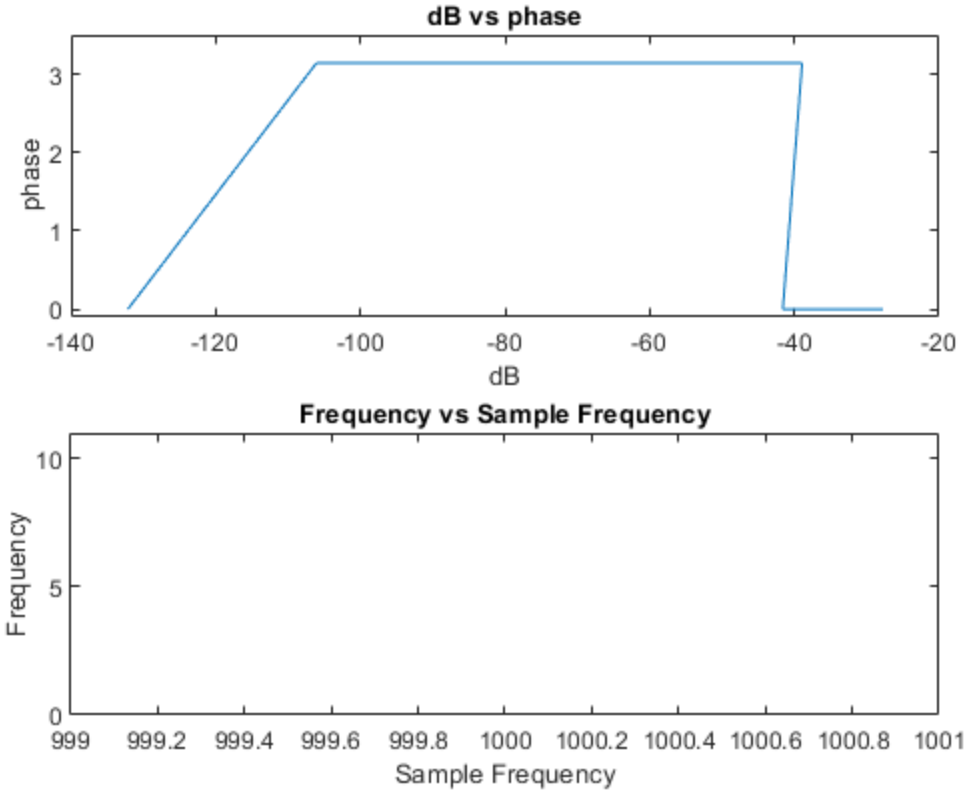
```
% Christopher Morales
% EE 4820 4/24/2022
```

Problem 1

```
%%%%%%%%%
% PART A %
%%%%%%%%%

% 4y[n] + 8y[n-1] - 7y[n-3] -84y[n-3] - 201y[n-4] + 740y[n-5] - 201y[n-6]
% -84y[n-7] - 7y[n-8] + 8y[n-9] + 4y[n-10] = x[n]

% y * (4[n] + 8[n-1] - 7[n-3] -84[n-3] - 201[n-4] + 740[n-5] - 201[n-6]
% -84[n-7] - 7[n-8] + 8[n-9] + 4[n-10]) = x[n]

%  blah = (4[n] + 8[n-1] - 7[n-3] -84[n-3] - 201[n-4] + 740[n-5] - 201[n-6]
% -84[n-7] - 7[n-8] + 8[n-9] + 4[n-10])

% y * blah = x -> H = y / x = 1 / blah

% Frequency Sampling rate (f = fs/nfft -> fs = f*nfft)
fs = 1 * 1000;

% Amount of samples going into the tranfer function
z = 1:fs;

% Transfer Function Hn = Yn/Xn
Xn = 1;
Yn = [4 8 -7 -84 -201 740 -201 -84 -7 8 4];

% Attempt to updated 1000 samples per function
for i = 1 : length(Yn)
    for j = 1 : length(fs)

        Yn(i) = Yn(i)*z(j).^(i);

    end
end

Hn = Xn./Yn;

% Frequency domain vector
f = [ 1 : length(Hn) ] * fs/1000 ;

% Getting the phase of the transfer function
Hnphase = angle(Hn);
```

```
% Signal to dB ( y = 20 * log10(abs(Hn)) )
Hn = 20 * log(abs(Hn));

% Making a vector of ones
one = fs * ones(1, length(f));

% Creating a figure
figure(1);

subplot(2,1,1);
plot(Hn, Hnphase);
title('dB vs phase');
xlabel('dB');
ylabel('phase');
axis([-140 -20 -0.1 3.5]);

subplot(2, 1, 2);
plot(fs, f );
title('Frequency vs Sample Frequency');
xlabel('Sample Frequency');
ylabel('Frequency');

% By looking at the first plot (cB vs phase), we can tell the type of
% filter is a bandpass filter. Eliminating very high and low frequencies
% around -38 dB the filter haves a sharp cutoff while the begining has a
% slope

% A passband gain is a logrithmic scale in decibel. Looking at the singal
% to dB line, we can see the decibel, the output when given will be
% subtracted by three due to ripple effect or small impulses of gain of db
% that is not in steady state form

% The cutoff frequency, we need to apply the fast fourier transform and to
% plot the magnitude of the fft and frequency. In doing so, we will get a
% spectrum of of the signal, whenever steady state starts to becomes
% unstable then we want to cutoff that signal. So when visually looking at
% the graph, we can determine when the singal starts to become unstable,
% example will be demonstrated in problem 2. Where we need to determine
% what is fc and we are approaximating fc leaving steady state.

Xf = fft(Hn);
Xfmag = abs(Hn);
```

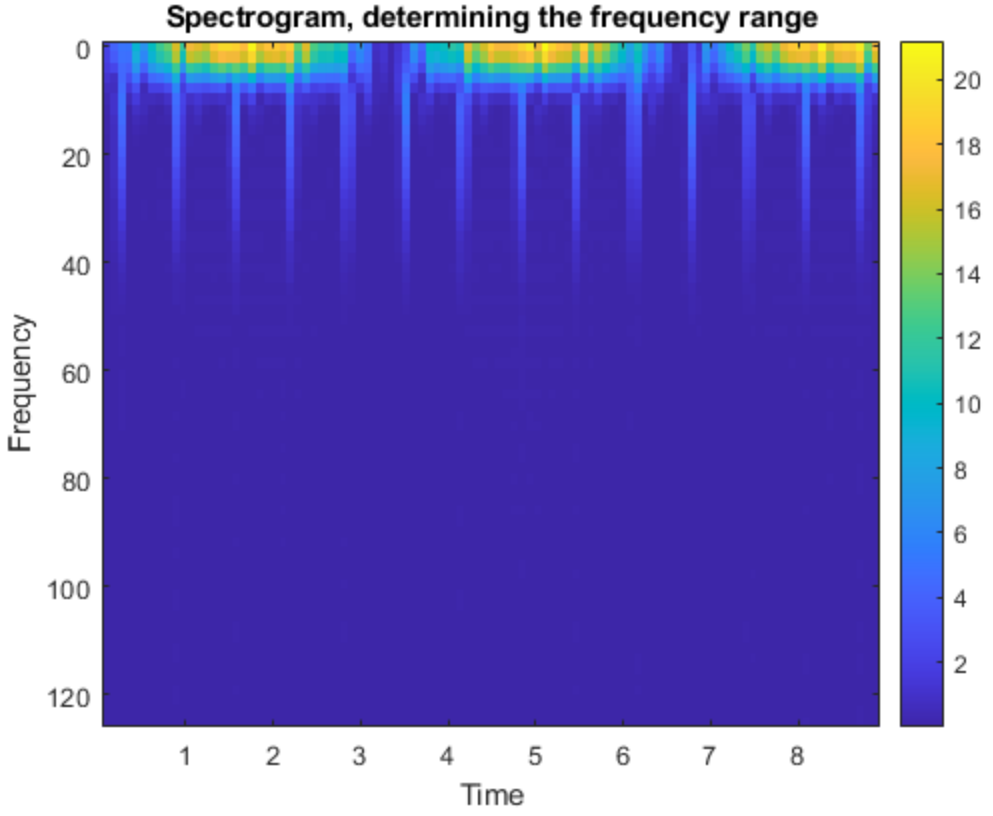


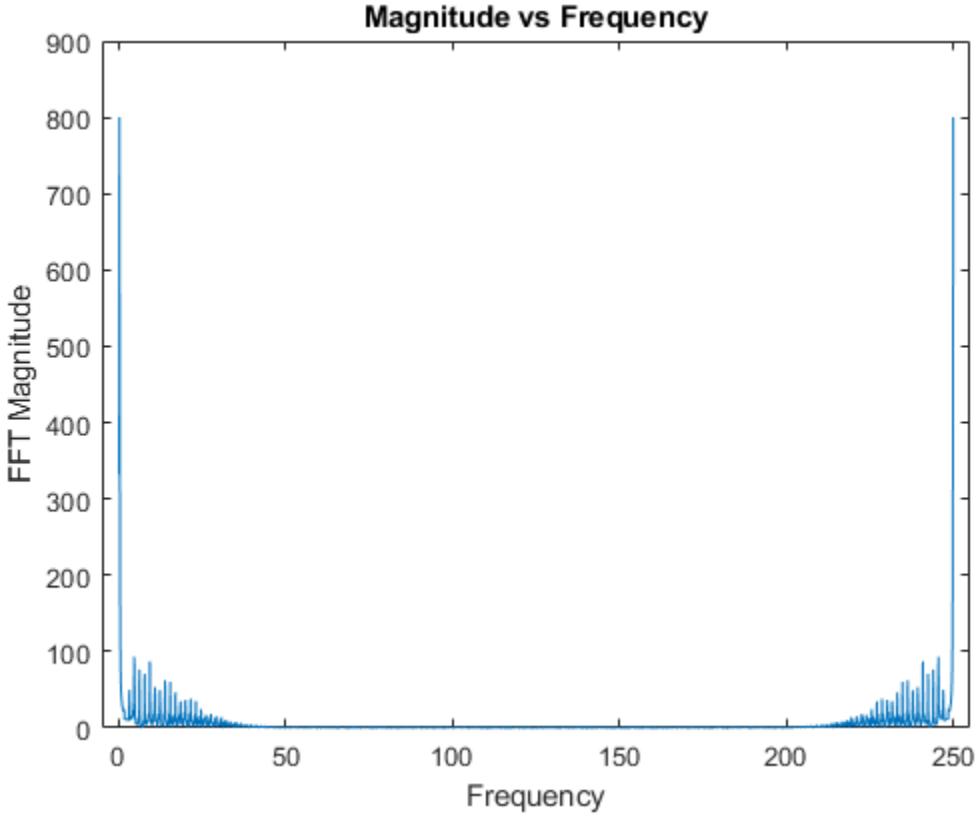
Problem 2

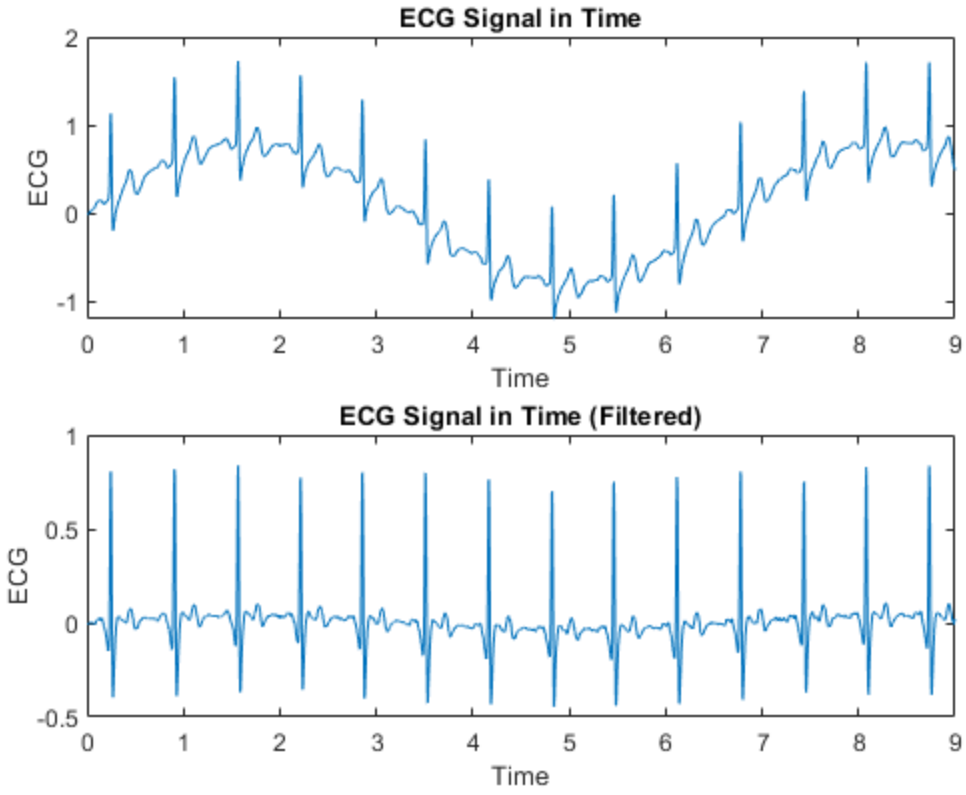
```
% Loading file
load ecg_fs250Hz.mat;

% Creating a figure
figure(2);

% Creating the time domain vector
t1 = [1:length(x)]/fs;

% Plotting the signal and determining the what signal we are dealing with
subplot(2,1,1);
plot(t1, x);
title("ECG Signal in Time");
xlabel('Time');
ylabel('ECG');

% By looking at the plot we can see a bunch of high signals spiking. To
% eliminate these signals we need to use a low pass filter to allow the
% low frequency and eliminating the high frequency.

% Resolution
Nfft = 2^7;

% Creating the length
```

```
L = length(x);

% Creating how many windows
Lwin = round(L/50);

% Creating the type of window
win = window(@hamming, Lwin);

% How much to overlap on the window
Noverlap = round(Lwin*0.50);

% Applying the spectrogram
%[y, f, t] = stft(x, fs,'Window', win, 'OverlapLength', Noverlap, 'FFTlength', Nfft);
[y, f, t] = spectrogram(x, win, Noverlap, Nfft, fs);


% Creating a figure
figure(3);

% Plotting the signal before filtering for comparing before and after
imagesc(t,f,abs(y));
colorbar;
xlabel('Time');
ylabel('Frequency');
title('Spectrogram, determining the frequency range');

% By viewing the spectrum and zooming in we can see the range of the
% frequency is from 0 to 6 hertz with a duration of 1.5 seconds following
% with no signal for 1.5 seconds. I would make the cut off frequency
% between 0 to 6 hertz to eliminate any noise in the system.
% HOWEVER, the frequency is so low (0 to 6 hertz) and determining the cut
% off frequency would be difficult due to the bandwidth we have.
% With this in mind, we will do the fast fourier transform to determine the
% frequency cutoff happens

% Applying the fast fourier transform
Xf = fft(x);

% Getting the magnitude of the fft
Xfmag = abs(Xf);

% Creating figure
figure(4);

% Plotting the magnitude and the time vector to determine and visually
% inspect what will be the frequency cutoff that will be used for the
% filtering process

% Frequency vector
f1 = [1:length(t1)] * fs/length(t1);

% Plotting to determine fc
plot(f1, Xfmag);
```

```
title('Magnitude vs Frequency');
xlabel('Frequency');
ylabel('FFT Magnitude');

% Zooming out to see the giant spikes but we could see at the beginning and
% at the end of the spectrum that the signal is
axis([-5 255 0 900])

% My obsevation to determine frequency cutoff is:
fc = 5;

% Frequency Constraint
Wn = fc/(fs/2);

% Filter Order, Nyquist Frequnecy (stated by matlab)
N = Lwin - 1;

% Filter Coeffiecient, high pass filter
B = fir1(N, Wn, "high");

% Performs zero phase filtering (eliminates oscillation of the signal)
y1 = filtfilt(B, 1, x);

% Recalling figure 1
figure(2);

% Plotting the original signal
subplot(2,1,1);
plot(t1, x);
title("ECG Signal in Time");
xlabel('Time');
ylabel('ECG');

% Plotting the filtered signal
subplot(2,1,2);
plot(t1,y1);
title('ECG Signal in Time (Filtered)');
xlabel('Time');
ylabel('ECG');

% Final Observations, the original signal had high frequencies while the
% ECG looks like a sinsoidal (showing oscillation). By  using fft to
% determine when the frequency starts to be "noisy" (f vs XFmag) we will
% cut off the frequncy. In doing so, when we applied the FIR1 and FILTFILT,
% we are stopping/eliminating the noise/oscillation. Once we plot the
% filtered ECG signal, we can see the magnitude have been reduce by half
% and the signal is ub a constant state (not acting like a sinsoidal)
```


Spectrogram, determining the frequency range
Frequency
Time
0
20
40
60
80
100
120
1
2
3
4
5
6
7
8
2
4
6
8
10
12
14
16
18
20

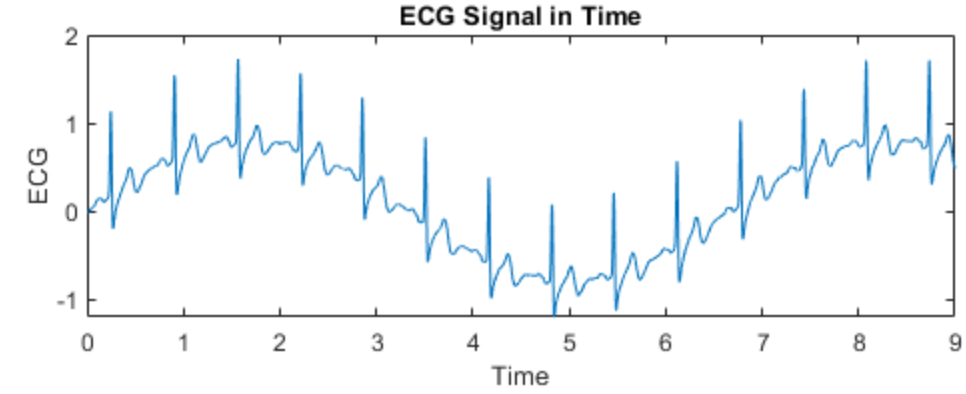

Magnitude vs Frequency
FFT Magnitude
Frequency
0
100
200
300
400
500
600
700
800
900
0
50
100
150
200
250

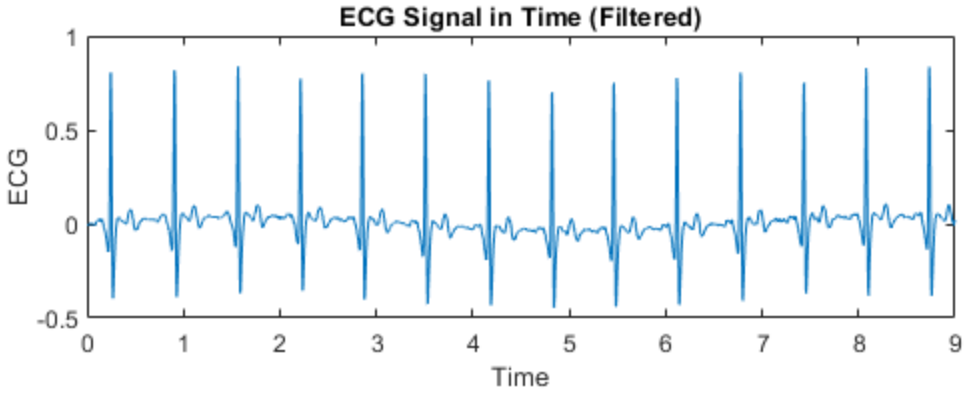


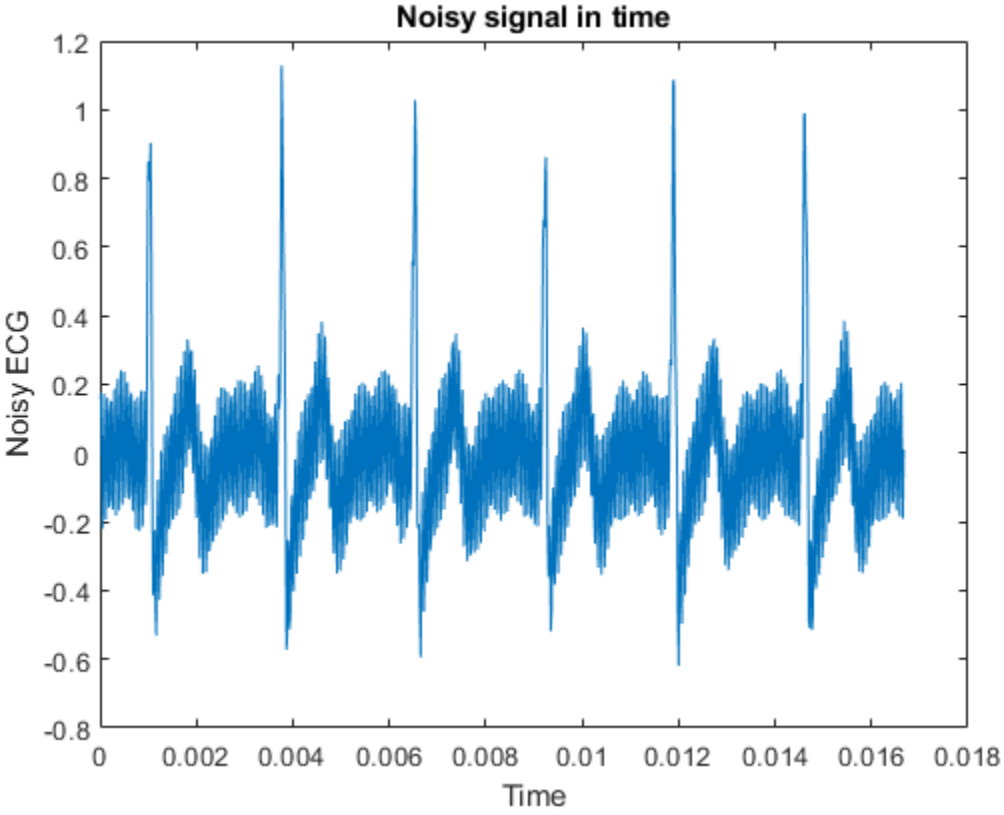
Problem 3

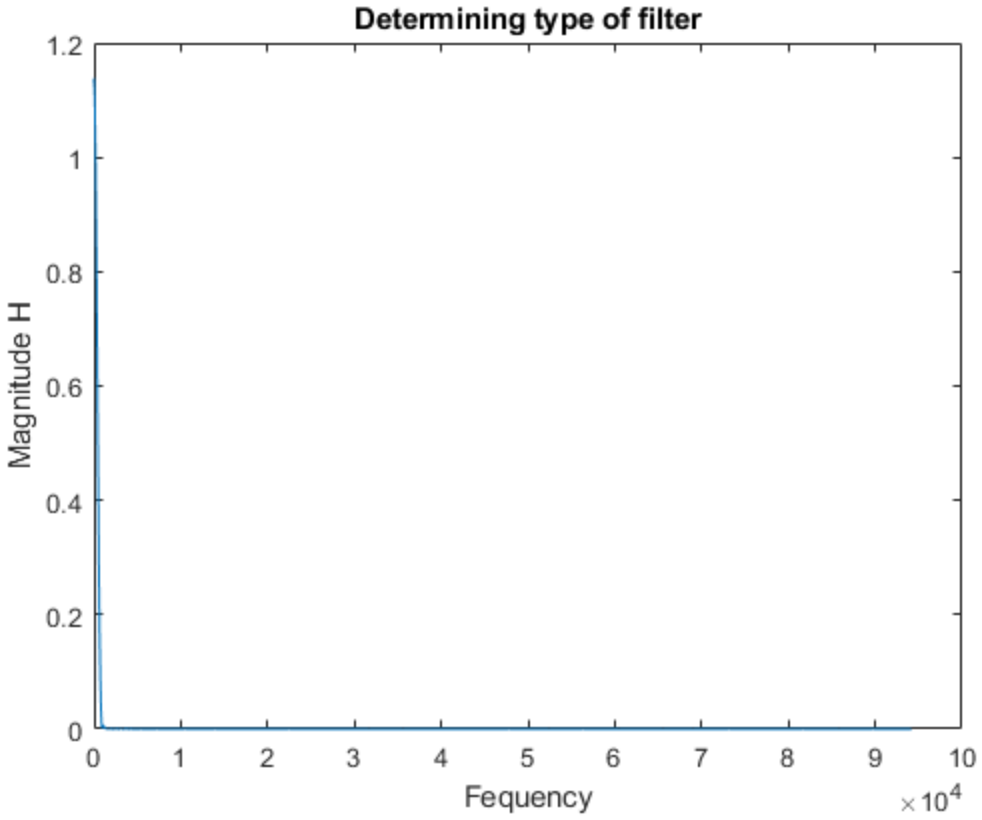
```
% Load file
load ecg60HzNoise.mat;

% Frequency (Noise)
fn = 60;

% Number of samples
nfft = length(x);

% Frequency Sample (f = fs/nfft -> fs = f*nfft, f = fn)
fs = fn*nfft;

% Creating the time vector
t = [1:length(x)]/fs;

% Creating the figure
figure(5);

% Plotting corrupt ECG signal
plot(t, x);
title('Noisy signal in time');
xlabel('Time');
ylabel('Noisy ECG');
```

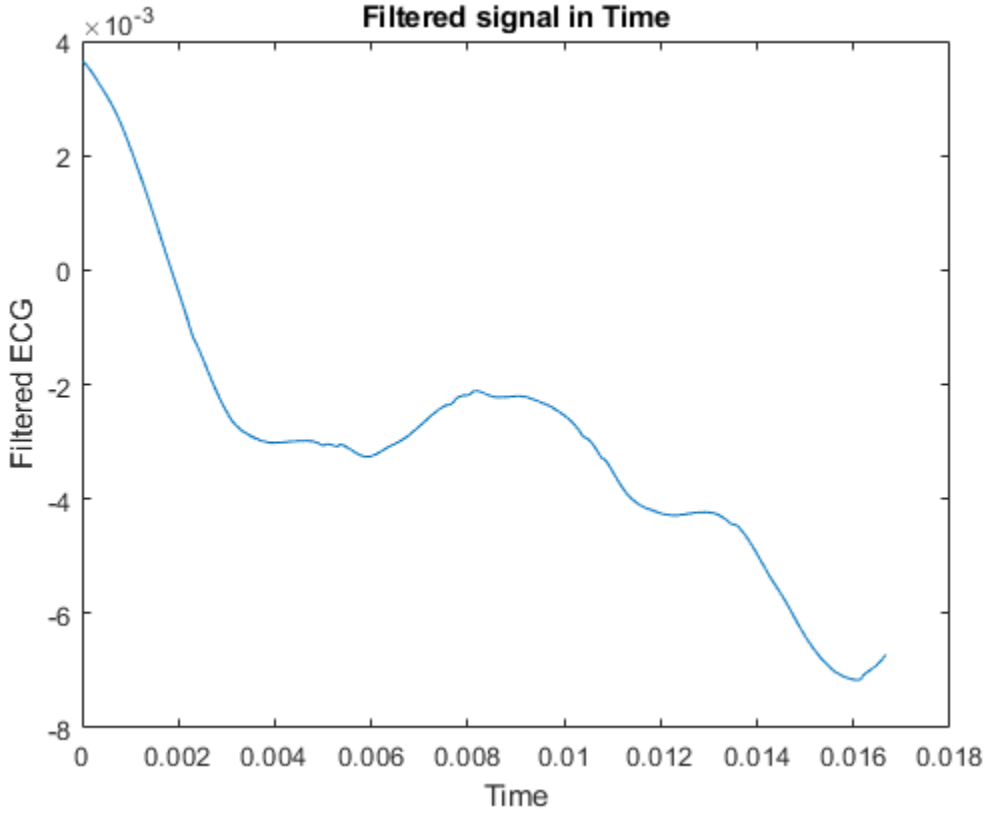
```
% Frequency cutoff
fc = [57:62];
%fc = 60;

% Creating the length
L = length(x);

% Creating how many windows
Lwin = round(L/200);

% Frequency Constraint
Wn = fc/(fs/2);

% Filter Order
N = 512;

% Filter Coeffiecient, high pass filter
B = fir1(N, Wn, "bandpass");

% Getting the convolution
y = conv(x, B, 'same');

% Samples for the xero pahse to work
Nfft = 2^10;

% Performs zero phase filtering
[H, w] = freqz(B, 1, Nfft);

% Creating the frequency vector (f = (fs/2) * w )
f = (fs/2) * w ;

% Creating a figure
figure(6);

% Checking/verifying what type of filter
plot(f,abs(H));
title('Determining type of filter');
xlabel('Fequency');
ylabel('Magnitude H');

% Creating a figure
figure(7);

% Plotting the filter signal
plot(t, y)
title('Filtered signal in Time');
xlabel('Time');
ylabel('Filtered ECG');
```


ECG Signal in Time
ECG
Time
2
1
0
-1
0
1
2
3
4
5
6
7
8
9


ECG Signal in Time (Filtered)
ECG
Time
1
0.5
0
-0.5
0
1
2
3
4
5
6
7
8
9


Noisy signal in time
Noisy ECG
Time
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
0.002
0.004
0.006
0.008
0.01
0.012
0.014
0.016
0.018


Determining type of filter
Magnitude H
Fequency
×10^4
1.2
1
0.8
0.6
0.4
0.2
0
0 1 2 3 4 5 6 7 8 9 10


Filtered signal in Time
×10^-3
Filtered ECG
Time
4
2
0
-2
-4
-6
-8
0 0.002 0.004 0.006 0.008 0.01 0.012 0.014 0.016 0.018

Published with MATLAB® R2021b