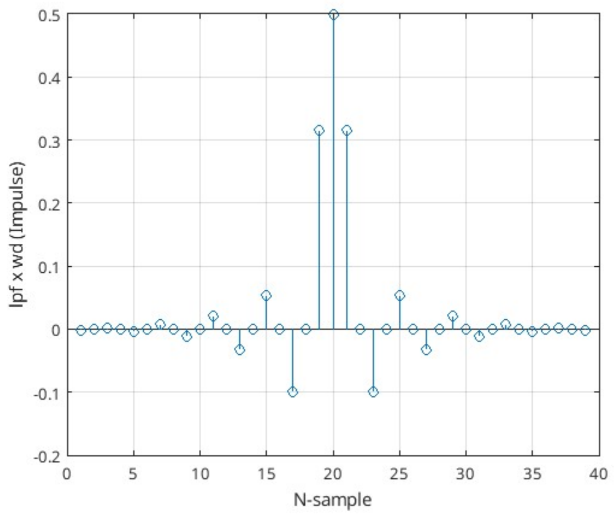


Fig. 0. Impulse response plot of lpf

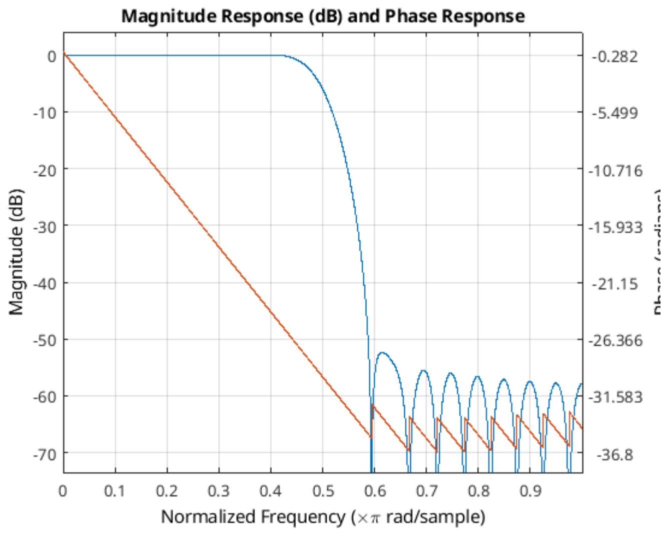


Fig. 0. The magnitude plot of the impulse response of lpf using 'fvtool' cmd

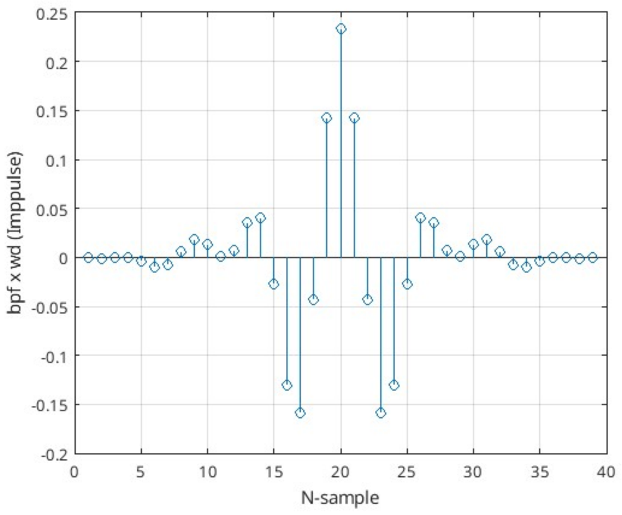


Fig. 0. Impulse response plot of bpf

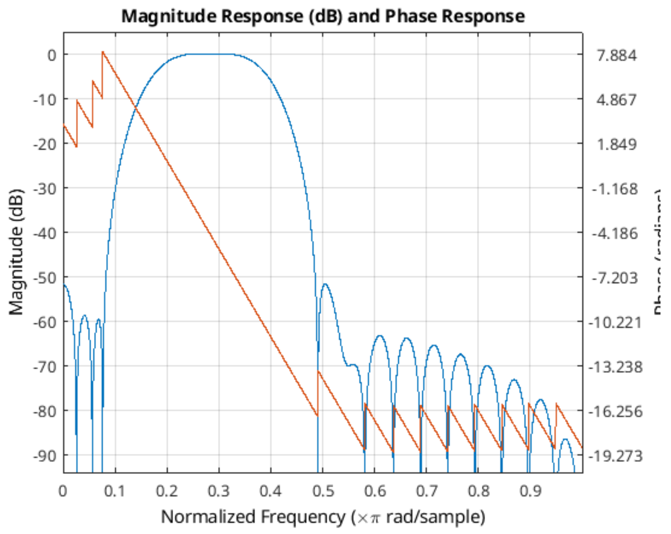
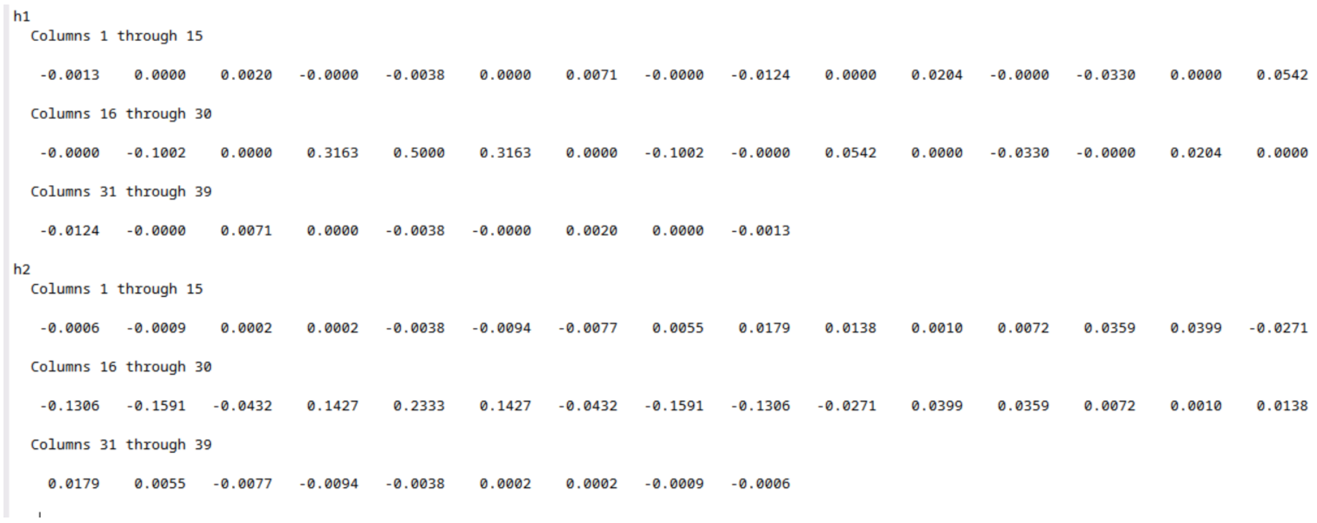


Fig. 0. The magnitude plot of the impulse response of bpf using 'fvtool' cmd



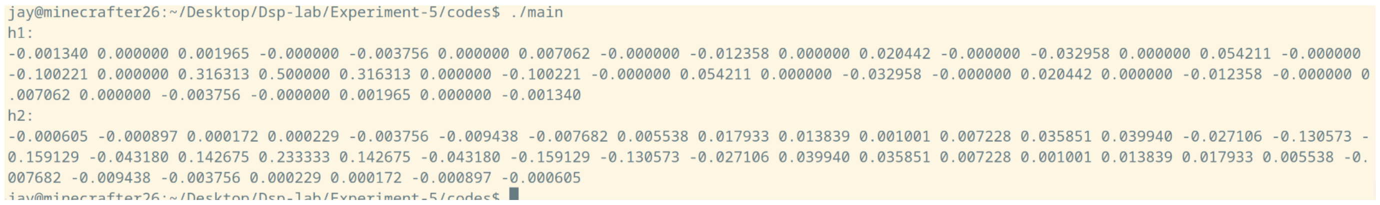
```
h1
  Columns 1 through 15

   -0.0013    0.0000    0.0020   -0.0000   -0.0038    0.0000    0.0071   -0.0000   -0.0124    0.0000    0.0204   -0.0000   -0.0330    0.0000    0.0542

  Columns 16 through 30

   -0.0000   -0.1002    0.0000    0.3163    0.5000    0.3163    0.0000   -0.1002   -0.0000    0.0542    0.0000   -0.0330   -0.0000    0.0204    0.0000

  Columns 31 through 39

   -0.0124   -0.0000    0.0071    0.0000   -0.0038   -0.0000    0.0020    0.0000   -0.0013

h2
  Columns 1 through 15

   -0.0006   -0.0009    0.0002    0.0002   -0.0038   -0.0094   -0.0077    0.0055    0.0179    0.0138    0.0010    0.0072    0.0359    0.0399   -0.0271

  Columns 16 through 30

   -0.1306   -0.1591   -0.0432    0.1427    0.2333    0.1427   -0.0432   -0.1591   -0.1306   -0.0271    0.0399    0.0359    0.0072    0.0010    0.0138

  Columns 31 through 39

    0.0179    0.0055   -0.0077   -0.0094   -0.0038    0.0002    0.0002   -0.0009   -0.0006
```

Fig. 0. Impulse response of lpf(h1) and bpf(h2)

C simulation,

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

#define N 39
#define M_PI 3.14159265358979323846

void lpf_func(double fc, double fs, double* h) {
    double wc = (2 * M_PI * fc) / fs;
    for (int k = −(N − 1) / 2, n = 0; k <= (N − 1) / 2; ++k, ++n) {
        if (k == 0) {
            h[n] = (wc / M_PI) ;
        } else {
            h[n] = (sin(wc * k) / (M_PI * k)) ;
        }
    }
}

void bpf_func(double fc1, double fc2, double fs, double* h) {
    double wc1 = (2 * M_PI * fc1) / fs;
    double wc2 = (2 * M_PI * fc2) / fs;
    for (int k = −(N − 1) / 2, n = 0; k <= (N − 1) / 2; ++k, ++n) {
        if (k == 0) {
            h[n] = ((wc2 − wc1) / M_PI) ;
        } else {
            h[n] = ((sin(wc2 * k) / (M_PI * k)) − (sin(wc1 * k) / (M_PI * k)));
        }
    }
}

double hw(int i) {
    double window[N];
    for (int k = 0; k < N; ++k) {
        window[k] = 0.54 − 0.46 * cos((2 * M_PI * k) / (N − 1));
    }
    return window[i];
}

int main() {
    double h1[N], h2[N];
    double fc = 400, fs_lpf = 1600;
    lpf_func(fc, fs_lpf, h1);

    double fc1 = 500, fc2 = 1200, fs_bpf = 6000;
    bpf_func(fc1, fc2, fs_bpf, h2);


    printf(”h1:\n”);
    for (int i = 0; i < N; ++i) {
        printf("%f␣", h1[i]*hw(i));
    }
    printf("\n");

    printf("h2:\n");
    for (int i = 0; i < N; ++i) {
        printf("%f␣", h2[i]*hw(i));
    }
    printf("\n");

    return 0;
}
```

The following got computed in C,



```
jay@minecrafter26:~/Desktop/Dsp-lab/Experiment-5/codes$ ./main
h1:
-0.001340 0.000000 0.001965 -0.000000 -0.003756 0.000000 0.007062 -0.000000 -0.012358 0.000000 0.020442 -0.000000 -0.032958 0.000000 0.054211 -0.000000
-0.100221 0.000000 0.316313 0.500000 0.316313 0.000000 -0.100221 -0.000000 0.054211 0.000000 -0.032958 -0.000000 0.020442 0.000000 -0.012358 -0.000000 0
.007062 0.000000 -0.003756 -0.000000 0.001965 0.000000 -0.001340
h2:
-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
jay@minecrafter26:~/Desktop/Dsp-lab/Experiment-5/codes$
```

Fig. 0. array of lpf and bpf

VI. Observations and Conclusion

In this experiment, we conducted a digital Finite Impulse Response (FIR) filter implementation to create both a Low-Pass Filter (LPF) and a Band-Pass Filter (BPF). The design of these filters utilized the windowing method, the window function length influenced the trade-off between the width of the main lobe and the levels of sidelobes in the frequency response in the case lpf and bpf.

The LPF demonstrated efficient attenuation of high-frequency components with a smooth roll-off near the cutoff(400Hz). The performance of the LPF was influenced by the choice of window function, impacting parameters such as the band width and sidelobe levels. Similarly, the BPF successfully exhibited a passband centered around the frequency(500 to 1200 Hz), effectively suppressing frequencies outside the desired band.