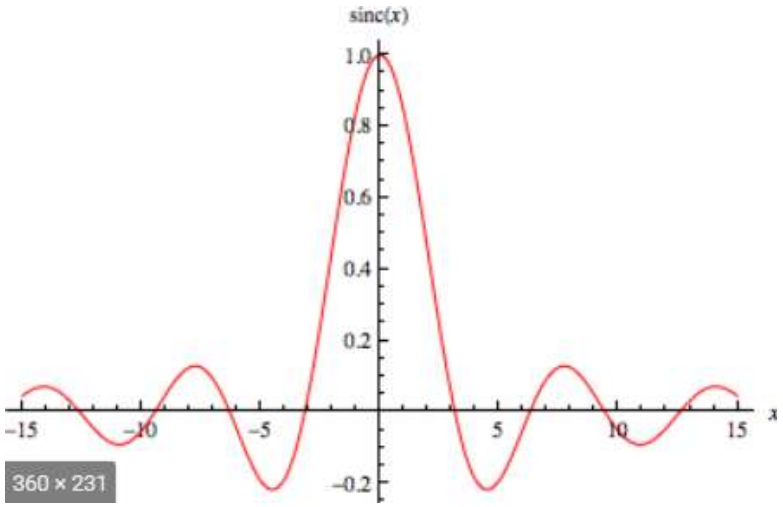
Intro

My project is a windowed sinc filter generator. This project is mean to be similar to filter designer tool found in Matlab. It can create low pass, high pass, band pass, and band stop windowed sinc filters. It has a simple graphical user interface that allows you to input parameters. You have the option to export the filter coefficients as a csv file or as a c header file. The tool will also plot the filter kernel.

Theory

The sinc function, also known as the sampling function or cardinal sine, is a function that is often used in signal processing. The plot and formula are shown below.



$$\operatorname{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$$

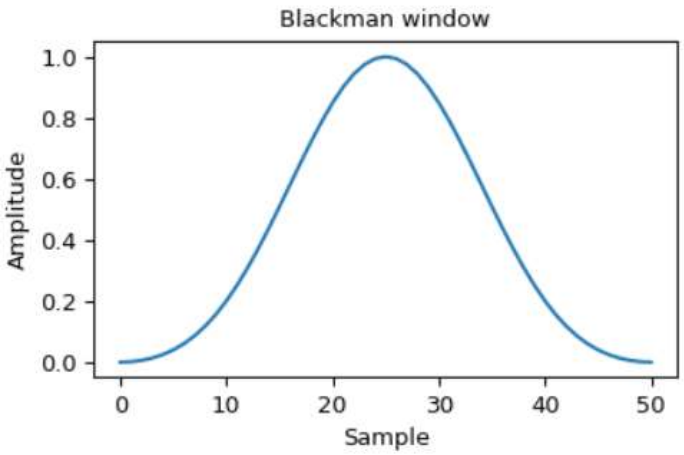
It can be implemented in Matlab as follows.

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

In the Matlab implementation There are three input parameters. They are fc, n, and N. The first parameter fc is the cutoff frequency. It is used to determine what band of frequencies are passed and rejected. The second parameter n is an array going from 1 to the third parameter N. There is a fourth parameter called b that is not used in this equation. This is the transition

band. This value determines how often the filter transitions from passing and rejecting values. The lower this value is, the higher quality the filter will be. This parameter is used to calculate N. N is simply calculated as ceil(4/b). While a sinc filter works in theory, it is unrealizable in the real world. This is because it is an infinite filter. You could truncate values off of the filter, but this would result in excessive ripple. The solution is to use a window. A window function is just a function that is 0 at a certain interval. For this project I chose to use a Blackman window. The formula and plot are shown below.

$$w[n] = 0.42 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) + 0.08\cos\left(\frac{4\pi n}{N-1}\right)$$



The sinc function and Blackman window are then multiplied together to create a windowed sinc filter low pass filter. To create a high pass filter, you perform spectral inversion. This is simply where you invert the filter. Multiply every element in the filter by negative one and then add one to the middle element. To create a band pass filter you create a low pass filter using one cutoff frequency and a high pass filter using the other cutoff frequency. You then convolve these two filters together. To create a band stop filter, you do the exact same thing except instead of convolving the two filters, you add them together.

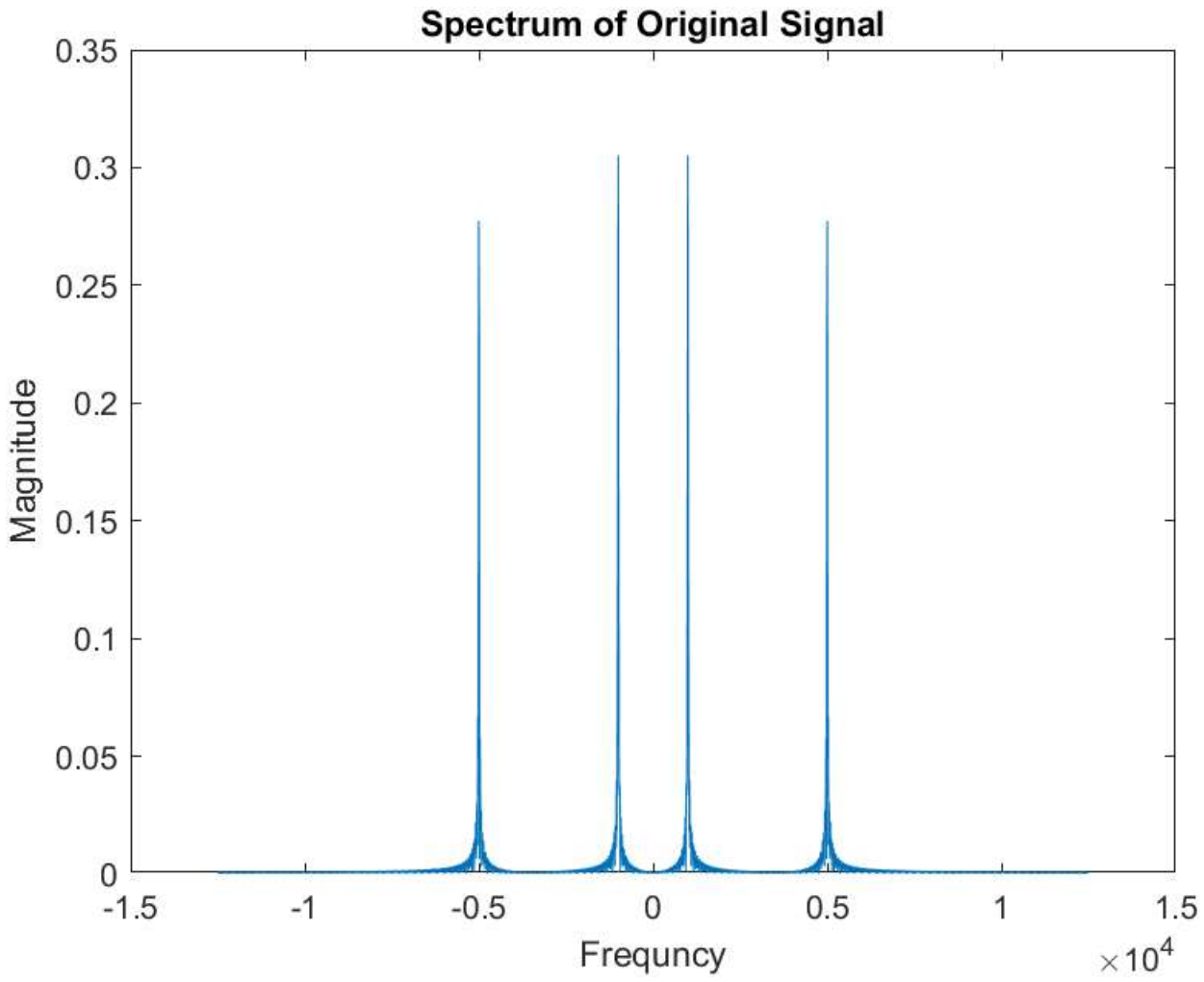
Simulation

To simulate the project, I used a simple test script. It created a signal with the two frequencies 1000 Hz and 5000 Hz. It simply reads in a csv file to variable, and convolves the signal with the filter read from the csv file. It then time and frequency domains of the result. The initial signal that is generated is shown below.

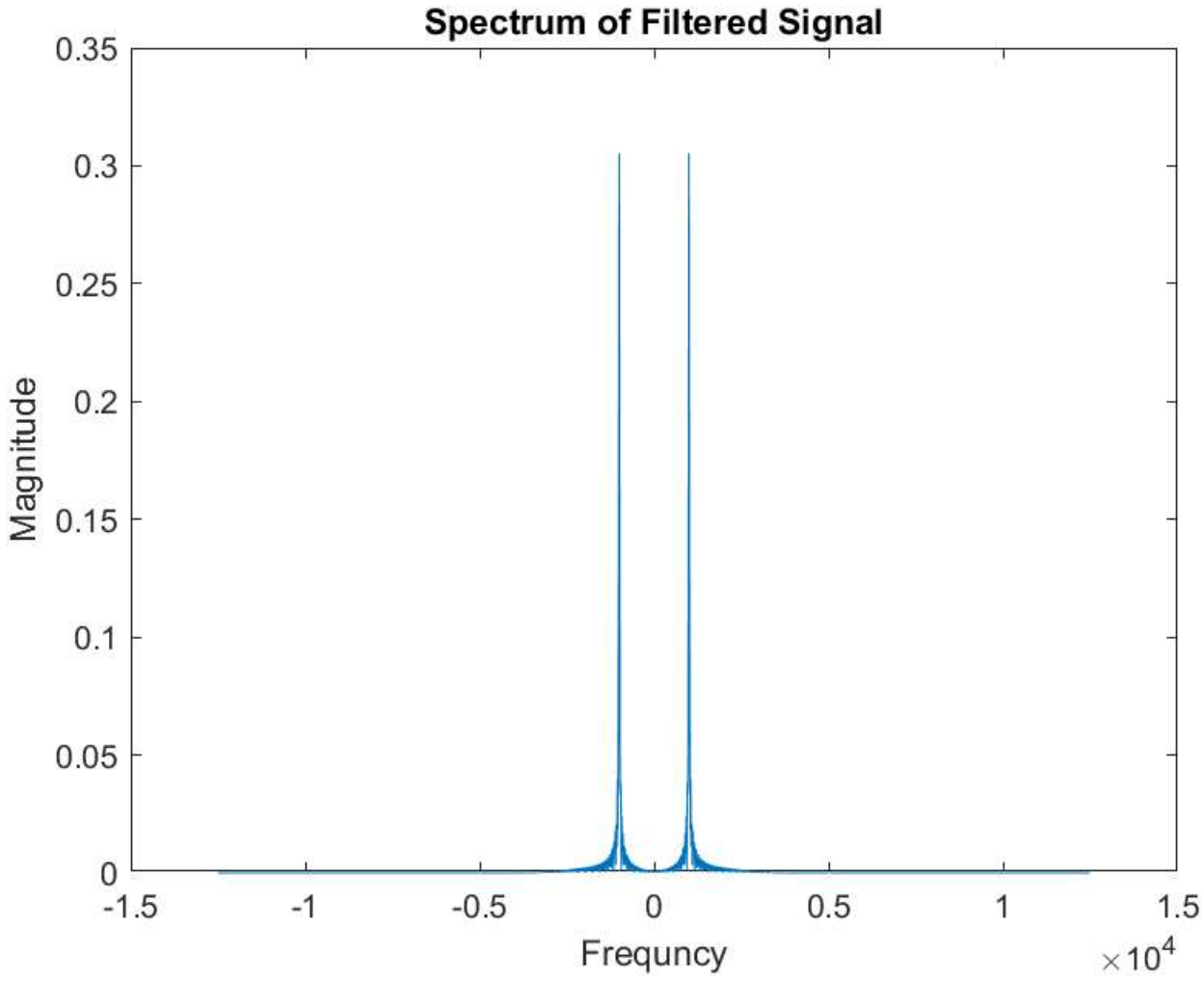
```
fs = 25000;
t = 0:1/fs:0.05;
f1 = 1000;
f2 = 5000;

x = cos(2*pi*f1*t) + cos(2*pi*f2*t);
```

Result

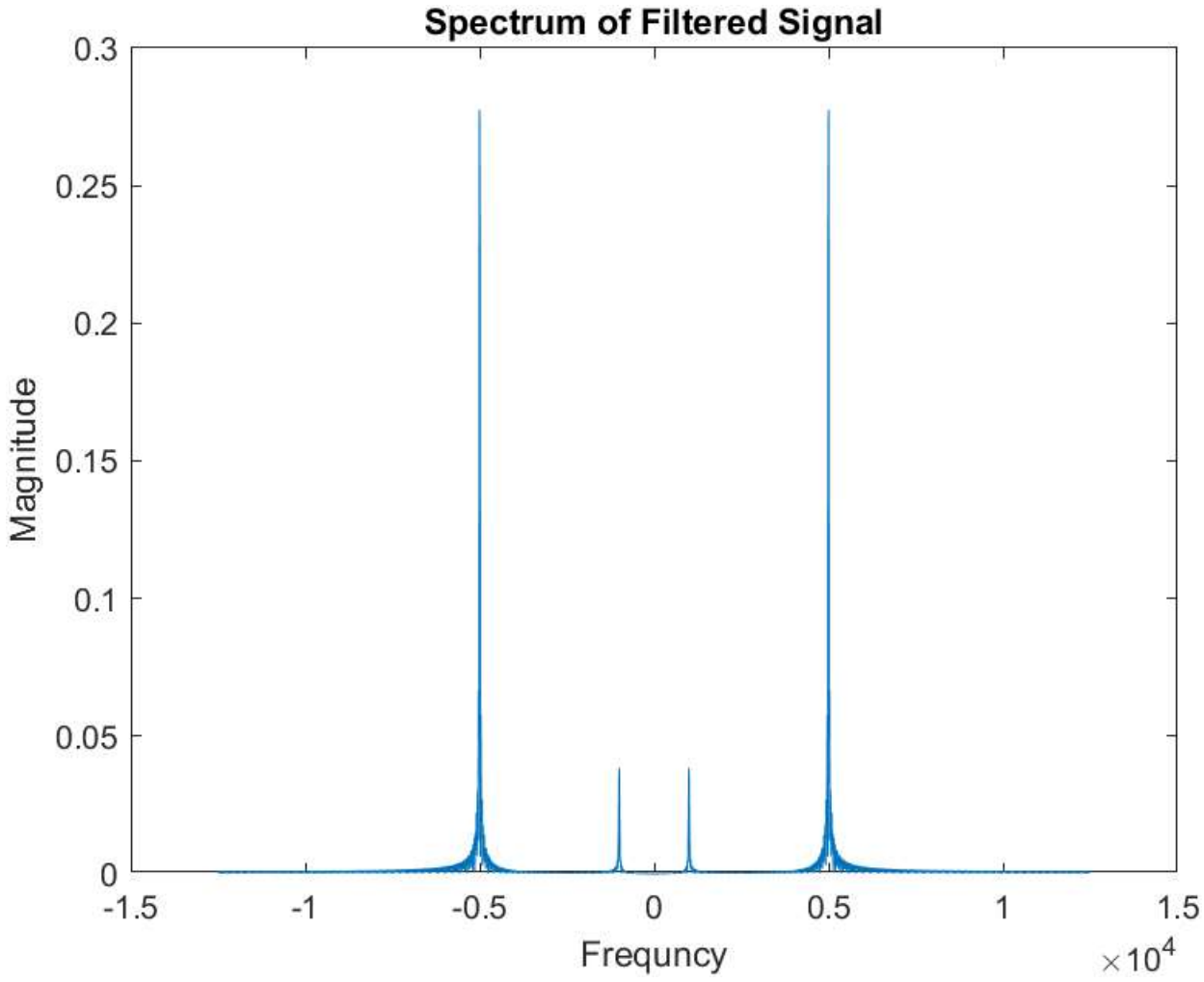
Below is the spectrum of the original plot.



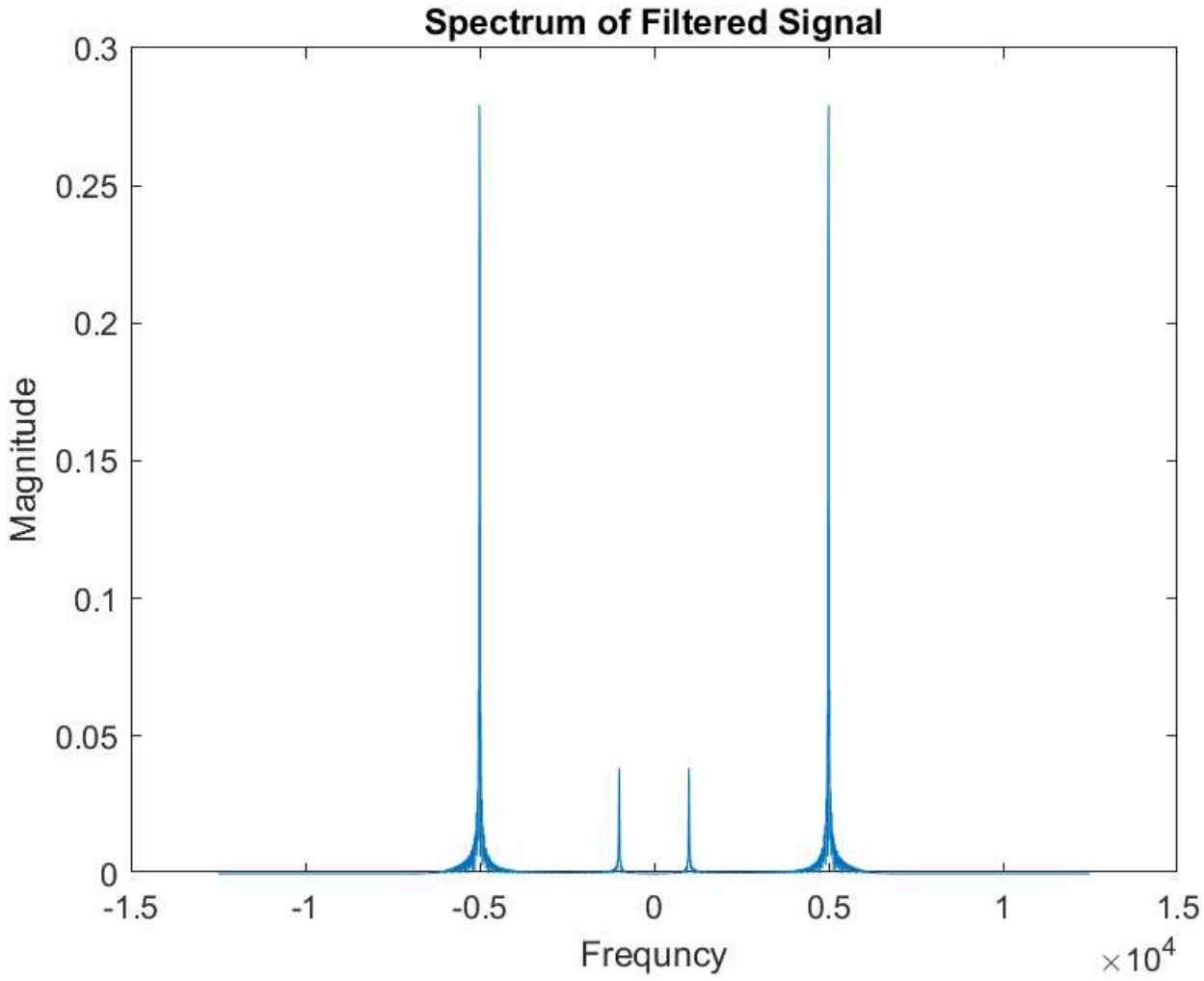
Below is the result of filtering the input signal with a low pass filter. The cutoff is 3000 and the transition, band of 0.08.



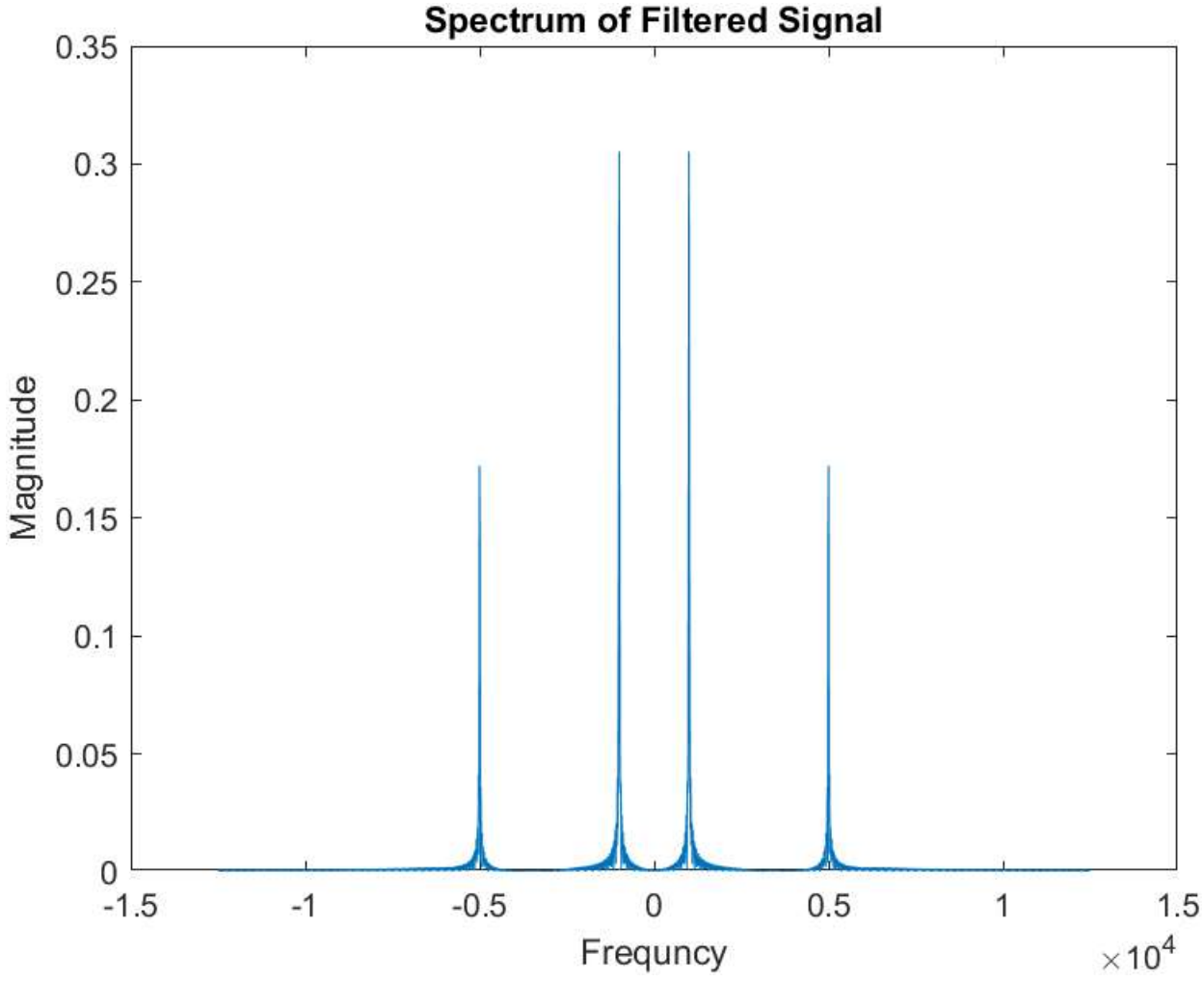
Below is the result of filtering the input signal with a high pass filter. The cutoff is 3000 and, the transition band of 0.08.



Below is the result of filtering the input signal with a band pass filter. The cutoffs of 3000 and 6000 and, the transition band of 0.08.



Below is the result of filtering the input signal with a band stop filter. The cutoffs of 3000 and 6000 and, the transition band of 0.08.



Appendix

Test Script

```
close all; clear;

fs = 25000;
t = 0:1/fs:0.05;
f1 = 1000;
f2 = 5000;

x = cos(2*pi*f1*t) + cos(2*pi*f2*t);

figure
spectrum_plot(x, fs)
title('Spectrum of Original Signal')
ylabel('Magnitude'), xlabel('Frequncy')

h = csvread('filter.csv');
y = convolute(x, h);

figure
spectrum_plot(y, fs)
title('Spectrum of Filtered Signal')
ylabel('Magnitude'), xlabel('Frequncy')
```

Spectrum Plot Function

```
function spectrum_plot(y, fs)
    l = length(y);
    n = pow2(nextpow2(l));
    y_dft = fft(y, n);
    y_s = fftshift(y_dft);
    f = (-n/2:n/2-1)*(fs/n);
    plot(f, abs(y_s)/n);
end
```

Generate Low pass Function

```
function h = generate_lowpass(fs, fc, b)
    fc = fc/fs;
    N = ceil(4/b);
    n = 1:N;
    h = sinc(2*fc*(n-(N-1)/2));
    w = 0.42-0.5*cos(2*pi*n/(N-1)) + 0.08*cos(4*pi*n(N-1));
    h = h.*w;

    h = h/sum(h);
end
```

Generate High pass Function

```
function h = generate_bandpass(fs, fl, fh, b)
    fl = fl/fs;
    fh = fh/fs;

    N = ceil(4/b);
    n = 1:N;

    hlpf = sinc(2*fh*(n-(N-1)/2));
    w = 0.42-0.5*cos(2*pi*n/(N-1)) + 0.08*cos(4*pi*n(N-1));
    hlpf = hlpf.*w;
    hlpf = hlpf/sum(hlpf);

    hhpf = sinc(2*fl*(n-(N-1)/2));
    hhpf = hhpf.*w;
    hhpf = hhpf/sum(hhpf);
    hhpf = hhpf*-1;
    hhpf(ceil(end/2)) = hhpf(ceil(end/2))+1;

    h = convolute(hlpf, hhpf);
end
```

Generate Band pass Function

```
function h = generate_bandpass(fs, fl, fh, b)
    fl = fl/fs;
    fh = fh/fs;

    N = ceil(4/b);
    n = 1:N;

    hlpf = sinc(2*fh*(n-(N-1)/2));
    w = 0.42-0.5*cos(2*pi*n/(N-1)) + 0.08*cos(4*pi*n(N-1));
    hlpf = hlpf.*w;
    hlpf = hlpf/sum(hlpf);

    hhpf = sinc(2*fl*(n-(N-1)/2));
    hhpf = hhpf.*w;
    hhpf = hhpf/sum(hhpf);
    hhpf = hhpf*-1;
    hhpf(ceil(end/2)) = hhpf(ceil(end/2))+1;

    h = convolute(hlpf, hhpf);
end
```

Generate Band Stop Function

```
function h = generate_bandstop(fs, fl, fh, b)
    fl = fl/fs;
    fh = fh/fs;

    N = ceil(4/b);
    n = 1:N;

    hlpf = sinc(2*fl*(n-(N-1)/2));
    w = 0.42-0.5*cos(2*pi*n/(N-1)) + 0.08*cos(4*pi*n(N-1));
    hlpf = hlpf.*w;
    hlpf = hlpf/sum(hlpf);

    hhpf = sinc(2*fh*(n-(N-1)/2));
    hhpf = hhpf.*w;
    hhpf = hhpf/sum(hhpf);
    hhpf = hhpf*-1;
    hhpf(ceil(end/2)) = hhpf(ceil(end/2))+1;

    h = hhpf + hlpf;
end
```

Generate Band Stop Function

```
function generate_c_header(h)
    fid = fopen('filter.h','wt');
    fprintf(fid, 'double filter[');
    fprintf(fid, '%d', length(h));
    fprintf(fid, '] = {\n');

    for i = 1:length(h)
        fprintf(fid, '%.25f', h(i));
        if(i ~= length(h))
            fprintf(fid, ', ');
        end
        if(mod(i, 4) == 0)
            fprintf(fid, '\n');
        end
    end
    fprintf(fid, '};');
    fclose(fid);
end
```