Project Report: EESC 6360, Digital Signal Processing I

Topic:
To develop a usable software package to design linear phase FIR filters.

Parth Parikh (UTD ID: 2021165845)
Department of Electrical Engineering, The University of Texas at Dallas
pxp126130@utdallas.edu

To develop a usable software package to design linear phase FIR filters.

Parth Parikh (UTD ID: 2021165845)
Department of Electrical Engineering, The University of Texas at Dallas
pxp126130@utdallas.edu

I. INTRODUCTION

Digital Filters form one of the most important tools in Digital Signal Processing. Digital Filtering is involved in as simple tasks as separating two signals, to important tasks like eliminating noise from medical signal data. Thus signal filtering must be performed carefully and accurately. Digital Filters provide the much needed ease and accuracy to perform this task.

The design process for such filters generally includes specifications like type of filter, nature of frequency response, cut off frequencies, transition widths and filter order. It is after this crucial process of data gathering, that actual designing is carried out.

In this project we concentrate on designing FIR Filters of the following types:

- Low Pass Filter
- High Pass Filter
- Bad Pass Filter
- Band Stop Filter

In order to achieve finite length filters, windowing operation is performed and the following windows are included in the scope of this project:

- Rectangular
- Hamming
- Kaiser

Thus this project explores 3 versions of each type of Filter, giving us the full depth and understanding of the way FIR filters function.

In this project we implement afore mentioned filter combinations and study their frequency responses. In section 2 of this report we discuss the problem definition and its relevance in detail. Section 3 is a discussion of FIR filter design. Section 4 contains the crucial learning from this project. Section 5 results and discussion. Section 6 concludes the report followed by the codes.

II. PROBLEM DEFINITION

To develop a usable software package to design linear phase FIR filters.

The essence of Digital Filters is that we design a finite impulse response function that serves the purpose of filtering as closely as an Ideal (infinite impulse response) filter, but is practically achievable to be implemented.

To keep the package as realistic as possible, all user inputs are in the form of CT frequencies thus making this an even challenging and an interesting project. To keep the navigation user friendly, extensive error handling has been performed.

Also, all attempts have been made to keep the design as modular as possible. Thus making changes to a particular section or design feasible. The filter designs are tested for various values of frequencies and other specification to ensure the integrity of the package.

III. FIR FILTERS AND WINDOW DESIGN

A. Low Pass Filter

A low pass filter is one that allows frequencies below the cut off frequency to pass through un-attenuated and attenuates the ones above that. And ideal low pass filter is governed by the equation:

$$H = \begin{cases} 1, & f \le f_c \\ 0, & otherwise \end{cases} \cdots (1)$$

The response of an ideal LPF looks like Fig A.

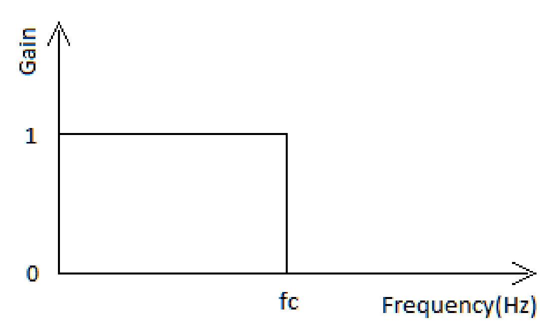


Fig. A: Ideal Low Pass Filter frequency response.

B. High Pass Filter

A High pass filter is the exact negated version of a low pass filter. It allows frequencies greater than a certain cut off to pass though, but attenuates all the frequencies lower to it. The governing equation is given below:

$$H = \begin{cases} 1, & f \geq f_c \\ 0. & otherwise \end{cases} \cdots (2)$$

The response of an ideal HPF looks like Fig B.

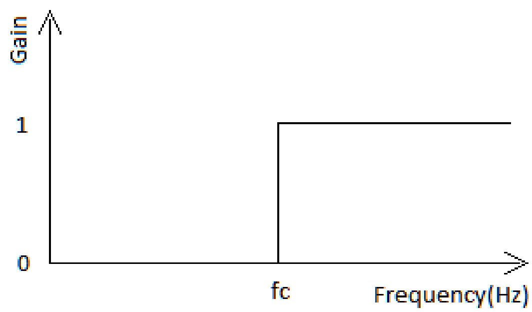


Fig. B: Ideal High Pass Filter frequency response.

C. Band Pass Filter

A band pass filter is one which allows frequencies that lie within a range to pass through, and attenuates the rest. A typical application of this filter is to selectively choose a set of frequencies from a mixed signal input. Equation 3 depicts a band pass filter in the frequency domain.

$$H = \begin{cases} 1, & f_{c1} \leq f \leq f_{c2} \\ 0, & otherwise \end{cases} \cdots (3)$$

Frequency response of the same is shown below.

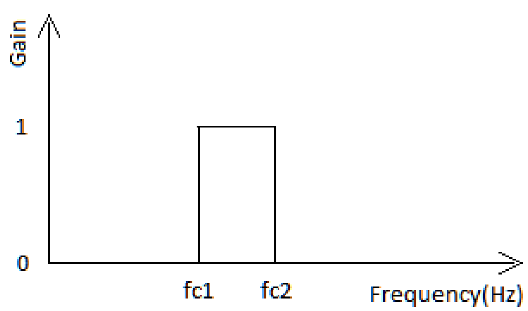


Fig. C: Ideal Band Pass Filter frequency response.

D. Band Stop Filter

Like HPF and LPF, BPF and BSF are complementary filters. Band stop filters are ones that notch out a range of frequencies, maintaining zero attenuation at all others. Major applications are to separate two signals, or supress noise of a particular frequency range. The equation for this filter is shown below.

$$H = \begin{cases} 1, & f \leq f_{c1} \\ 1, & f \geq f_{c2} \\ 0, & otherwise \end{cases} \cdots (4)$$

The frequency response of the same is given as under

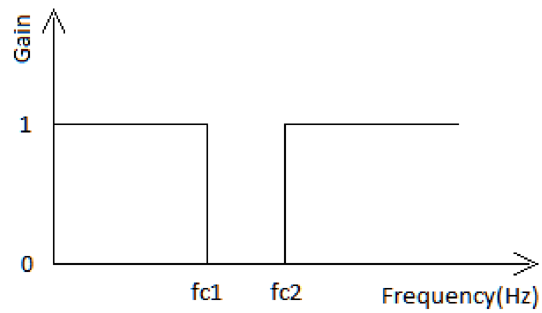
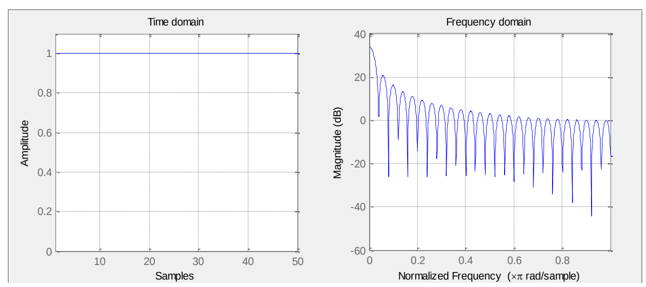


Fig. D: Ideal Band Stop Filter frequency response.

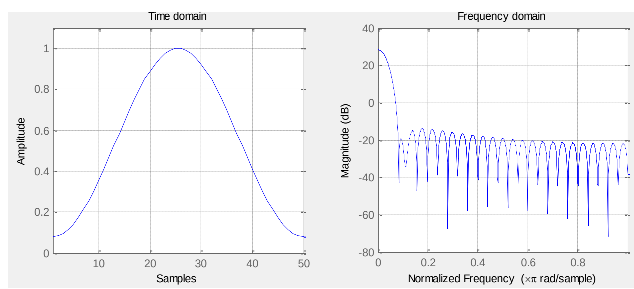
E. Window Designs

Following are the three types of Windows used in this project:

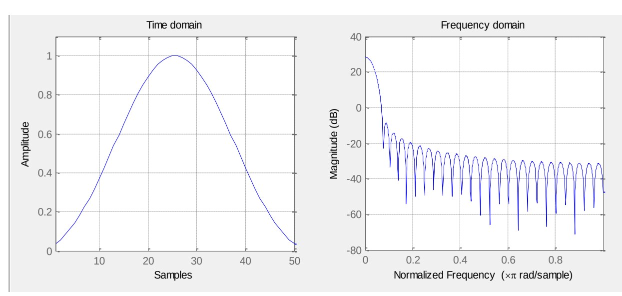
- Rectangular



- Hamming



- Kaiser



IV. LEARNING

Not only did this project teach more about FIR filters; it gave a very close hands-on experience at using and developing packages for specific functions. The highlight of learning through the project is as under:

- FIR design is a complex task and involves various constraints that need to be satisfied for a filter to give the intended performance.
- It is important to develop modular packages so as to ease further debugging and reusability of code.
- The shorter the transition width, the higher the order of the filter.
- Practical implementation of concepts discussed in class.
- The project report also gave a rich experience on writing documents in the IEEE format.

V. RESULTS AND ANALYSIS

Below are shown several sample filter responses and screen shots from the package designed for this project. To facilitate error handling and validation, radio buttons have been used to select types of filters and windows.

The selection of order based on the transition width makes for an efficient and accurate filter. Observations lead to the filtering operation based on Kaiser window gives the most smoothed out curve.

VI. CONCLUSION

Filters based on the rectangular window have the lowest order; however the collateral expense is that the frequency response is not as smooth. The smoothening increases with as we move towards the Kaiser window; however, this increases the order of the filter to a much higher value.

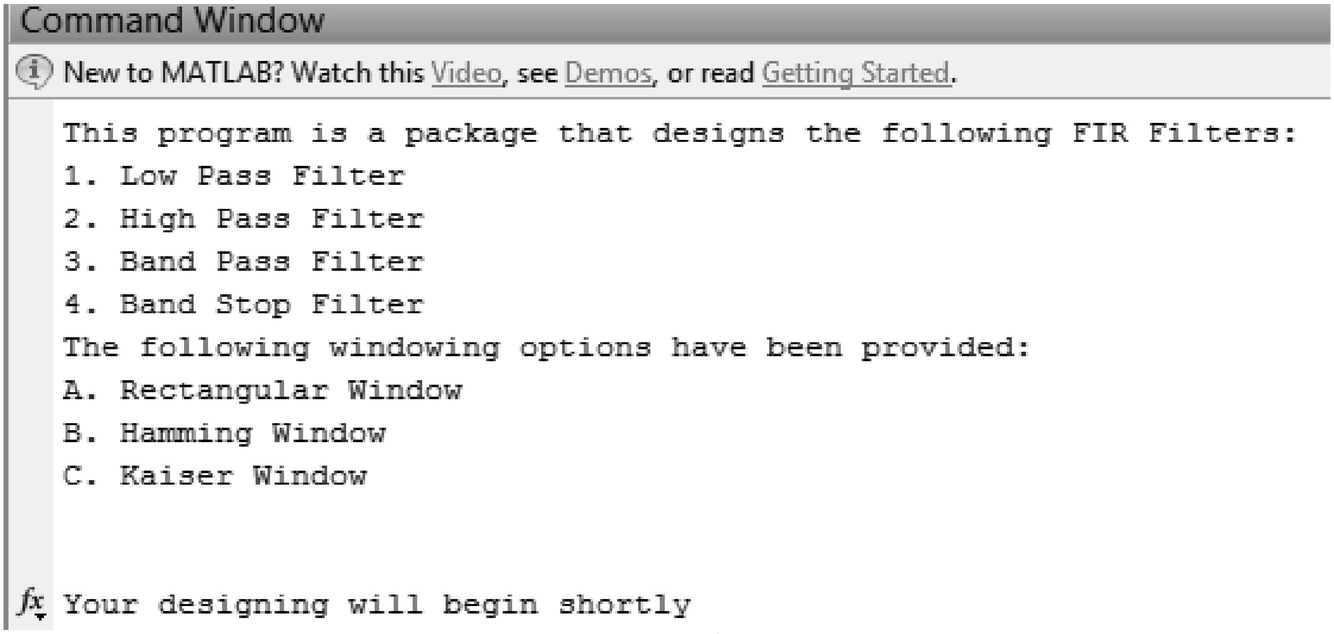


Fig. E: First screen of the package

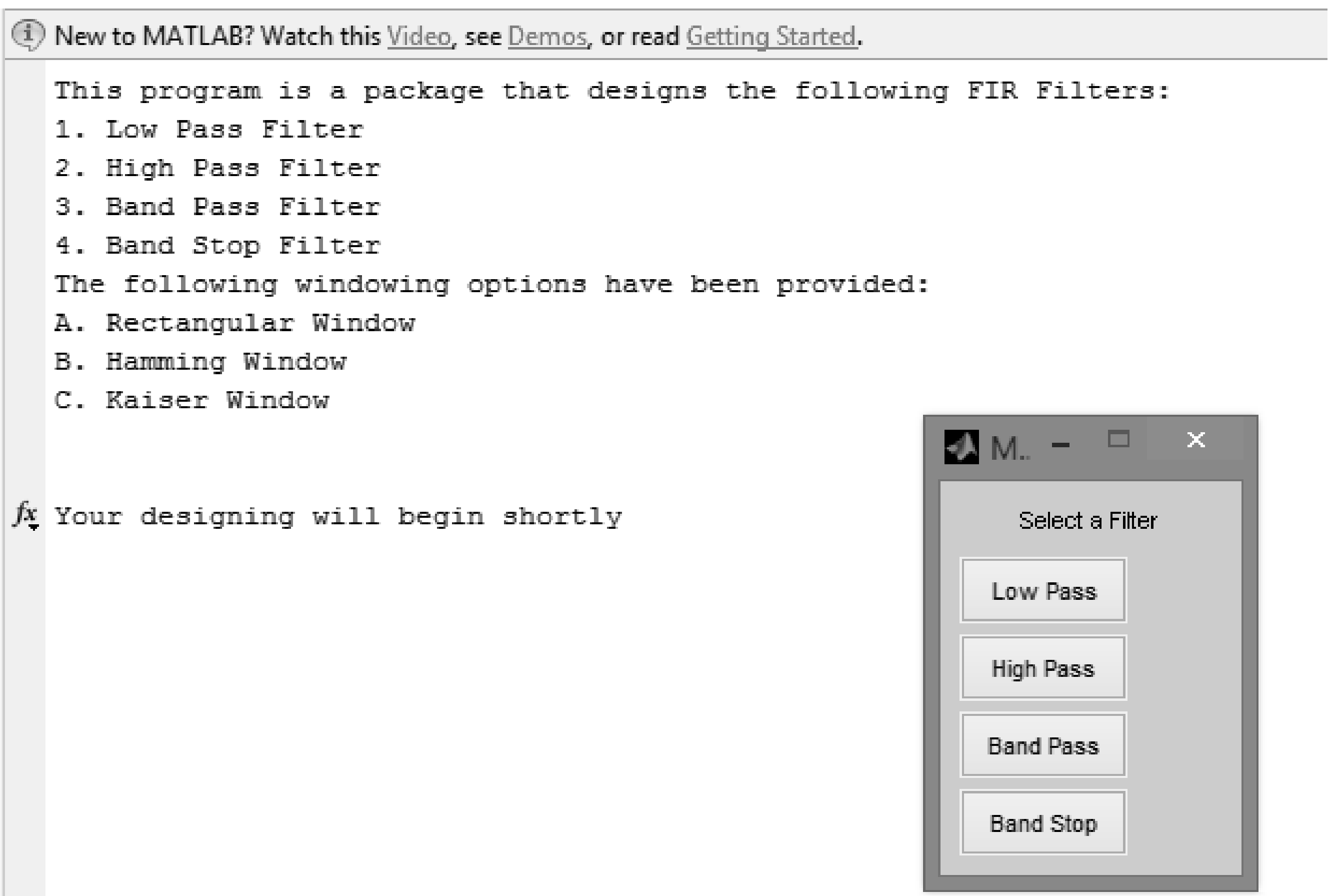


Fig F: Filter Selection Tool

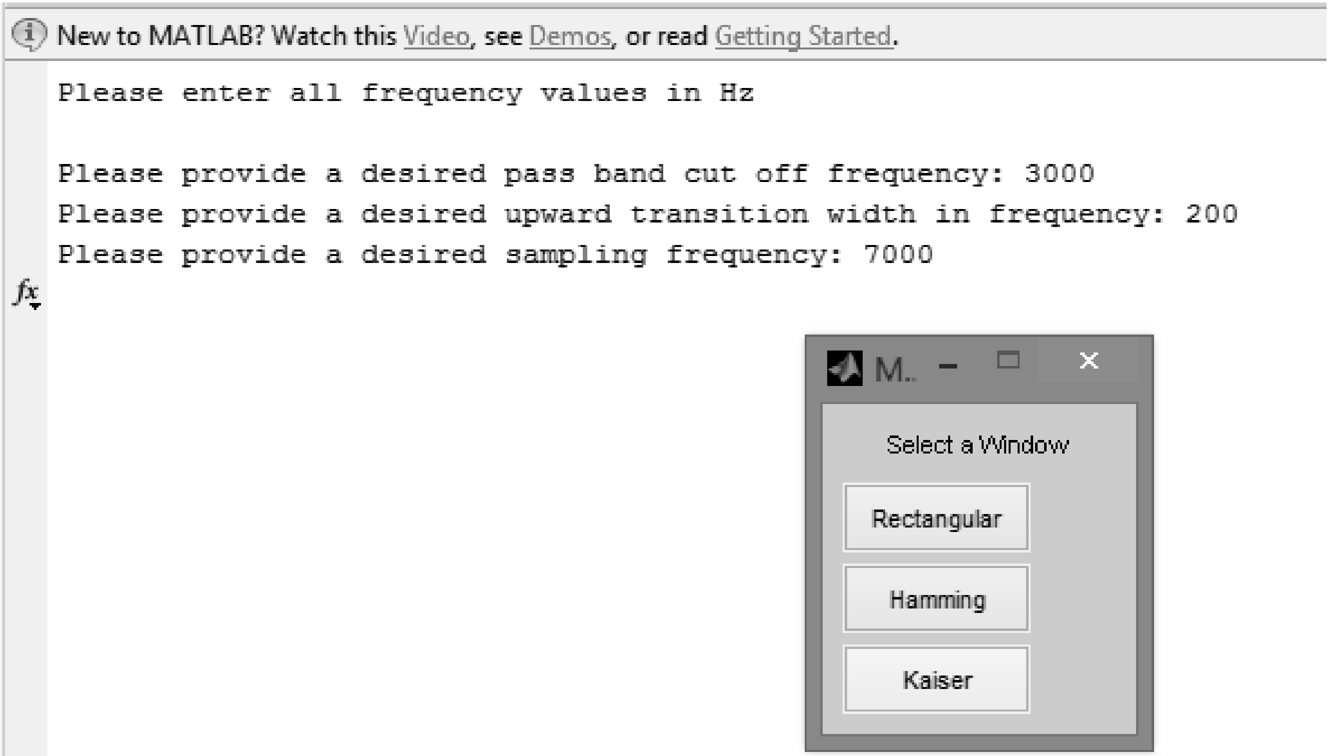


Fig. G: Window Selection Tool

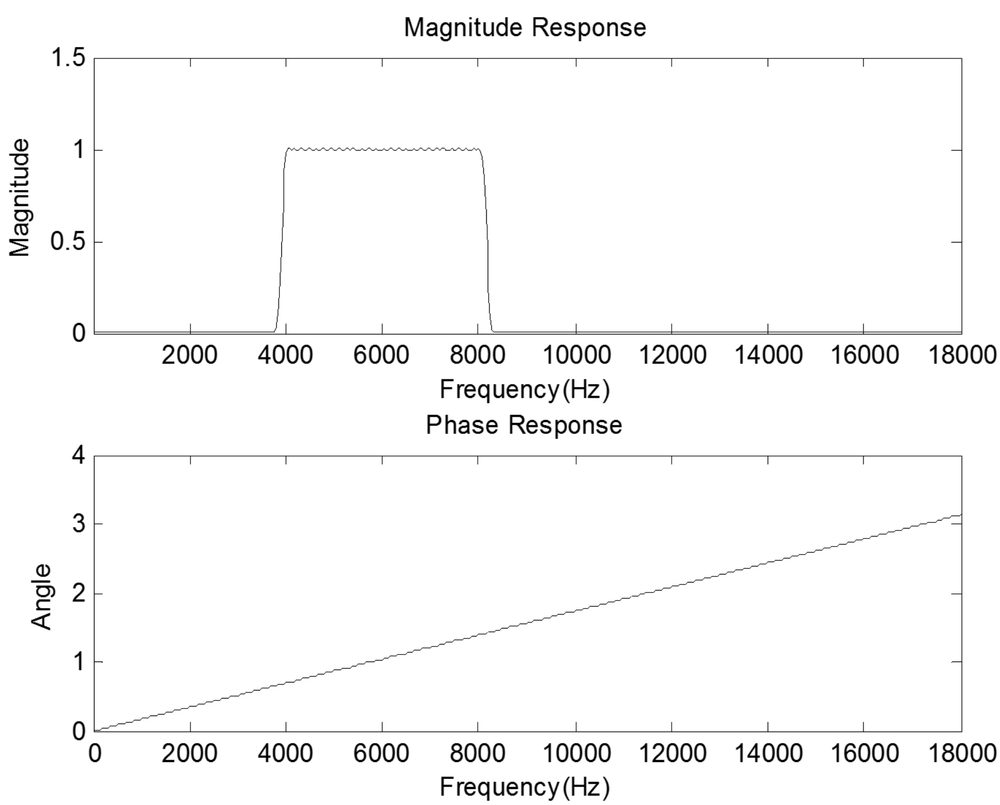


Fig. H: Band Pass Filter using Hamming Window

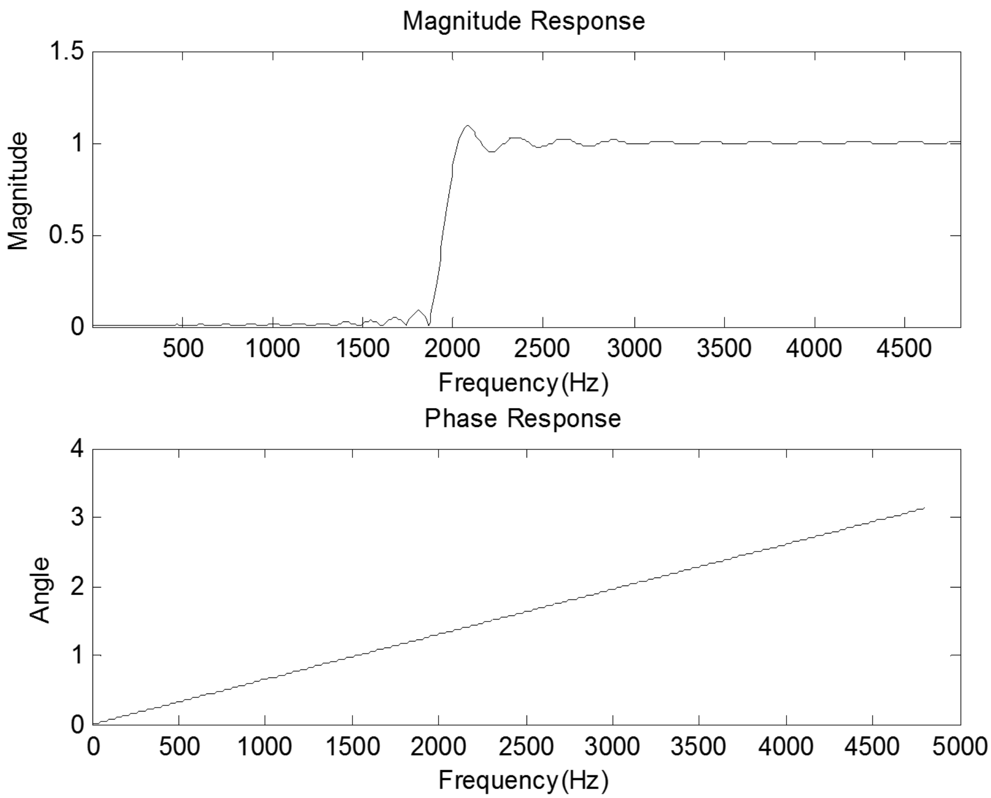


Fig. I: High Pass Filter using Rectangular Window

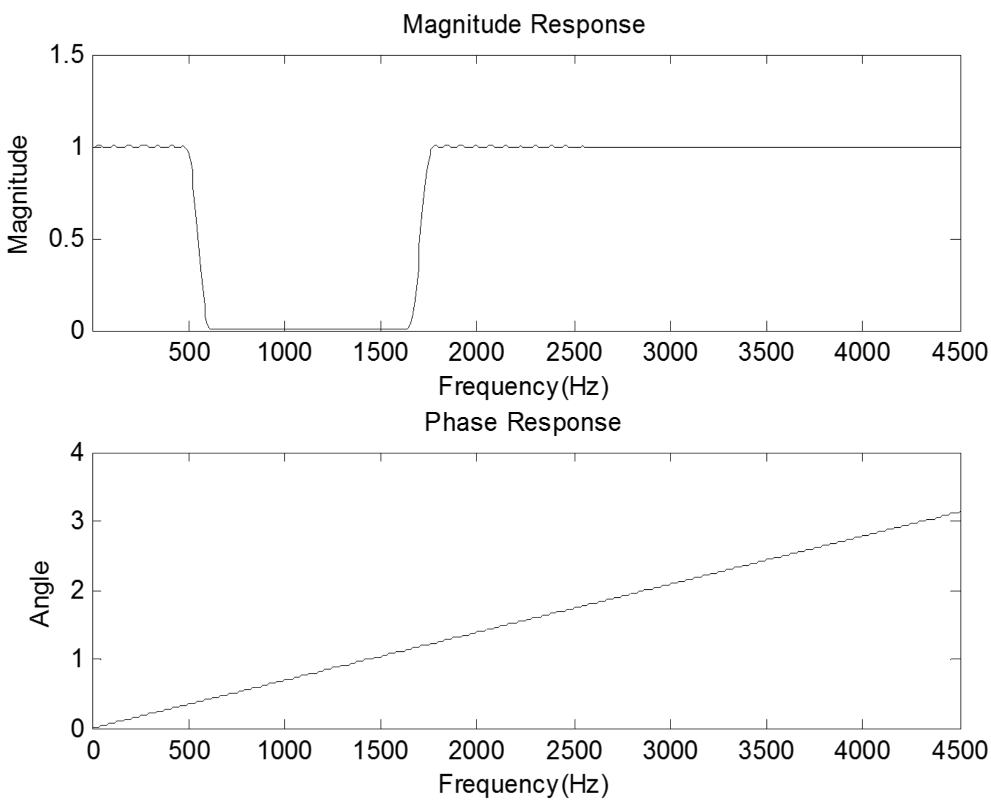


Fig. K: Band Stop Filter using Kaiser Window

PROGRAMS

- Front end: myfilter.m

```
clear all;
close all;
clc;

e = input('Press enter to start this pakage:');
clc;
display('This program is a package that designs the following FIR Filters:');
fprintf('1. Low Pass Filter\n');
fprintf('2. High Pass Filter\n');
fprintf('3. Band Pass Filter\n');
fprintf('4. Band Stop Filter\n');
fprintf('The following windowing options have been provided:\n');
fprintf('A. Rectangular Window\n');
fprintf('B. Hamming Window\n');
fprintf('C. Kaiser Window\n\n');
fprintf('\nYour designing will begin shortly');
pause(2)
ch = menu('Select a Filter','Low Pass','High Pass','Band Pass','Band Stop');

if(ch == 1)

    fprintf('\nYou have selected a Low Pass filter\n');
    pause(2)
    [mag, phase,N,Fcp,Fcs,Fs] = lpfcode();

elseif(ch == 2)

    fprintf('\nYou have selected a High Pass filter\n');
    pause(2)
    [mag, phase,N,Fcp,Fcs,Fs] = hpfcode();

elseif(ch == 3)

    fprintf('\nYou have selected a Band Pass filter\n');
    pause(2)
    [mag,phase,N,Fcp1,Fcs1,Fcp2,Fcs2,Fs] = bpfcode();

elseif(ch == 4)

    fprintf('\nYou have selected a Band Stop filter\n');
    pause(2)
    [mag,phase,N,Fcp1,Fcs1,Fcp2,Fcs2,Fs] = bsfcode();

end

n = linspace(1,Fs,512);
subplot(2,1,1)
plot(n,mag,'r')
title('Magnitude Response');
xlabel('Frequency(Hz)');
ylabel('Magnitude');
axis([1,Fs,0,1.5]);
```

```
subplot(2,1,2)
plot(n,phase,'b')
title('Phase Response');
xlabel('Frequency(Hz)');
ylabel('Angle');
```

- Low pass Filter controlling code

```
function [mag,phase,N,Fcp,Fcs,Fs] = lpfcode()

    [Fcp,Fcs,Fs] = lpfvalidate();
    ch = menu('Select a Window','Rectangular','Hamming','Kaiser');

   if(ch == 1)
        [mag,phase,N] = lpfrect(Fcp,Fcs,Fs);

    elseif(ch == 2)
        [mag,phase,N] = lpfhamming(Fcp,Fcs,Fs);

    elseif(ch == 3)
        [mag,phase,N] = lpfkaiser(Fcp,Fcs,Fs);

    end
end
```

- High pass Filter controlling code

```
function [mag,phase,N,Fcp,Fcs,Fs] = hpfcode()

    [Fcp,Fcs,Fs] = hpfvalidate();
    ch = menu('Select a Window','Rectangular','Hamming','Kaiser');

    if(ch == 1)
        [mag,phase,N] = hpfrect(Fcp,Fcs,Fs);

    elseif(ch == 2)
        [mag,phase,N] = hpfhamming(Fcp,Fcs,Fs);

    elseif(ch == 3)
        [mag,phase,N] = hpfkaiser(Fcp,Fcs,Fs);

    end
end
```

- Band Pass Filter Controlling Code

```
function [mag,phase,N,Fcp1,Fcs1,Fcp2,Fcs2,Fs] = bpfcode()

    [Fcp1,Fcs1,Fcp2,Fcs2,Fs] = bpfvalidate();
    ch = menu('Select a Window','Rectangular','Hamming','Kaiser');

    if(ch == 1)
        [mag,phase,N] = bpfrect(Fcp1,Fcs1,Fcp2,Fcs2,Fs);

    elseif(ch == 2)
        [mag,phase,N] = bpfhamming(Fcp1,Fcs1,Fcp2,Fcs2,Fs);
```

```
    elseif(ch == 3)
        [mag,phase,N] = bpfkaiser(Fcp1,Fcs1,Fcp2,Fcs2,Fs);

    end
end
```

- Band Stop Filter Controlling Code

```
function [mag,phase,N,Fcp1,Fcs1,Fcp2,Fcs2,Fs] = bsfcode()

    [Fcp1,Fcs1,Fcp2,Fcs2,Fs] = bsfvalidate();
     ch = menu('Select a Window','Rectangular','Hamming','Kaiser');

    if(ch == 1)
        [mag,phase,N] = bsfrect(Fcp1,Fcs1,Fcp2,Fcs2,Fs);

    elseif(ch == 2)
        [mag,phase,N] = bsfhamming(Fcp1,Fcs1,Fcp2,Fcs2,Fs);

    elseif(ch == 3)
        [mag,phase,N] = bsfkaiser(Fcp1,Fcs1,Fcp2,Fcs2,Fs);

    end
end
```

- LPF Data Validation

```
function [Fcp,Fcs,Fs] = lpfvalidate()
    er = 1;
    while(er == 1)
    clc;
    [Fcp,Fcs,Fs] = fselectlpf();

    if(Fcp > Fcs)
    fprintf('\n\nPass Band cut off should be greater than Stop Band cut off\n');
    fprintf('Please re-enter');

    pause(5)

     elseif(Fcs >= Fs/2)
    fprintf('\n\nStop Band cut off should be less than half the Sampling 
Frequency\n');
    fprintf('Please re-enter');
    pause(5)

    else
     er = 0;

    end
    end
end
```

- HPF Data Validation

```
function [Fcp,Fcs,Fs] = hpfvalidate()
    er = 1;
    while(er == 1)
    clc;
    [Fcp,Fcs,Fs] = fselecthpf();
```

```
    if(Fcp < Fcs)
    fprintf('\n\nPass Band cut off should be greater than Stop Band cut off\n');
    fprintf('Please re-enter');
    pause(5)

    elseif(Fcp >= Fs/2)
    fprintf('\n\Pass Band cut off should be less than half the Sampling Frequency\n');
    fprintf('Please re-enter');
    pause(5)

    else
            er = 0;
    end    end     end
```

- BPF Data Validation

```
function [Fcp1,Fcs1,Fcp2,Fcs2,Fs] = bpfvalidate()

er = 1;
while(er == 1)
clc;
[Fcp1,Fcs1,Fcp2,Fcs2,Fs] = fselectbpf();
  if(Fcp1 < Fcs1 || Fcp2 > Fcs2)
  fprintf('\n\nTransition width must be positive.\n');
  fprintf('Please re-enter');
  pause(5)

  elseif(Fcp1 >= Fs/2 || Fcs2 >= Fs/2)
  fprintf('\n\n Cut off should be less than half the Sampling Frequency\n');
  fprintf('Please re-enter');
  pause(5)

  else
   er = 0;
  end     end      end
```

- BSF Data Validation

```
function [Fcp1,Fcs1,Fcp2,Fcs2,Fs] = bsfvalidate()

er = 1;
while(er == 1)
clc;
Fcp1,Fcs1,Fcp2,Fcs2,Fs] = fselectbsf();
  if(Fcp1 > Fcs1 || Fcp2 < Fcs2)
  fprintf('\n\nTransition width must be positive.\n');
  fprintf('Please re-enter');
  pause(5)

  elseif(Fcp2 >= Fs/2 || Fcs1 >= Fs/2)
  fprintf('\n\n Cut off should be less than half the Sampling Frequency\n');
  fprintf('Please re-enter');
  pause(5)

  else
  er = 0;

end    end    end
```

- LPF Data Entry

```
function [Fcp,Fcs,Fs] = fselectlpf()

fprintf('Please enter all frequency values in Hz\n\n');
Fcp = input('Please provide a desired pass band cut off frequency: ');
t = input('Please provide a desired downward transition width in frequency: ');
Fcs = Fcp + t;
Fs = input('Please provide a desired sampling frequency: ');

end
```

- HPF Data Entry

```
function [Fcp,Fcs,Fs] = fselecthpf()

fprintf('Please enter all frequency values in Hz\n\n');
Fcp = input('Please provide a desired pass band cut off frequency: ');
t = input('Please provide a desired upward transition width in frequency: ');
Fcs = Fcp - t;
Fs = input('Please provide a desired sampling frequency: ');

end
```

- BPF Data Entry

```
function [Fcp1,Fcs1,Fcp2,Fcs2,Fs] = fselectbpf()

fprintf('Please enter all frequency values in Hz\n\n');
Fcp1 = input('Please provide a desired lower pass band cut off frequency: ');
t1 = input('Please provide a desired upward transition width in frequency: ');
Fcs1 = Fcp1 - t1;
Fcp2 = input('Please provide a desired upper pass band cut off frequency: ');
t2 = input('Please provide a desired downward transition width in frequency: ');
Fcs2 = Fcp2 + t2;
Fs = input('Please provide a desired sampling frequency: ');

end
```

- BSF Data Entry

```
function [Fcp1,Fcs1,Fcp2,Fcs2,Fs] = fselectbsf()

fprintf('Please enter all frequency values in Hz\n\n');
Fcp1 = input('Please provide a desired lower pass band cut off frequency: ');
t1 = input('Please provide a desired downward transition width in frequency: ');
Fcs1 = Fcp1 + t1;
Fcp2 = input('Please provide a desired upper pass band cut off frequency: ');
t2 = input('Please provide a desired upward transition width in frequency: ');
Fcs2 = Fcp2 - t2;
Fs = input('Please provide a desired sampling frequency: ');

end
```

- LPF using Rectangular window

```
function [mag,phase,N] = lpfrect(Fcp,Fcs,Fs)

[w,N] = myrect(Fcp,Fcs,2*Fs);
hd = lpf(N,Fcp,Fcs,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- LPF using Hamming window

```
function [mag,phase,N] = lpfhamming(Fcp,Fcs,Fs)

[w,N] = myhamming(Fcp,Fcs,2*Fs);
hd = lpf(N,Fcp,Fcs,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- LPF using Kaiser window

```
function [mag,phase,N] = lpfkaiser(Fcp,Fcs,Fs)

[w,N] = mykaiser(Fcp,Fcs,2*Fs);
hd = lpf(N,Fcp,Fcs,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- HPF using Rectangular window

```
function [mag,phase,N] = hpfrect(Fcp,Fcs,Fs)

[w,N] = myrect(Fcp,Fcs,2*Fs);
hd = hpf(N,Fcp,Fcs,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- HPF using Hamming window

```
function [mag,phase,N] = hpfhamming(Fcp,Fcs,Fs)

[w,N] = myhamming(Fcp,Fcs,2*Fs);
hd = hpf(N,Fcp,Fcs,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- HPF using Kaiser window

```
function [mag,phase,N] = hpfkaiser(Fcp,Fcs,Fs)

[w,N] = mykaiser(Fcp,Fcs,2*Fs);
hd = hpf(N,Fcp,Fcs,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- BPF using Rectangular Window

```
function [mag,phase,N] = bpfrect(Fcp1,Fcs1,Fcp2,Fcs2,Fs)

[wa,Na] = myrect(Fcp1,Fcs1,2*Fs);
[wb,Nb] = myrect(Fcp2,Fcs2,2*Fs);

if(Na > Nb)
    w = wa;
    N = Na;
else
    w = wb;
    N = Nb;
end

hd = bpf(N,Fcp1,Fcs1,Fcp2,Fcs2,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- BPF using Hamming window

```
function [mag,phase,N] = bpfhamming(Fcp1,Fcs1,Fcp2,Fcs2,Fs)

[wa,Na] = myhamming(Fcp1,Fcs1,2*Fs);
[wb,Nb] = myhamming(Fcp2,Fcs2,2*Fs);

if(Na > Nb)
    w = wa;
    N = Na;
else
    w = wb;
    N = Nb;
end

hd = bpf(N,Fcp1,Fcs1,Fcp2,Fcs2,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- BPF using Kaiser window

```
function [mag,phase,N] = bpfkaiser(Fcp1,Fcs1,Fcp2,Fcs2,Fs)

[wa,Na] = mykaiser(Fcp1,Fcs1,2*Fs);
[wb,Nb] = mykaiser(Fcp2,Fcs2,2*Fs);

if(Na > Nb)
    w = wa;
    N = Na;
else
    w = wb;
    N = Nb;
end

hd = bpf(N,Fcp1,Fcs1,Fcp2,Fcs2,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- BSF using Rectangular window

```
function [mag,phase,N] = bsfrect(Fcp1,Fcs1,Fcp2,Fcs2,Fs)

[wa,Na] = myrect(Fcp1,Fcs1,2*Fs);
[wb,Nb] = myrect(Fcp2,Fcs2,2*Fs);

if(Na > Nb)
    w = wa;
    N = Na;
else
    w = wb;
    N = Nb;
end

hd = bsf(N,Fcp1,Fcs1,Fcp2,Fcs2,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- BSF using Hamming window

```
function [mag,phase,N] = bsfhamming(Fcp1,Fcs1,Fcp2,Fcs2,Fs)

[wa,Na] = myhamming(Fcp1,Fcs1,2*Fs);
[wb,Nb] = myhamming(Fcp2,Fcs2,2*Fs);

if(Na > Nb)
    w = wa;
    N = Na;
else
    w = wb;
    N = Nb;
end
```

```
hd = bsf(N,Fcp1,Fcs1,Fcp2,Fcs2,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- BSF using Kaiser window

```
function [mag,phase,N] = bsfkaiser(Fcp1,Fcs1,Fcp2,Fcs2,Fs)

[wa,Na] = mykaiser(Fcp1,Fcs1,2*Fs);
[wb,Nb] = mykaiser(Fcp2,Fcs2,2*Fs);

if(Na > Nb)
    w = wa;
    N = Na;
else
    w = wb;
    N = Nb;
end

hd = bsf(N,Fcp1,Fcs1,Fcp2,Fcs2,2*Fs);
h = hd.*w';
[f,phase] = freqz(h);
mag = abs(f);

end
```

- Rectangular Window

```
function [w,N] = myrect(Fcp,Fcs,Fs)

fcp = Fcp/Fs;
fcs = Fcs/Fs;
f = abs(fcp - fcs);
N = round(0.9/f);
if( mod(N,2) ~= 0 )
    N = N+1;
end
w = rectwin(N+1);

end
```

- Hamming window

```
function [w,N] = myhamming(Fcp,Fcs,Fs)

fcp = Fcp/Fs;
fcs = Fcs/Fs;
f = abs(fcp - fcs);
N = round(3.3/f);
if( mod(N,2) ~= 0 )
    N = N+1;
end
w = hamming(N+1);

end
```

- Kaiser Window

```
function [w,N] = mykaiser(Fcp,Fcs,Fs)

fcp = Fcp/Fs;
fcs = Fcs/Fs;
f = abs(fcp - fcs);
N = round(2.507/f);
if( mod(N,2) ~= 0 )
    N = N+1;
end
w = kaiser(N+1,5.1822);

end
```

- Low Pass filter

```
function [h] = lpf(N,Fcp,Fcs,Fs)

fcp = Fcp/Fs;
fcs = Fcs/Fs;
fc = (fcp + fcs)/2;

for i = 1 : N+1
    h(i) = sin( 2 * pi * fc * ( (i) - N/2) )/(((i) - N/2) * pi );
end

h(N/2) = 2*fc;
end
```

- High Pass filter

```
function [h] = hpf(N,Fcp,Fcs,Fs)

fcp = Fcp/Fs;
fcs = Fcs/Fs;
fc = (fcp + fcs)/2;

for i = 1 : N+1
    h(i) = -sin( 2 * pi * fc * ( (i) - N/2) )/(((i) - N/2) * pi );
end

h(N/2) = 1 - 2*fc;
end
```

- Band Pass filter

```
function [h] = bpf(N,Fcp1,Fcs1,Fcp2,Fcs2,Fs)

fcp1 = Fcp1/Fs;
fcs1 = Fcs1/Fs;
fc1 = (fcp1 + fcs1)/2;

fcp2 = Fcp2/Fs;
fcs2 = Fcs2/Fs;
fc2 = (fcp2 + fcs2)/2;

hp = hpf(N,Fcp1,Fcs1,Fs);
hs = lpf(N,Fcp2,Fcs2,Fs);
```

```
h = hp+hs;

h(N/2) = 2 * (fc2 - fc1);

end
```

- Band Stop filter

```
function[h] = bsf(N,Fcp1,Fcs1,Fcp2,Fcs2,Fs)

fcp1 = Fcp1/Fs;
fcs1 = Fcs1/Fs;
fc1 = (fcp1 + fcs1)/2;

fcp2 = Fcp2/Fs;
fcs2 = Fcs2/Fs;
fc2 = (fcp2 + fcs2)/2;

hp = lpf(N,Fcp1,Fcs1,Fs);
hs = hpf(N,Fcp2,Fcs2,Fs);

h = hp+hs;

h(N/2) = 1- 2 * (fc2 - fc1);

end
```