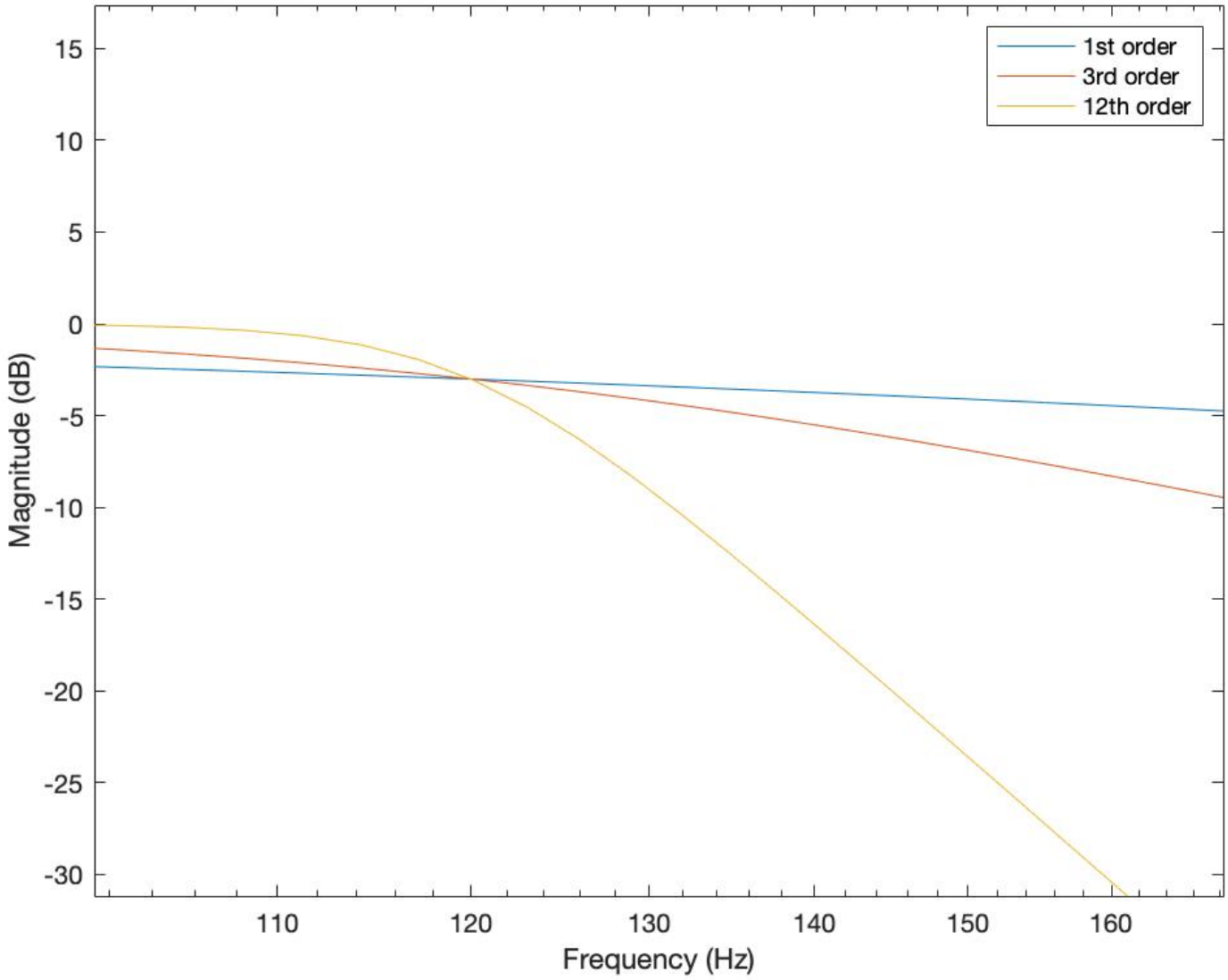
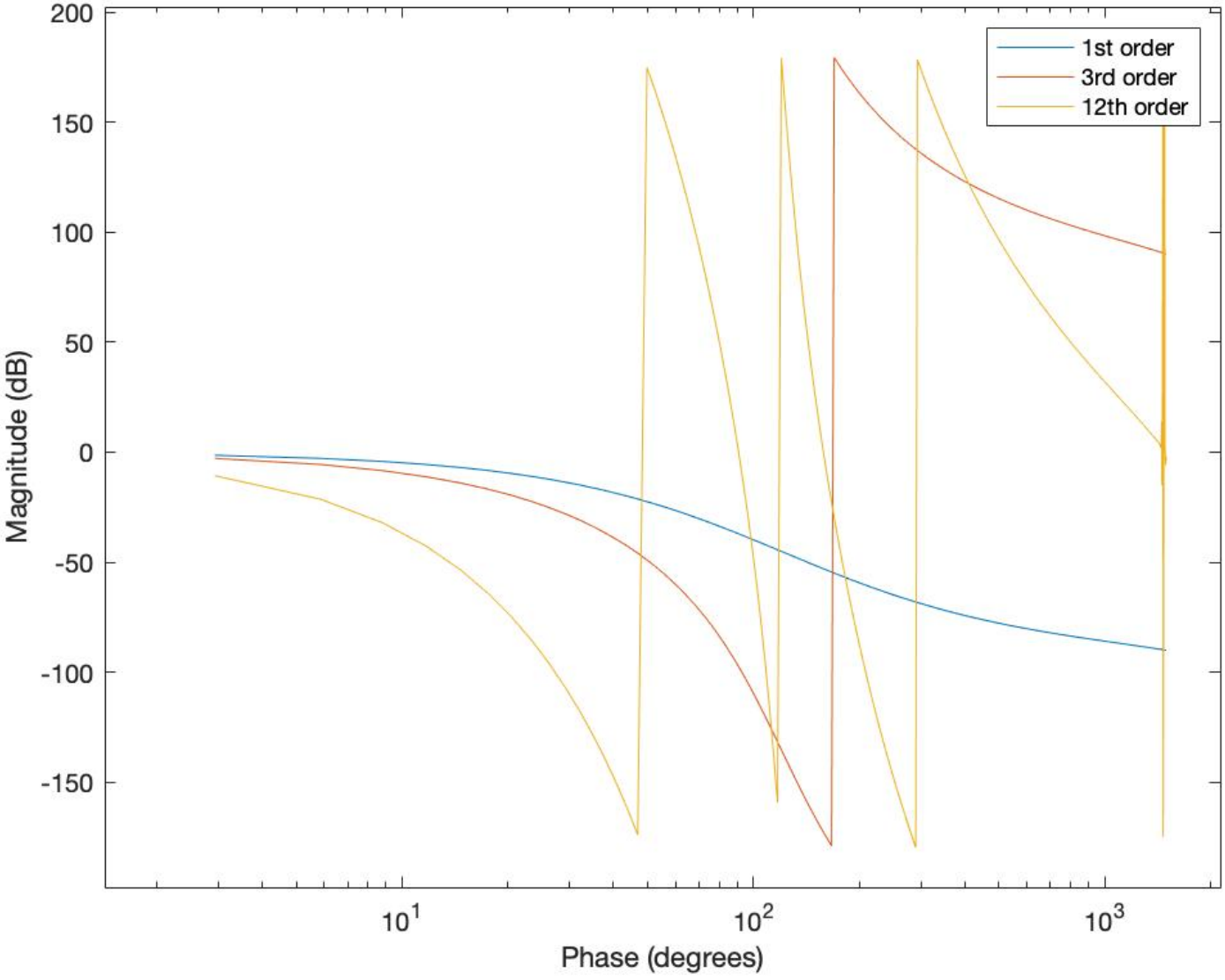
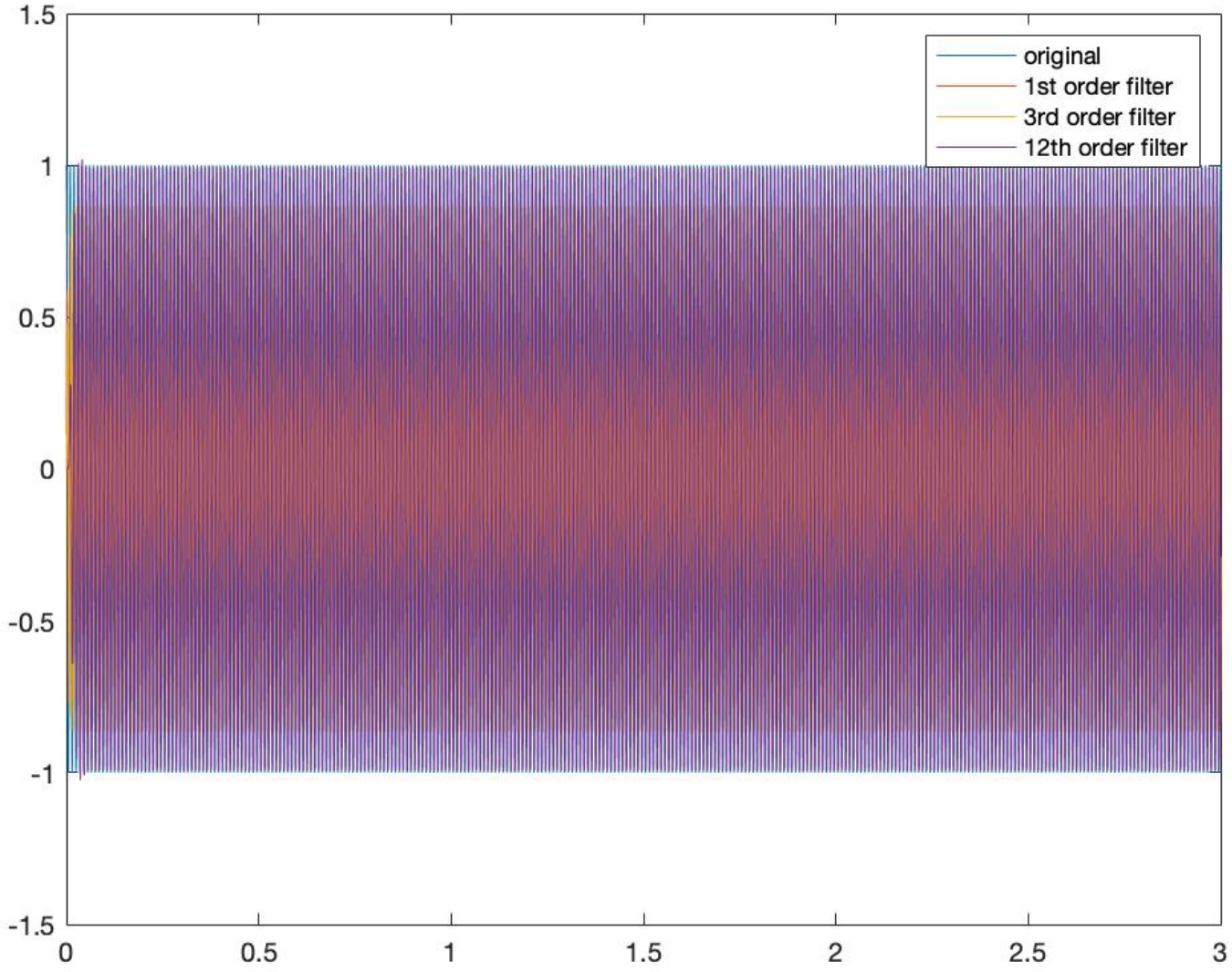
```
%% BME 306 Lab 6 - Filters
% Alexander Ross
% 11/28/19

%% Question 1

fs = 3000;
t = 0:1/fs:3;
s = cos(2*pi*100*t);
plot(t,s)

%% Question 2
[b,a] = butter(3, 120/(fs/2));
[H,w] = freqz(b,a,512,fs);
% w = (w)*((fs)/(2*pi));
semilogx(w,20*log10(abs(H)));
semilogx(w,angle(H)*(180/pi));

%% Question 3

[b1,a1] = butter(1, 120/(fs/2));
[b2,a2] = butter(3, 120/(fs/2));
[b3,a3] = butter(12, 120/(fs/2));

[H1,w1] = freqz(b1,a1,512,fs);
[H2,w2] = freqz(b2,a2,512,fs);
[H3,w3] = freqz(b3,a3,512,fs);

% w1 = (w1)*((fs)/(2*pi));
% w2 = (w2)*((fs)/(2*pi));
% w3 = (w3)*((fs)/(2*pi));

semilogx(w1,20*log10(abs(H1)))
hold on
semilogx(w2,20*log10(abs(H2)))
semilogx(w3,20*log10(abs(H3)))
hold off
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
legend('1st order','3rd order','12th order');


semilogx(w1,angle(H1)*(180/pi))
hold on
semilogx(w2,angle(H2)*(180/pi))
semilogx(w3,angle(H3)*(180/pi))
hold off
xlabel('Phase (degrees)');
ylabel('Magnitude (dB)');
legend('1st order','3rd order','12th order');

% Explanation: Higher order filters make the transition from pass band to
% stop band occur more quickly than lower order filters.  This can be seen
% on the bode plots, where the magnitude decreases more sharply for the
% higher order filters.  However, higer order filters are more complex than
% low order filters when it comes to designing them practially, so there is
```

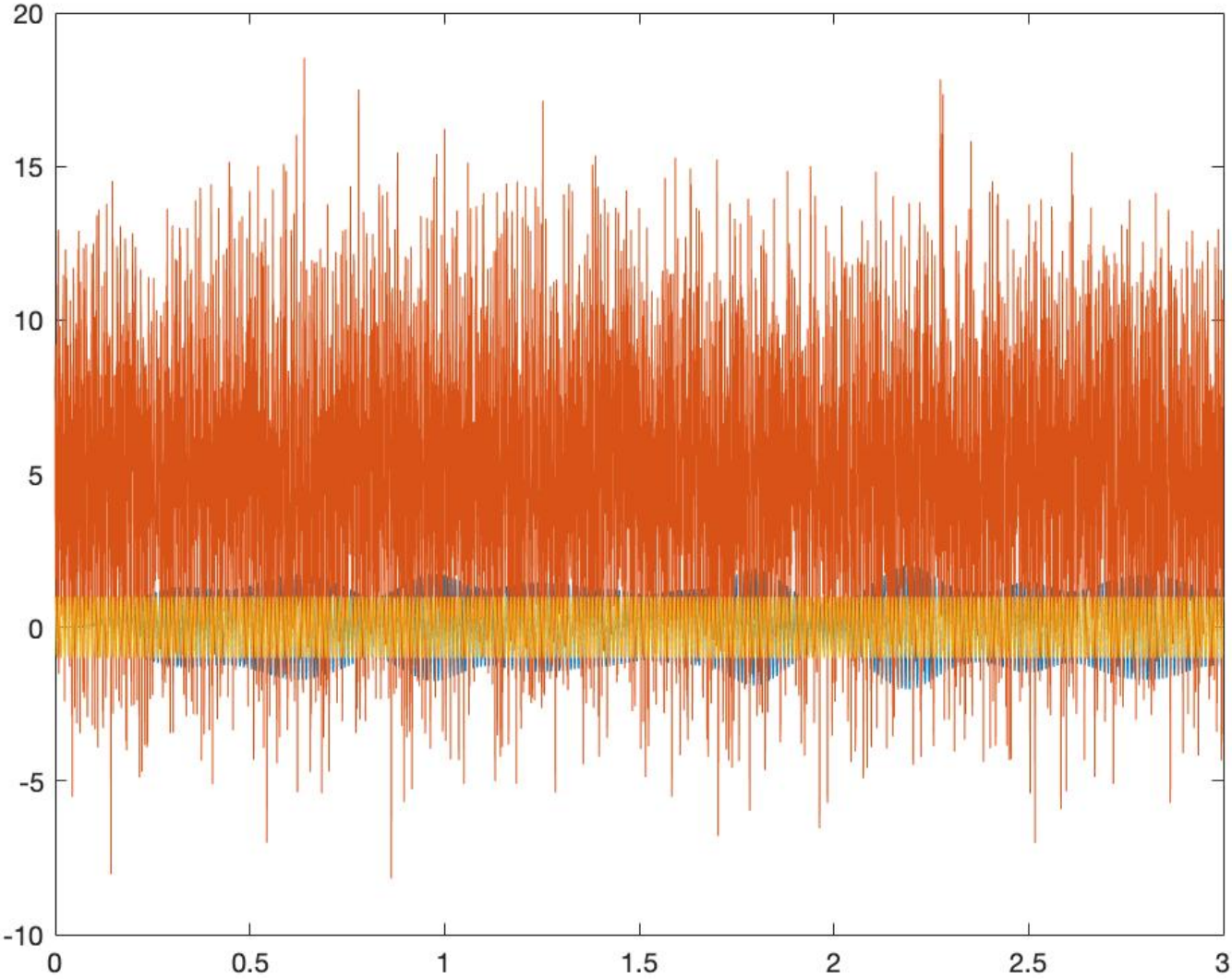
```
% a tradeoff between efficiency and utility.

%% Question 4

y1 = filter(b1,a1,s);
y2 = filter(b2,a2,s);
y3 = filter(b3,a3,s);

plot(t,s)
hold on
plot(t,y1)
hold on
plot(t,y2)
hold on
plot(t,y3)
hold off
legend('original','1st order filter','3rd order filter','12th order filter')

%% Question 5

signal = s + 3*cos(2*pi*150*t) + 3*randn(size(t)) + 5;

[b,a] = butter(5, [97 104]/(fs/2));
[b1,a1] = butter(5, [99 101]/(fs/2));
[b2,a2] = butter(5, [93 107]/(fs/2));
[b3,a3] = butter(5, [90 110]/(fs/2));

y = filter(b,a,signal);
y1 = filter(b1,a1,signal);
y2 = filter(b2,a2,signal);
y3 = filter(b3,a3,signal);

plot(t,y);
hold on
plot(t,signal)
hold on
plot(t,s)
hold off

% plot(t,y1);
% hold on
% plot(t,signal)
% hold on
% plot(t,s)
% hold off
%
% plot(t,y2);
% hold on
% plot(t,signal)
% hold on
% plot(t,s)
% hold off
%
% plot(t,y3);
% hold on
% plot(t,signal)
```

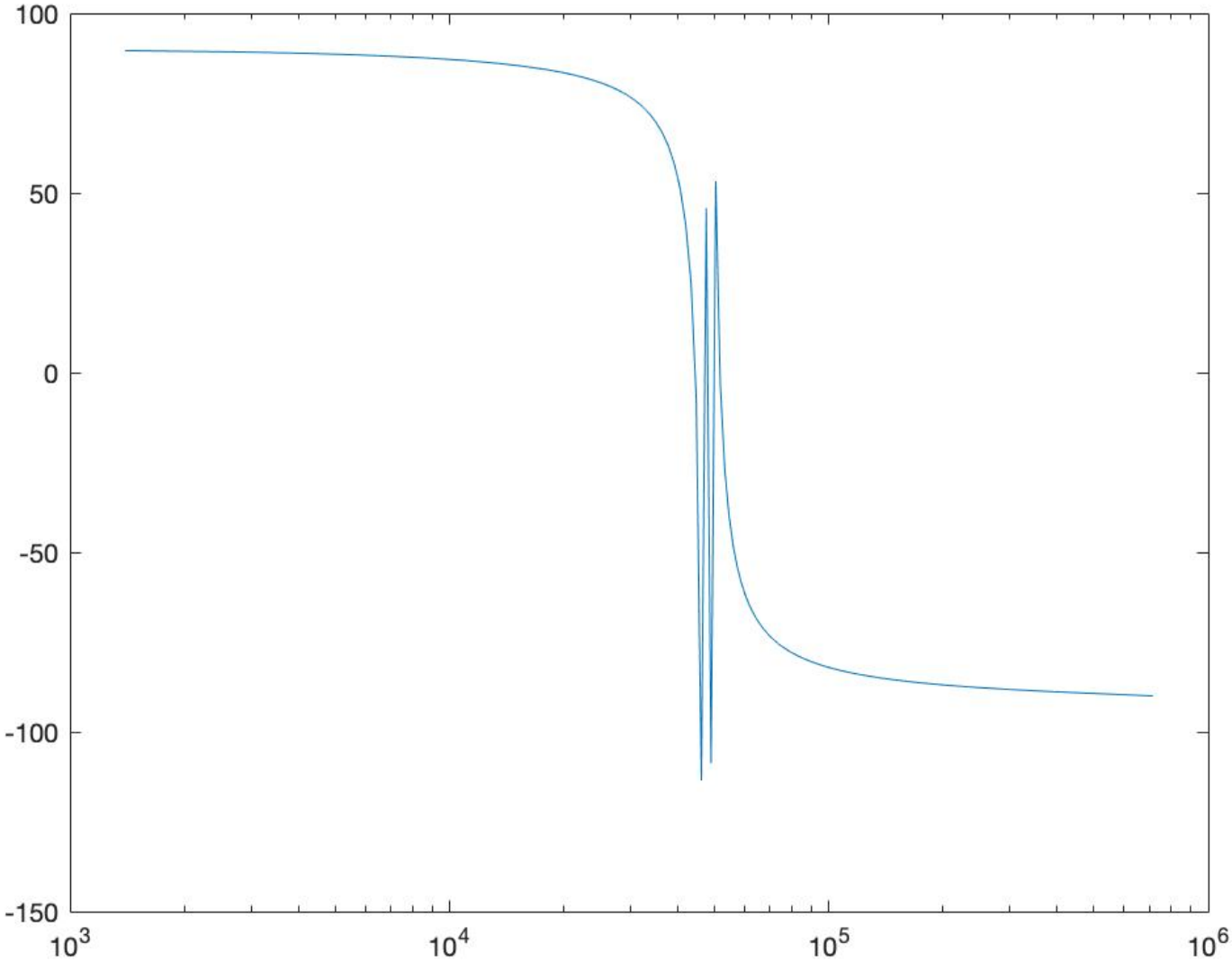
```
% hold on
% plot(t,s)
% hold off

[H,w] = freqz(b,a,512,fs);
w = (w)*((fs)/(2*pi));
semilogx(w,20*log10(abs(H)));
semilogx(w,angle(H)*(180/pi));

inputfft = fft(signal);
outputfft = fft(y);
plot(t,inputfft);
plot(t,outputfft);

% Explanation: The filter does not filter all of the noise in the signal,
% but it does filter most of it.  The filtered signal looks very similar to
% the original signal without the noise.  The settings for the filter
% were chosen because they produced a filtered signal most similar to the
% noise-free signal in a series of tests where the order and bandpass
% frequencies were changed to find the optimal filter settings.


%% Question 6

% See attached TheFilter.m

%% Question 7

% See attached CochlearImplant.m
```

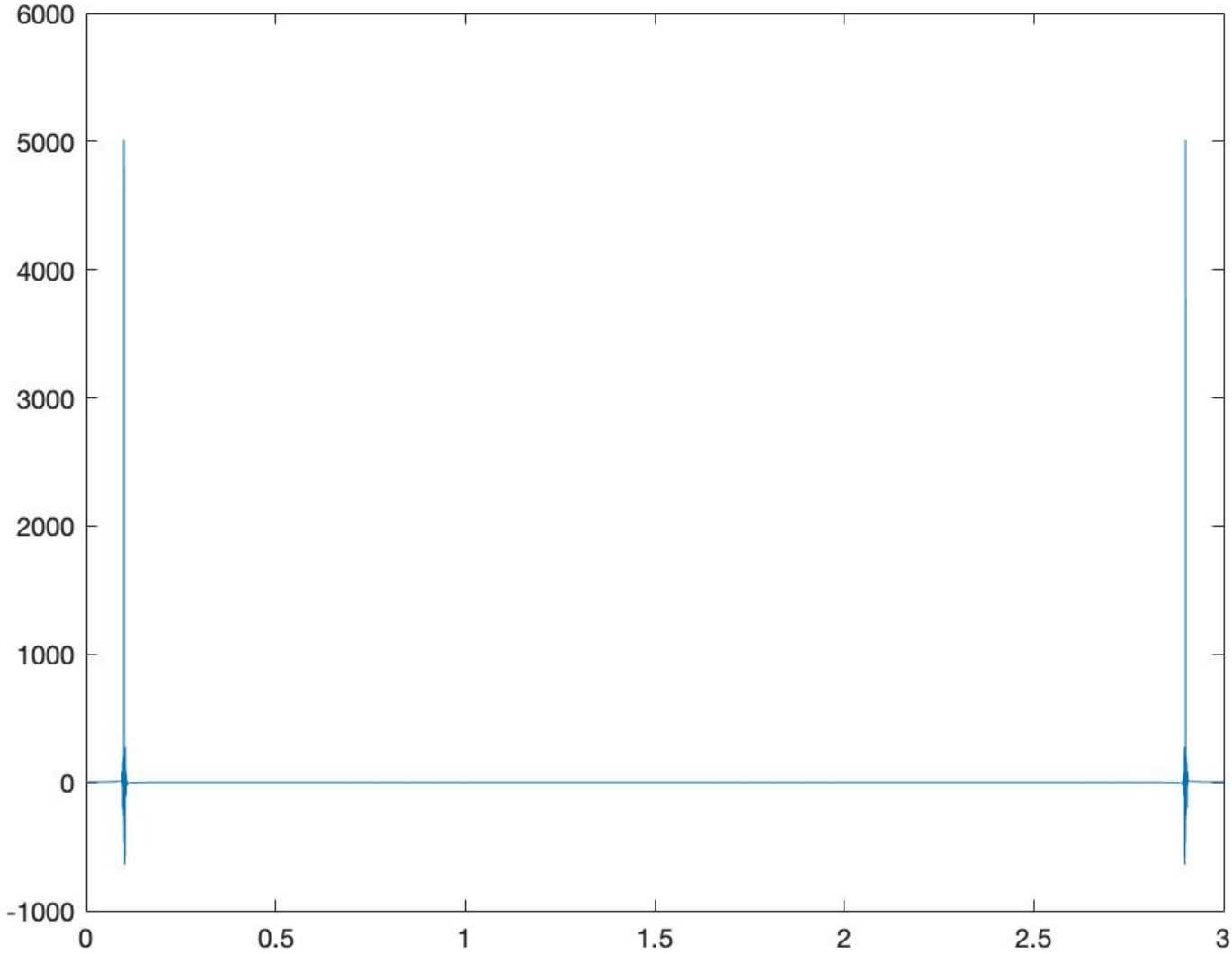

1st order
3rd order
12th order
15
10
5
0
-5
-10
-15
-20
-25
-30
Magnitude (dB)
110
120
130
140
150
160
Frequency (Hz)


1st order
3rd order
12th order
200
150
100
50
0
-50
-100
-150
Magnitude (dB)
10^1
10^2
10^3
Phase (degrees)

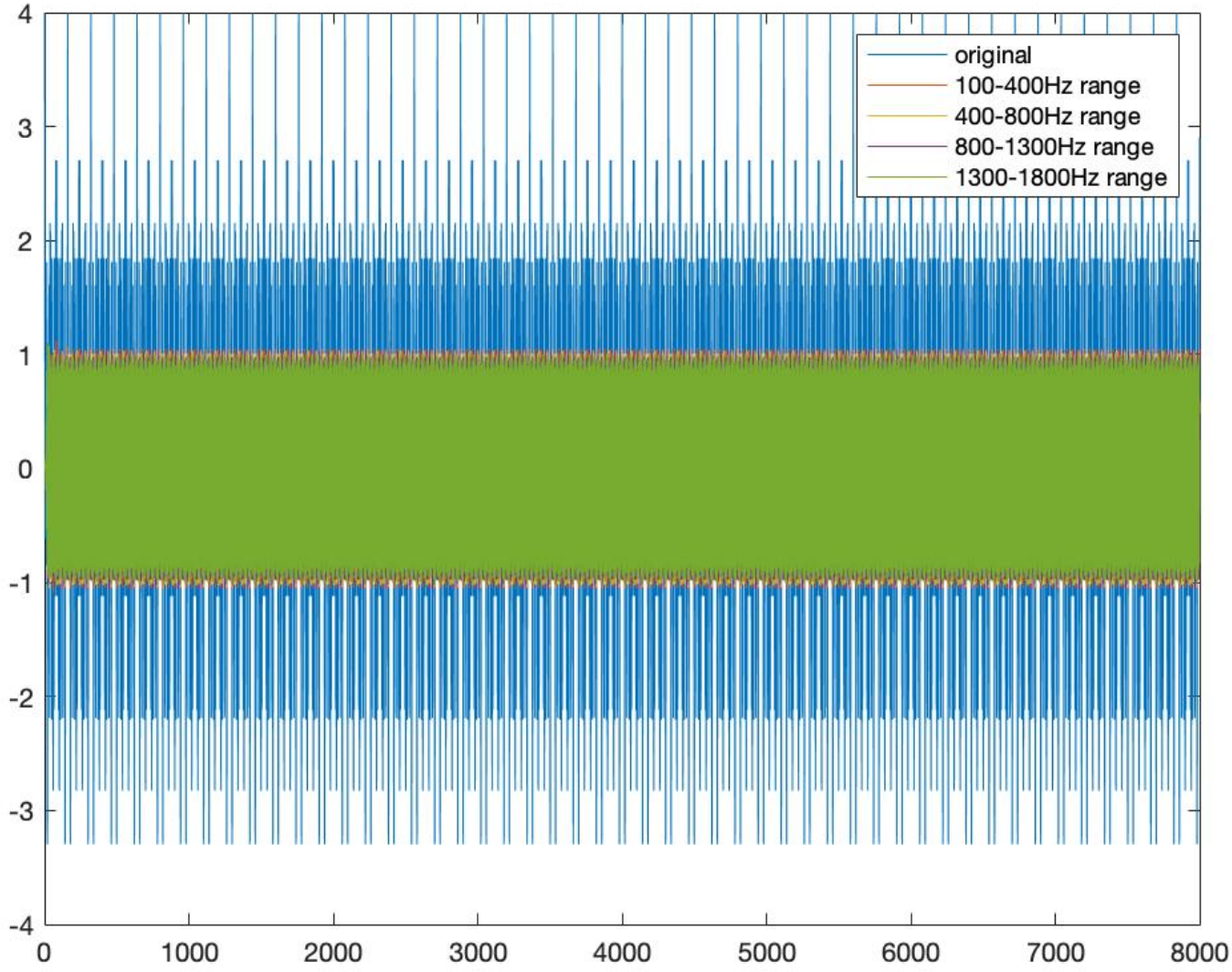

original
1st order filter
3rd order filter
12th order filter
1.5
1
0.5
0
-0.5
-1
-1.5
0
0.5
1
1.5
2
2.5
3

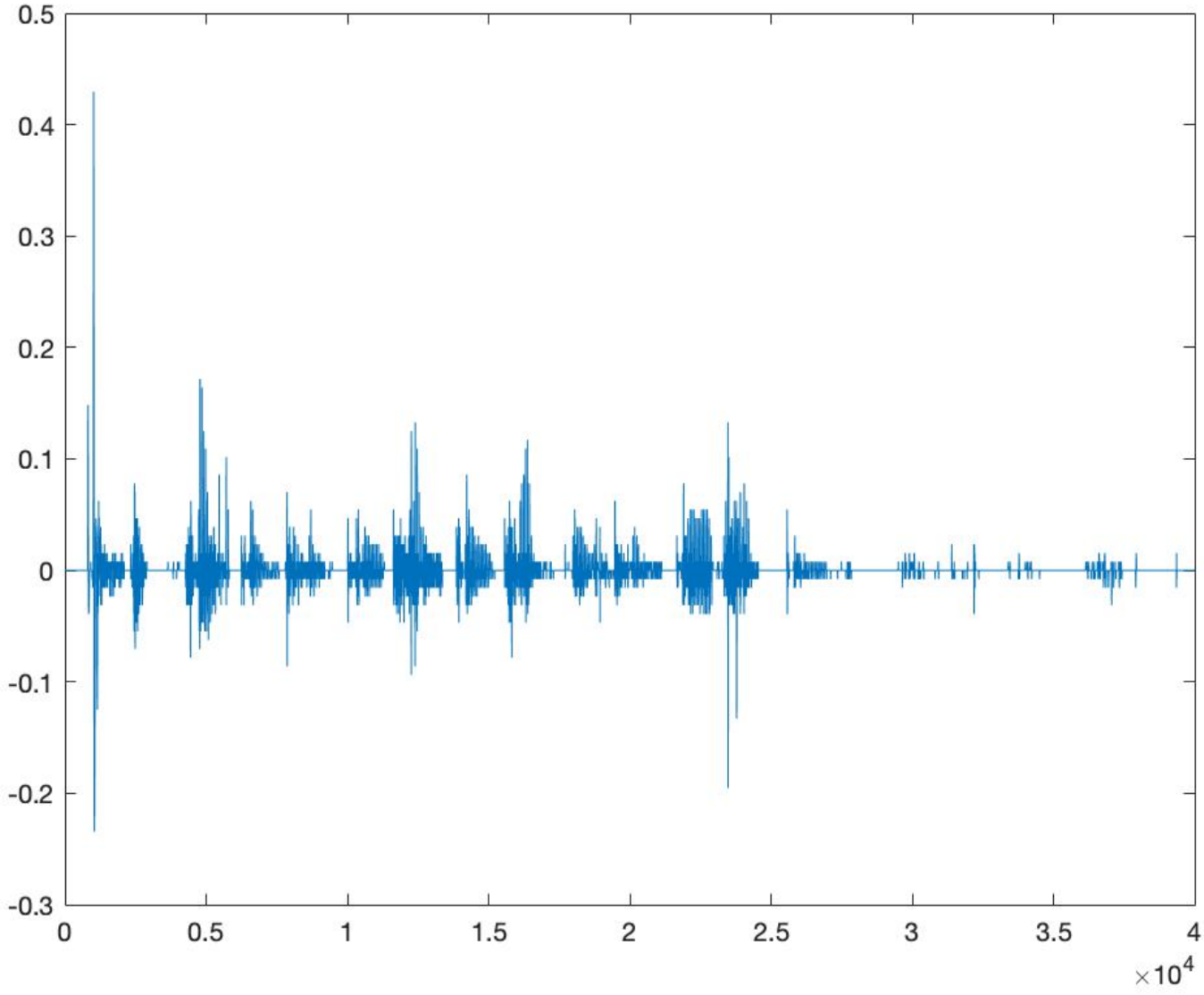

20
15
10
5
0
-5
-10
0
0.5
1
1.5
2
2.5
3

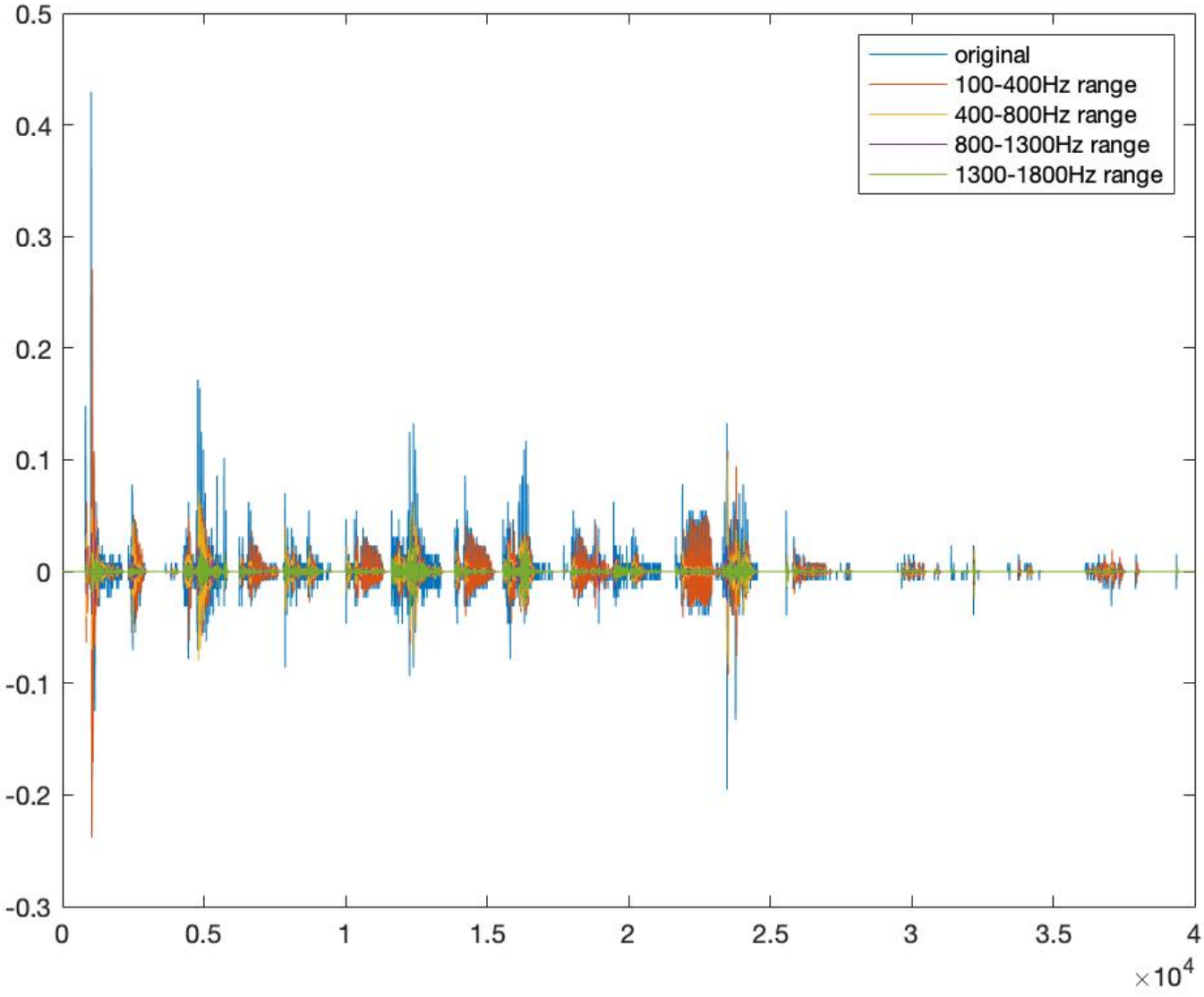

100
50
0
-50
-100
-150
10^3
10^4
10^5
10^6

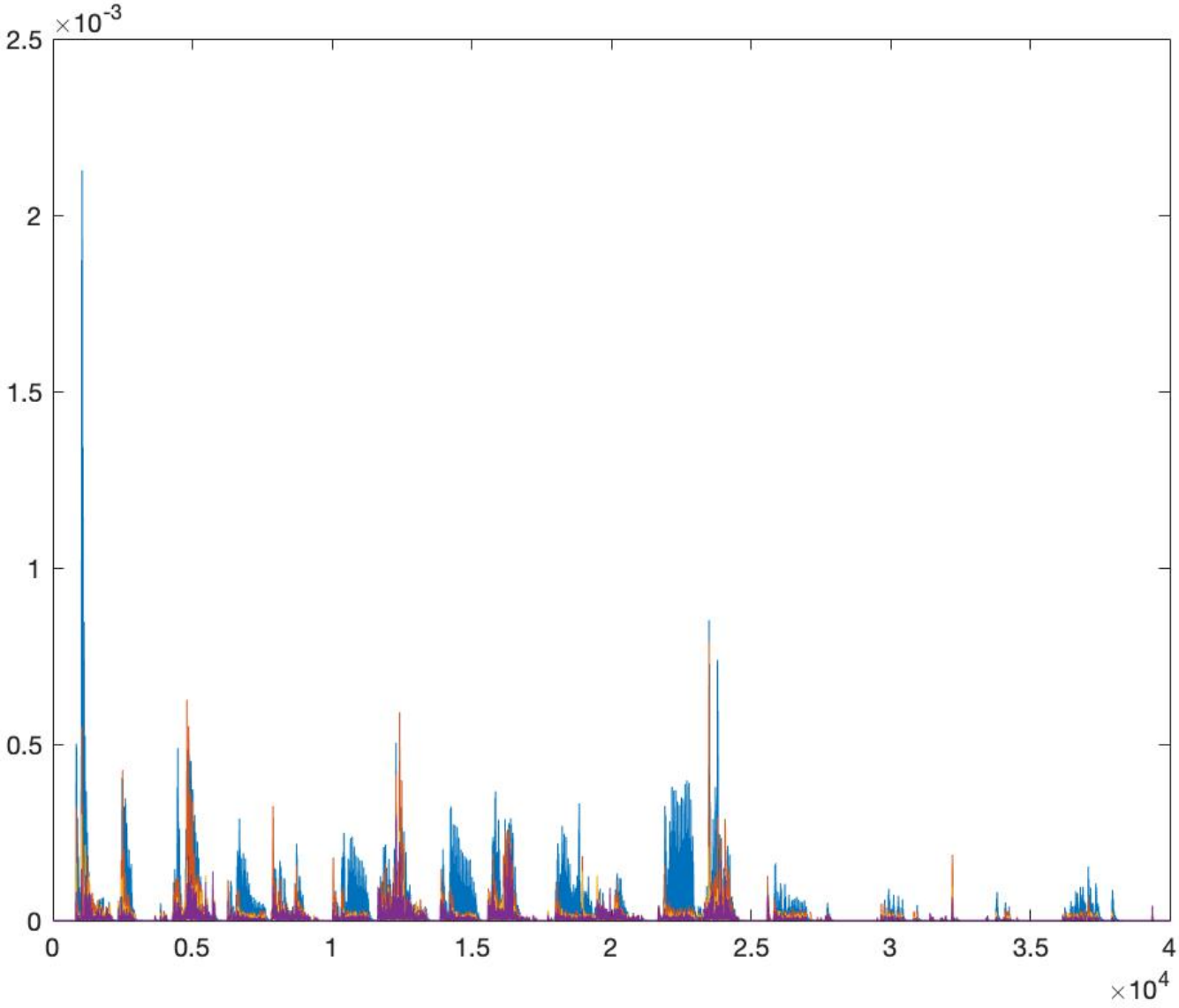

6000
5000
4000
3000
2000
1000
0
-1000
0
0.5
1
1.5
2
2.5
3

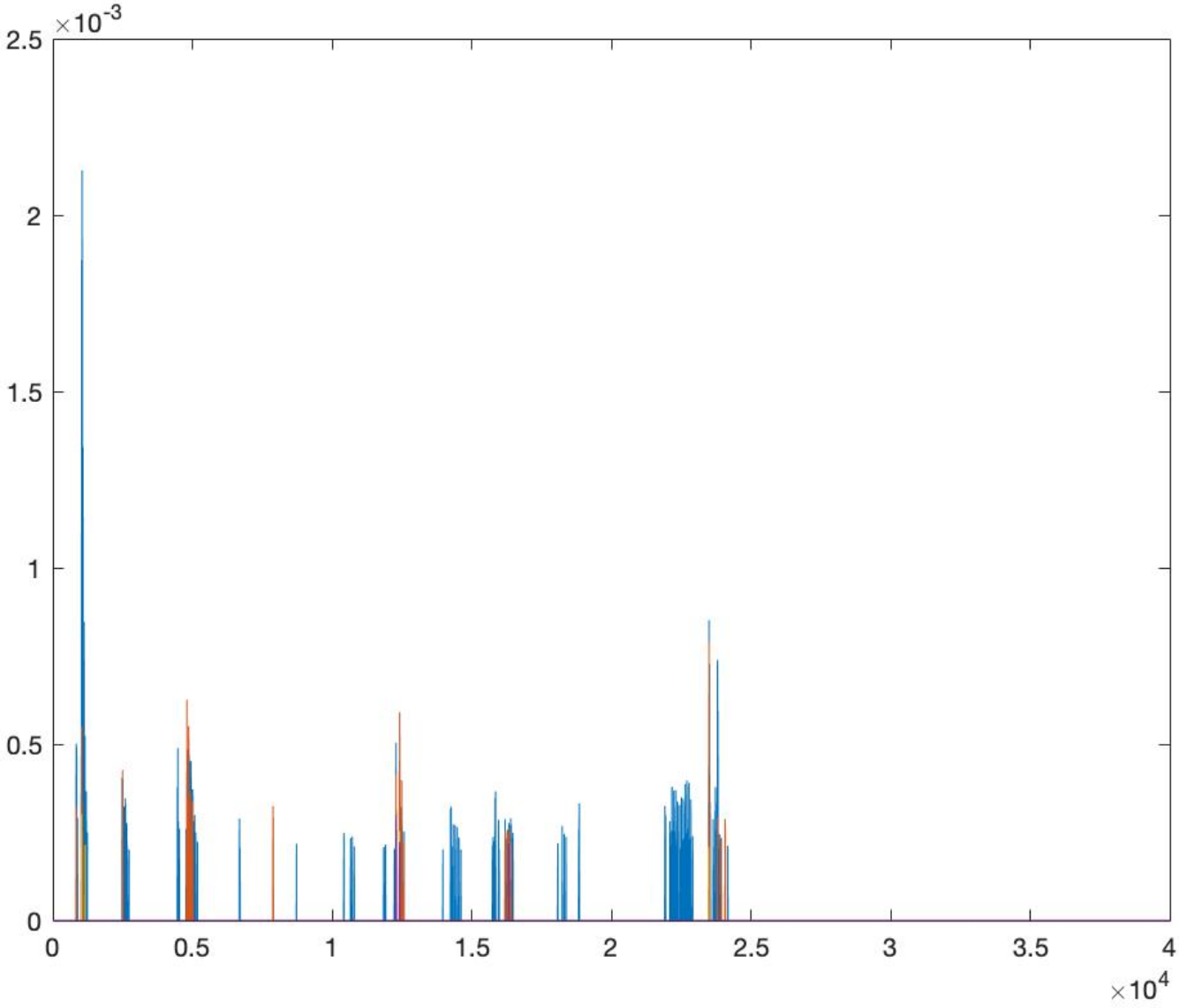
```
function [output] = TheFilter(AudioSignal,SamplingFrequency)

% THEFILTER  Filters an audio signal vector at 4 different frequency bands
%   [output] = TheFilter(doubleArray,fs) filters the audio signal
%   doubleArray at the given sampling frequency fs and returns the filtered
%   signal as an 4-column matrix, with each column being the filtered
%   signal filtered at a different fequency band.

fs = SamplingFrequency;

[b1,a1] = butter(5, [100 400]/(fs/2));
[b2,a2] = butter(5, [400 800]/(fs/2));
[b3,a3] = butter(5, [800 1300]/(fs/2));
[b4,a4] = butter(5, [1300 1800]/(fs/2));

y = filter(b1,a1,AudioSignal)';
y1 = filter(b2,a2,AudioSignal)';
y2 = filter(b3,a3,AudioSignal)';
y3 = filter(b4,a4,AudioSignal)';

% To test: initialize AudioSignal as s

t = (0:1/fs:0.9999)';
s = cos(2*pi*200*t)+cos(2*pi*600*t)+cos(2*pi*1050*t)+cos(2*pi*1550*t);

plot(AudioSignal)
hold on
plot(y);
hold on
plot(y1);
hold on
plot(y2);
hold on
plot(y3);
hold off
legend('original','100-400Hz range','400-800Hz range','800-1300Hz
range','1300-1800Hz range')

output = [y;y1;y2;y3]'; % Returns a matrix of the filtered audio signal at
different pass bands in 4 different columns

end
```

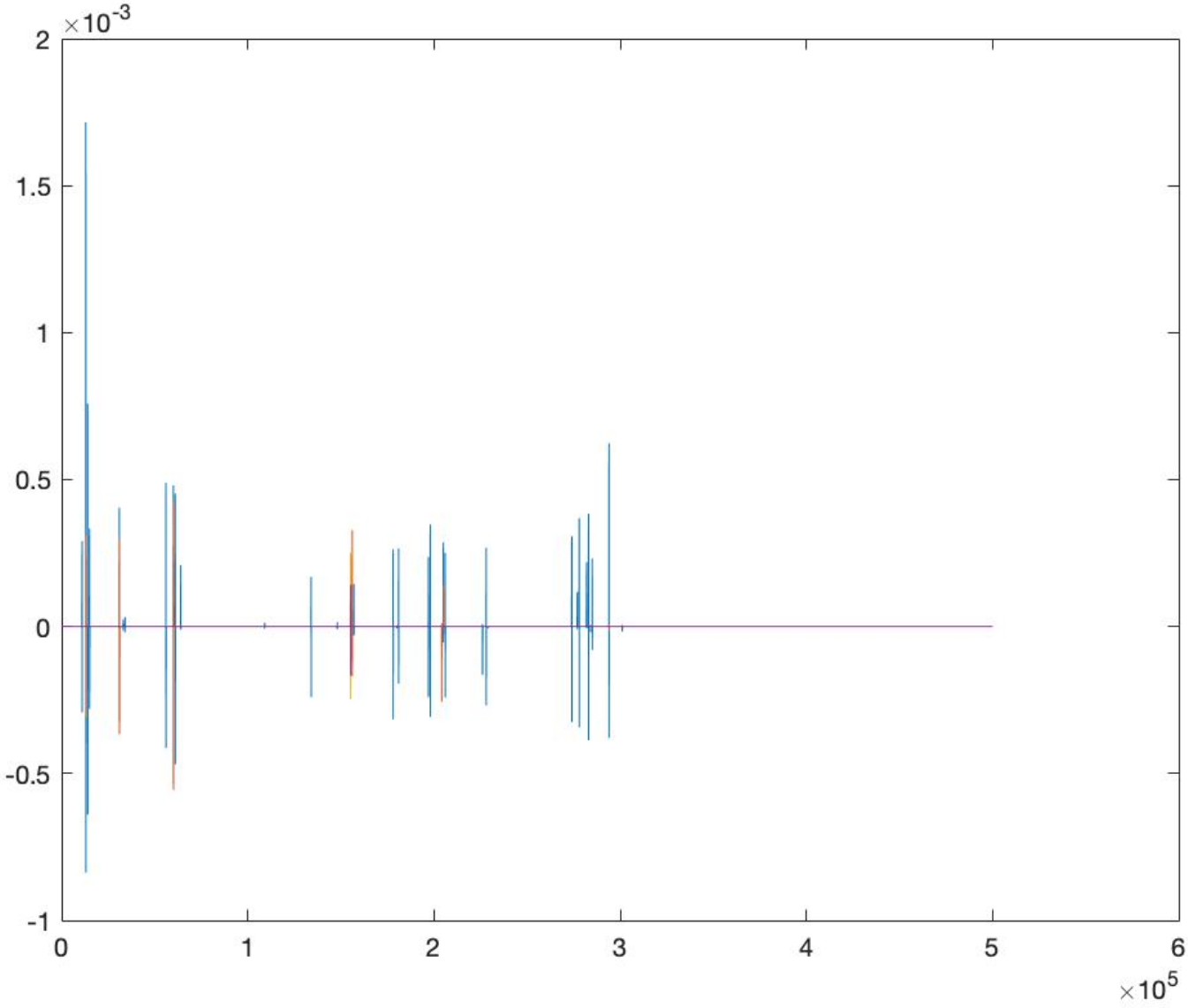

original
100-400Hz range
400-800Hz range
800-1300Hz range
1300-1800Hz range
4
3
2
1
0
-1
-2
-3
-4
0
1000
2000
3000
4000
5000
6000
7000
8000


0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4


original
100-400Hz range
400-800Hz range
800-1300Hz range
1300-1800Hz range
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4


×10^-3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4


×10^-3
2.5
2
1.5
1
0.5
0
0
0.5
1
1.5
2
2.5
3
3.5
4
×10^4


×10^-3
2
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
×10^5