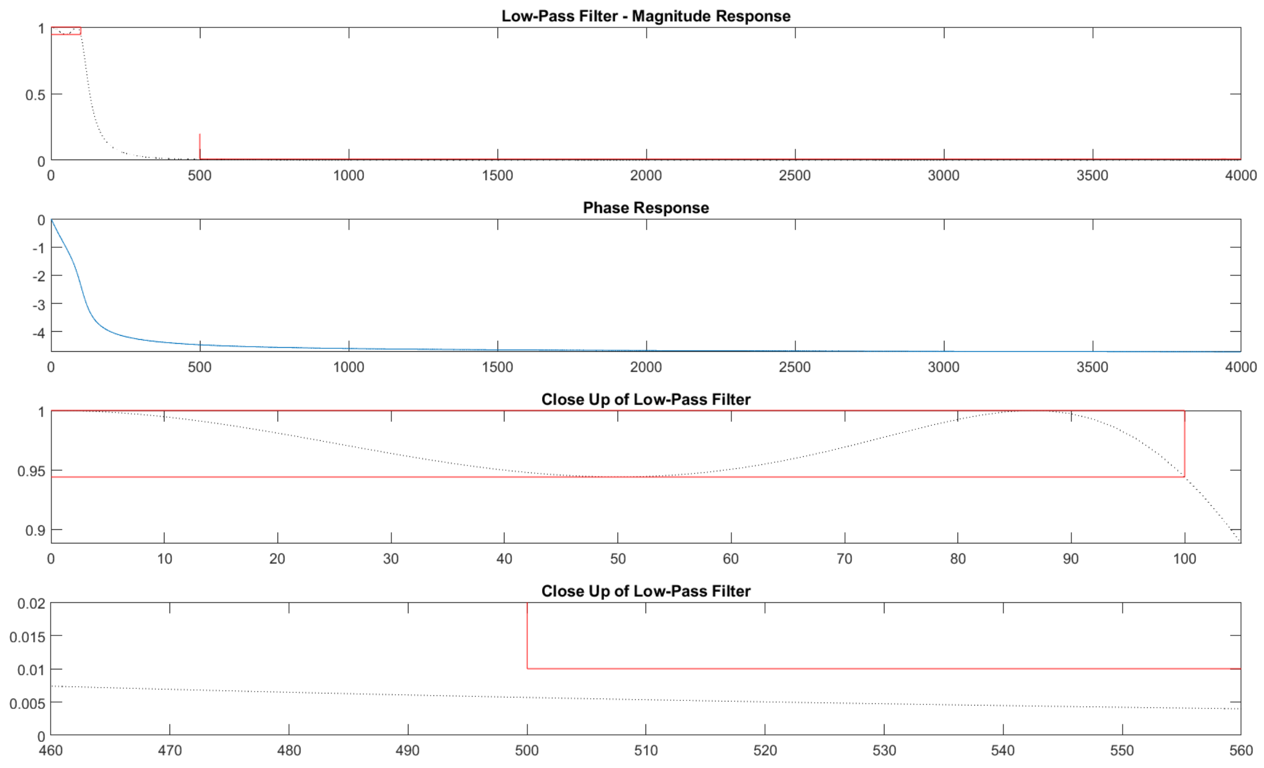
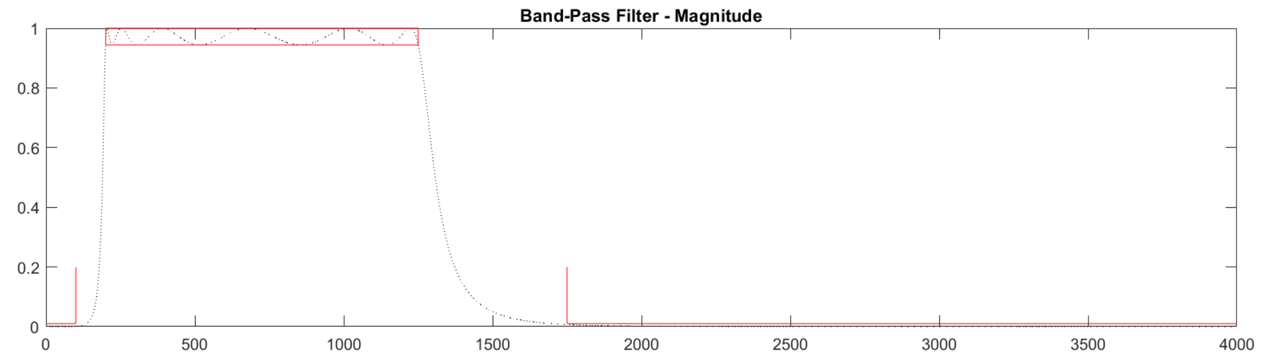
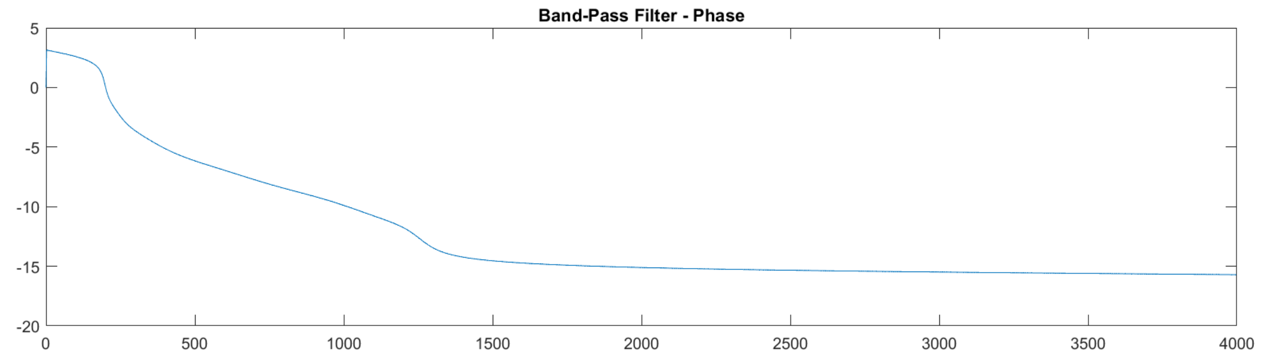
Matlab code for generating coefficients for 3 filters and writing them out to a file.

```
% The low-pass filter is built first.

% Using the frequencies we determine the order of Butterworth needed.
[nbwlp,wbwlp] = buttord( 0.1/4, 0.5/4, 0.5, 40 )

%Note that the cutoff is moved from 0.1/4 ( = 0.025 ), which would be the 3 dB point, out to
0.04.  If the 3 dB point is at 0.04, the gain at 0.025 is 0.5 dB, matching out
specifications.

%Using the frequencies we determine the order and gain of Chebyshev needed.
[nchbylp,wchbylp] = cheb1ord( 0.1/4, 0.5/4, 0.5, 40 )

% Choosing Chebyshev

% and create the poles, zeros and gain for the Chebyshev.
[zLp,pLp,kLp] = cheby1(nchbylp, 0.5, wchbylp );
[ChebyLpSos,gLp] = zp2sos(zLp,pLp,kLp);

% Generate impulse response of system.
d = zeros( 2048, 1 );
d(1) = 1.0;

hlp = sosfilt( ChebyLpSos, d );

% Produce a plot of the frequency response.
[Hlp,w] = freqz( ChebyLpSos, 2048 );
Hlp = gLp*Hlp;

figure(1);
subplot(411),plot(4000*w/pi,abs(Hlp),'k:', ...
                  [0 100 100 0], [1 1 0.944 0.944], 'r-', ... % Adds spec's
                  [4000 500 500], [ 0.01 0.01 0.2 ], 'r-');   % to plot
title('Low-Pass Filter - Magnitude Response');
subplot(412),plot(4000*w/pi,unwrap(angle(Hlp)));
title('Phase Response' );
subplot(413),plot(4000*w/pi,abs(Hlp),'k:', ...
                  [0 100 100 0], [1 1 0.944 0.944], 'r-', ...
                  [4000 500 500], [ 0.01 0.01 0.2 ], 'r-' );
xlim( [ 0 105 ] );
title( 'Close Up of Low-Pass Filter' );
subplot(414),plot(4000*w/pi,abs(Hlp),'k:', ...
                  [0 100 100 0], [1 1 0.944 0.944], 'r-', ...
                  [4000 500 500], [ 0.01 0.01 0.02 ], 'r-' );
xlim( [ 460 560] );
title( 'Close Up of Low-Pass Filter' );
```



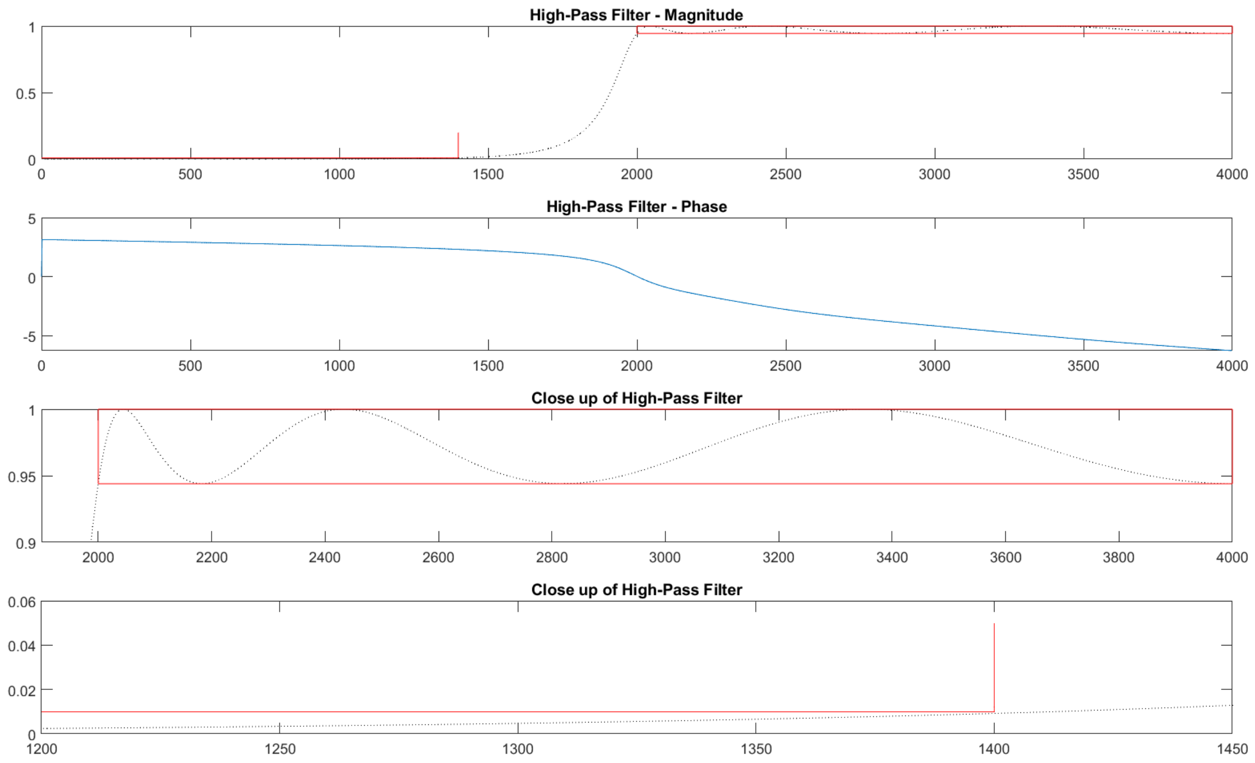
```
% The band-pass filter is next.
[ nbwbp, wbwbp ]     = buttord( [0.2/4 1.25/4], [0.1/4 1.75/4], 0.5, 40 )

% Test the Chebyshev.
[ nchbybp, wchbybp ] = cheb1ord( [0.2/4 1.25/4], [0.1/4 1.75/4], 0.5, 40 )

% Choosing Chebyshev

[zBp,pBp,kBp] = cheby1( nchbybp, 0.5, wchbybp );
[ChebyBpSos,gBp] = zp2sos(zBp,pBp,kBp);

hbp = sosfilt( ChebyBpSos, d );
% Produce a plot of the frequency response.
Hbp = gBp*freqz( ChebyBpSos, 2048 );
Hlp = gBp*Hbp;

figure(2);
subplot( 211 ),plot(4000*w/pi,abs(Hbp),'k:', ...
                [1250 200 200 1250 1250], [1 1 0.944 0.944 1], 'r-', ...
                [4000 1750 1750], [ 0.01 0.01 0.2 ], 'r-', ...
                [0 100 100], [ 0.01 0.01 0.2 ], 'r-' );
title('Band-Pass Filter - Magnitude');
subplot( 212 ),plot(4000*w/pi,unwrap(angle(Hbp)));
title('Band-Pass Filter - Phase');
figure(3);
subplot( 311 ),plot(4000*w/pi,abs(Hbp),'k:', ...
                [1250 200 200 1250 1250], [1 1 0.944 0.944 1], 'r-', ...
                [4000 1750 1750], [ 0.01 0.01 0.05 ], 'r-', ...
                [0 100 100], [ 0.01 0.01 0.05 ], 'r-' );
xlim( [ 0 150] );
title( 'Close Up of Lower Stop Band' );
subplot( 312 ),plot(4000*w/pi,abs(Hbp),'k:', ...
                [1250 200 200 1250 1250], [1 1 0.944 0.944 1], 'r-', ...
                [4000 1750 1750], [ 0.01 0.01 0.05 ], 'r-', ...
                [0 100 100], [ 0.01 0.01 0.05 ], 'r-' );
xlim( [ 175 1260 ] );
ylim( [ 0.9 1 ] );
title( 'Close Up of Pass Band' );
subplot( 313 ),plot(4000*w/pi,abs(Hbp),'k:', ...
                [1250 200 200 1250 1250], [1 1 0.944 0.944 1], 'r-', ...
                [4000 1750 1750], [ 0.01 0.01 0.05 ], 'r-', ...
                [0 100 100], [ 0.01 0.01 0.05 ], 'r-' );
xlim( [ 1700 2000] );
title( 'Close Up of Upper Stop Band' );
```



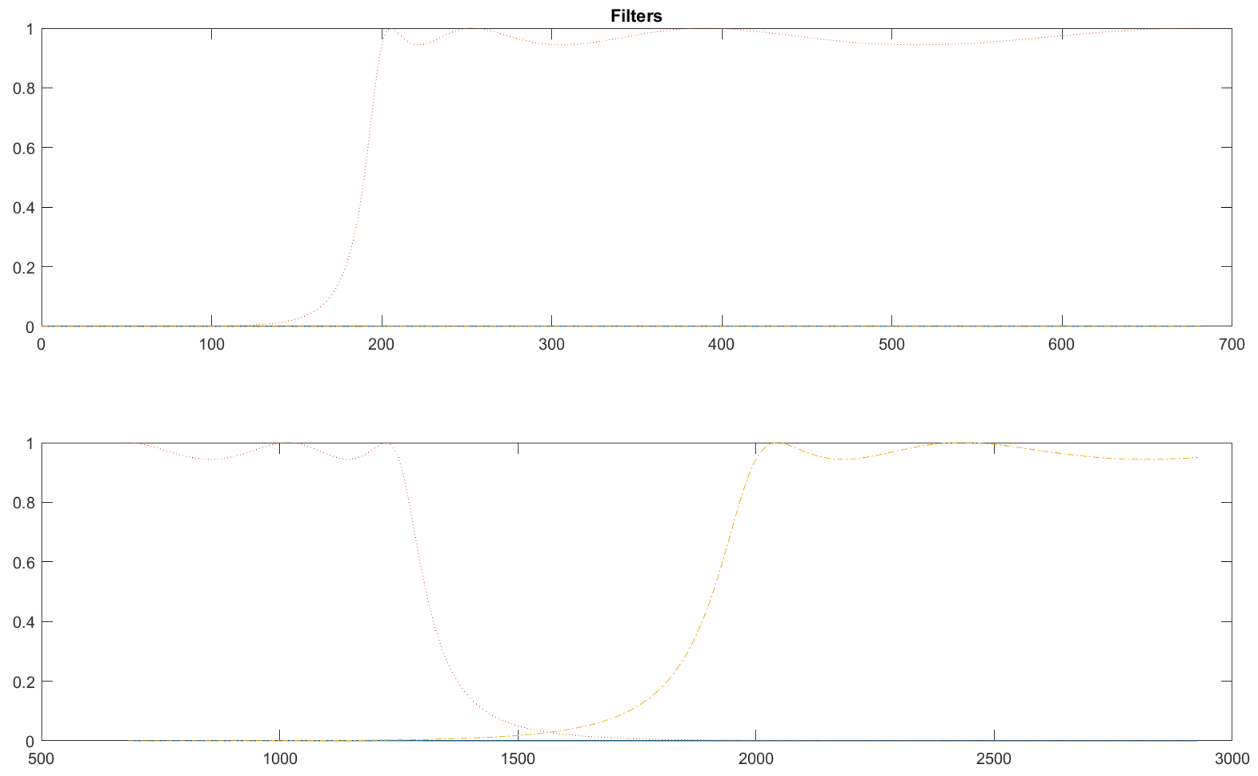


```
% The high-pass filter is last.
[nbwhp,wbwhp] = buttord( 2.0/4, 1.4/4, 0.5, 40 )

[nchbyhp,wchbyhp] = cheb1ord( 2.0/4, 1.4/4, 0.5, 40 )

% Choosing Chebyshev

[ zHp, pHp, kHp ] = cheby1( nchbyhp, 0.5, wchbyhp, 'high' );
[ChebyHpSos,gHp] = zp2sos( zHp,pHp,kHp );

hhp = sosfilt(ChebyHpSos, d );

Hhp = freqz( ChebyHpSos, 2048 );
Hhp = gHp*Hhp;

% Produce a plot of the frequency response.
figure(4);
subplot( 411 ), plot(4000*w/pi,abs(Hhp),'k:', ...
        [4000 2000 2000 4000 4000], [ 1 1 0.944 0.944 1 ], 'r-', ...
                [0 1400 1400], [ 0.01 0.01 0.2 ], 'r-');
title('High-Pass Filter - Magnitude');
subplot( 412 ), plot(4000*w/pi,unwrap(angle(Hhp)));
```

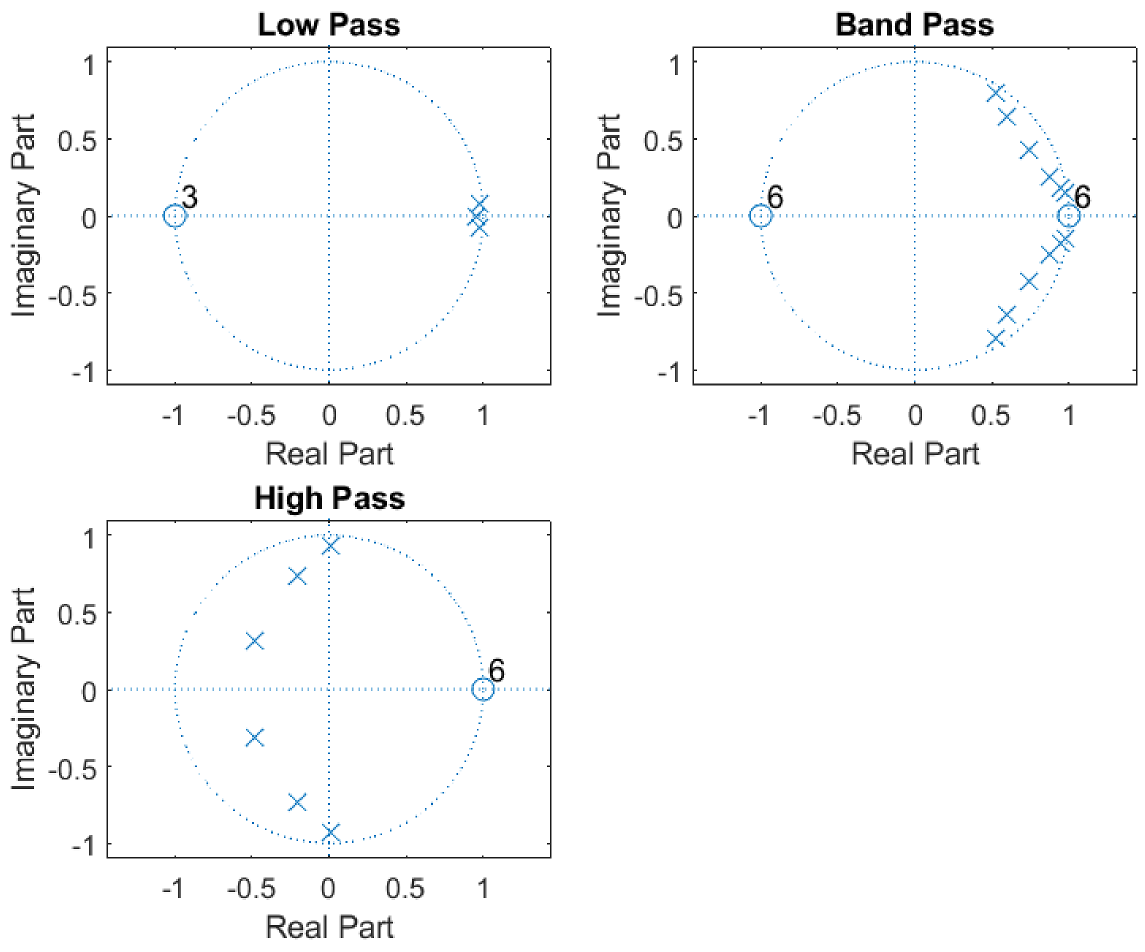
```
title('High-Pass Filter - Phase');
subplot( 413 ), plot(4000*w/pi,abs(Hhp),'k:', ...
        [4000 2000 2000 4000 4000], [ 1 1 0.944 0.944 1 ], 'r-', ...
               [0 1400 1400], [ 0.01 0.01 0.2 ], 'r-');
xlim([ 1900 4000 ] );
ylim([0.9 1.0] );
title('Close up of High-Pass Filter');
subplot( 414 ), plot(4000*w/pi,abs(Hhp),'k:', ...
        [4000 2000 2000 4000 4000], [ 1 1 0.944 0.944 1 ], 'r-', ...
               [0 1400 1400], [ 0.01 0.01 0.05 ], 'r-');
xlim([ 1200 1450 ] );
title('Close up of High-Pass Filter');
```



```
% Produce a plot of the frequency responses overlaid on one another.
figure(5);

subplot(211),plot(4000*w(1:350)/pi,abs(Hlp(1:350)),'-',...
                  4000*w(1:350)/pi,abs(Hbp(1:350)),':',...
                  4000*w(1:350)/pi,abs(Hhp(1:350)),'-.');
title('Filters');
subplot(212),plot(4000*w(350:1500)/pi,abs(Hlp(350:1500)),'-',...
                  4000*w(350:1500)/pi,abs(Hbp(350:1500)),':',...
                  4000*w(350:1500)/pi,abs(Hhp(350:1500)),'-.');
```



```
% Produce a plot of the frequency responses overlaid on one another.
figure(6);
subplot( 221),zplane( zLp, pLp );
title( 'Low Pass' );
subplot( 222),zplane( zBp, pBp );
title( 'Band Pass' );
subplot( 223),zplane( zHp, pHp );
title( 'High Pass' );
```



```
%% Write out filters.

fid = fopen( 'Filters.h', 'w' );

sos = ChebyLpSos;
[N,M] = size( sos );
sos(:,1:3) = (gLp^(1/N))*sos(:,1:3);

fprintf( fid, '// Define Low-Pass Filter\n' );
fprintf( fid, 'int LP_N = %d;\nfloat LP_b[%d] = { // Numerator\n', ...
              3*N, 3*N );
for k = 1:N-1
    fprintf( fid, '           %gF, %gF, %gF,\n', sos(k,1), sos(k,2), sos(k,3) );
end
fprintf( fid, '           %gF, %gF, %gF};\n', sos(N,1), sos(N,2), sos(N,3) );

fprintf( fid, 'float LP_a[%d] = { // Denominator\n', 3*N );
for k = 1:N-1
    fprintf( fid, '           %gF, %gF, %gF,\n', sos(k,4), sos(k,5), sos(k,6) );
end
fprintf( fid, '           %gF, %gF, %gF};\n', sos(N,4), sos(N,5), sos(N,6) );
fprintf( fid, 'float LP_delays[%d] = { 0.0F };//Delays for Low Pass',...
              2*N );

sos = ChebyBpSos;

[N,M] = size( sos );
sos(:,1:3) = (gBp^(1/N))*sos(:,1:3);

fprintf( fid, '\n\n// Define Band-Pass Filter\n' );
fprintf( fid, 'int BP_N = %d;\nfloat BP_b[%d] = { // Numerator\n', ...
               3*N, 3*N );
for k = 1:N-1
    fprintf( fid, '           %gF, %gF, %gF,\n', sos(k,1), sos(k,2), sos(k,3) );
end
fprintf( fid, '           %gF, %gF, %gF};\n', sos(N,1), sos(N,2), sos(N,3) );

fprintf( fid, 'float BP_a[%d] = { // Denominator\n', 3*N );
for k = 1:N-1
    fprintf( fid, '           %gF, %gF, %gF,\n', sos(k,4), sos(k,5), sos(k,6) );
end
fprintf( fid, '           %gF, %gF, %gF};\n', sos(N,4), sos(N,5), sos(N,6) );
fprintf( fid, 'float BP_delays[%d] = { 0.0F };//Delays for Band Pass',...
              2*N );

sos = ChebyHpSos;
[N,M] = size( sos );
sos(:,1:3) = (gHp^(1/N))*sos(:,1:3);

fprintf( fid, '\n\n// Define High-Pass Filter\n' );
fprintf( fid, 'int HP_N = %d;\nfloat HP_b[%d] = { // Numerator\n', ...
                3*N, 3*N );
for k = 1:N-1
    fprintf( fid, '           %gF, %gF, %gF,\n', sos(k,1), sos(k,2), sos(k,3) );
end
fprintf( fid, '           %gF, %gF, %gF};\n', sos(N,1), sos(N,2), sos(N,3) );

fprintf( fid, 'float HP_a[%d] = {// Denominator \n', 3*N );
for k = 1:N-1
    fprintf( fid, '           %gF, %gF, %gF,\n', sos(k,4), sos(k,5), sos(k,6) );
end
fprintf( fid, '           %gF, %gF, %gF};\n', sos(N,4), sos(N,5), sos(N,6) );
fprintf( fid, 'float HP_delays[%d] = { 0.0F };//Delays for High Pass',...
              2*N );

fclose( fid );
```

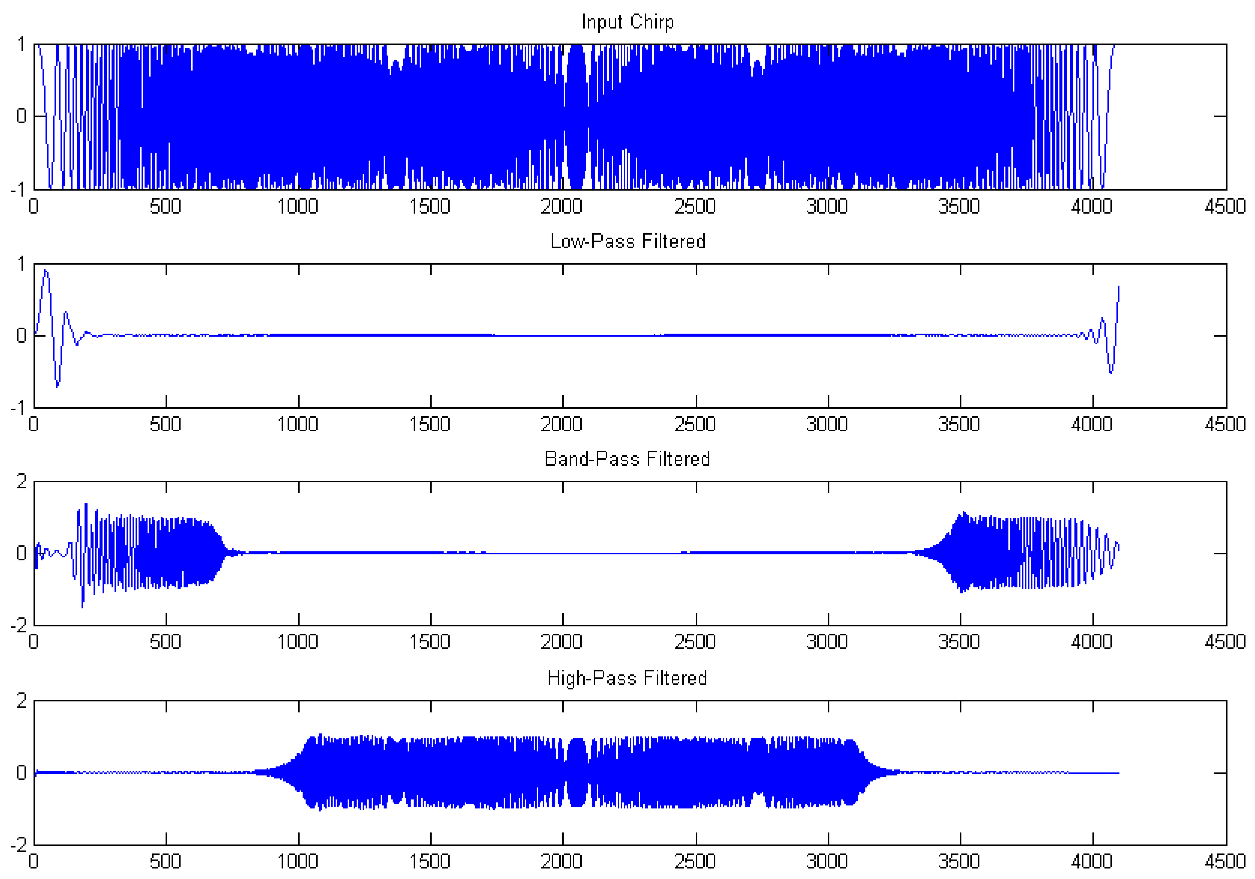
Filters.h written from matlab.

```
// Define Low-Pass Filter
int LP_N = 6;
float LP_b[6] = { // Numerator
          0.00642473F, 0.0064248F, 0.0F,
          0.00642473F, 0.0128494F, 0.00642466F};
float LP_a[6] = { // Denominator
          1.0F, -0.951955F, 0.0F,
          1.0F, -1.94516F, 0.952038F};
float LP_Delays[4] = { 0.0F };//Delays for Low Pass

// Define Band-Pass Filter
int BP_N = 18;
float BP_b[18] = { // Numerator
          0.258373F, 0.516747F, 0.258373F,
          0.258373F, -0.516747F, 0.258372F,
          0.258373F, 0.51732F, 0.258948F,
          0.258373F, 0.516172F, 0.2578F,
          0.258373F, -0.51732F, 0.258948F,
          0.258373F, -0.516173F, 0.257801F};
float BP_a[18] = { // Denominator
          1.0F, -1.47463F, 0.725089F,
          1.0F, -1.19651F, 0.765093F,
          1.0F, -1.75324F, 0.834348F,
          1.0F, -1.05038F, 0.908539F,
          1.0F, -1.89809F, 0.933409F,
          1.0F, -1.95829F, 0.982333F};
float BP_Delays[12] = { 0.0F };//Delays for Band Pass

// Define High-Pass Filter
int HP_N = 9;
float HP_b[9] = { // Numerator
          0.227282F, -0.454571F, 0.227286F,
          0.227282F, -0.455448F, 0.228169F,
          0.227282F, -0.453673F, 0.226394F};
float HP_a[9] = {// Denominator
          1.0F, 0.970874F, 0.332497F,
          1.0F, 0.40708F, 0.578724F,
          1.0F, -0.0211381F, 0.857413F};
float HP_Delays[6] = { 0.0F };//Delays for High Pass
```

Main test program that will apply these filters to a chirp signal.

```
// ExampleCode.cpp : Defines the entry point for the console application.
//
#include "stdafx.h" // Silly header created by MS Studio
#include <math.h>
#include <stdio.h>
#include "filters.h"  // Defines the filters to be run

// Function that implements a second order difference equation
// using a transposed format.
float SecondOrderfilter( float in, float *b, float *a, float *delays )
{
 // Compute output
 float out = *(b++) * in + delays[0];
 // Service delay line.
 delays[0] = *(b++) * in - *(a++) * out + delays[1];
 delays[1] = *b * in - *a * out;
 // return results.
 return out;
} // End of SecondOrderfilter
```

```c
// Test program that generates a 4096 point chirp and filters it with
// the three filters.
int main(int argc, char* argv[])
{

    FILE *fout;
    float in,outLP, outBP, outHP;
    float scale = 4.0F*(float) atan(1.0) / 4096.0F;  // equals ( pi / 4096 )
    int n, k,m;

    // Open output file and test for valid.
    fout = fopen( "Output.Txt", "w" );
    if( fout )
    {
        // Loop through time index of 0 to 4095
        for( n = 0; n < 4096; n++ )
        {
            // Chirp at index n
            in = (float) cos( scale*n*n );

            // Loop for implementing Low-Pass Filter.
            outLP = in;
            m = 0;
            for( k = 0; k < LP_N; k+=3 )
            {
                outLP = SecondOrderfilter( outLP, LP_b+k, LP_a+k+1, LP_delays+m );
                m += 2;
            }

            // Loop for implementing Band-Pass Filter.
            outBP = in;
            m = 0;
            for( k = 0; k < BP_N; k+=3 )
            {
                outBP = SecondOrderfilter( outBP, BP_b+k, BP_a+k+1, BP_delays+m );
                m += 2;
            }  // End of filter implementation loop.

            // Loop for implementing High-Pass Filter.
            outHP = in;
            m = 0;
            for( k = 0; k < HP_N; k+=3 )
            {
                outHP = SecondOrderfilter( outHP, HP_b+k, HP_a+k+1, HP_delays+m );
                m += 2;
            }  // End of filter implementation loop.

            // Write out data for this time index.
            fprintf( fout, "%f,%f,%f,%f\n", in, outLP, outBP, outHP ); // Write out

        }  // End of time loop.

        fclose( fout );   // Close file

    } // End of valid open test

    return 0;
} // end of main.
```


Input Chirp
Low-Pass Filtered
Band-Pass Filtered
High-Pass Filtered
1
0
-1
2
-2
0
500
1000
1500
2000
2500
3000
3500
4000
4500