Ahsanullah University of Science and Technology

Department of EEE

Report

Design and analysis of Digital Filter and its application to remove noise from Real Life

Date of submission: 30.01.2024

Submitted by:

Name: Navid Aziz
ID: 20200205160

Year: 3rd

Semester: 2nd

Session: Spring 23

Subject code: EEE 3218

Objectives:

i. Understanding Digital Filters.
ii. Noise Analysis.
iii. Filter Design.
iv. Implementation and Simulation.
v. Performance Metrics.
vi. Real-Life Applications.
vii. Optimization and Trade-offs.
viii. Adaptability and Robustness.
ix. Comparison with Analog Filters.
x. Documentation and Analysis.

Equipment/Software:

MATLAB/OCTAVE

Tasks:

In Part 1, we have to design a filter to separate a frequency from a signal generated in MATLAB.

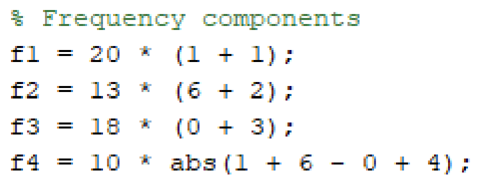
In part 2, we have to design a filter to remove noise from a real-world signal i.e., ECG signal.

Procedures:

Part-1:

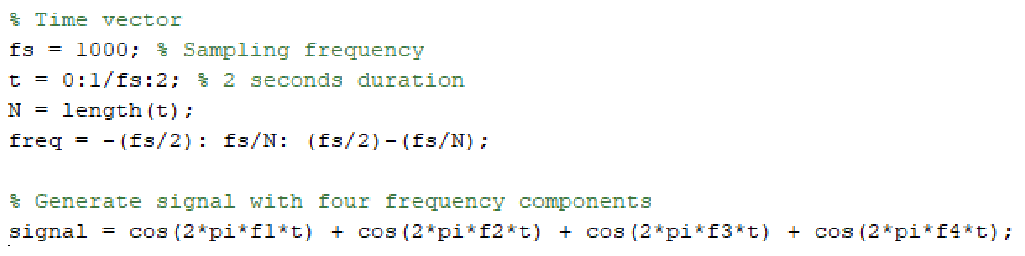
Here, L, M, N are the last three digits of your student ID. XXXXXX160.

1. f1 = 20 * (1 + 1): `f1` is calculated by multiplying 20 by the result of the expression (1 + 1). This means “f1” is twice the value of 20, which is 40.
2. f2 = 13 * (6 + 2): `f2` is calculated by multiplying 13 by the result of the expression (6 + 2). This means “f2” is 13 times the value of 8, which is 104.
3. f3 = 18 * (0 + 3): `f3` is calculated by multiplying 18 by the result of the expression (0 + 3). This means “f3”” is 18 times the value of 3, which is 54.
4. f4 = 10 * abs(1 + 6 - 0 + 4): `f4` is calculated by finding the absolute value of the expression (1 + 6 - 0 + 4) and then multiplying it by 10. The absolute value ensures that the result is non-negative. So, “f4” is 10 times the absolute value of 11, which is 110.



```
% Frequency components
f1 = 20 * (1 + 1);
f2 = 13 * (6 + 2);
f3 = 18 * (0 + 3);
f4 = 10 * abs(1 + 6 - 0 + 4);
```

5. The definition of the sampling frequency of the signal is 1000 Hz, which is defined by the expression "fs = 1000; % Sampling frequency."
6. is equal to 0:1/fs:2; the period of 2 seconds: It generates a time vector `t` that extends from 0 to 2 seconds and has a sample interval of `1/fs`, where `fs` is the sampling frequency. That the signal will be sampled at a rate of one thousand points per second for a total period of two seconds is what this indicates.
7. The function `N = length(t);` is assigned to the variable `N` once it has calculated the number of samples that are included inside the time vector.
8. the frequency is equal to -(fs/2): fs/N: (fs/2) -(fs/N); With a step size of `fs/N`, it generates a frequency vector `freq` that represents frequencies ranging from the negative half of the sampling frequency (`-fs/2`) to the positive half of the sampling frequency (`fs/2`). In a subsequent step, frequency-domain analysis will be performed using this vector.
9. The equation for the signal is as follows: signal = cos(2*pi*f1*t) + cos(2*pi*f2*t) + cos(2*pi*f3*t) + cos(2*pi*f4*t); Over the course of the time vector `t`, it creates a signal by merging four cosine waves with frequencies `f1`, `f2`, `f3`, and `f4`. The aggregate of these four cosine components is the signal that is produced as a consequence.



```
% Time vector
fs = 1000; % Sampling frequency
t = 0:1/fs:2; % 2 seconds duration
N = length(t);
freq = -(fs/2): fs/N: (fs/2)-(fs/N);

% Generate signal with four frequency components
signal = cos(2*pi*f1*t) + cos(2*pi*f2*t) + cos(2*pi*f3*t) + cos(2*pi*f4*t);
```

10. Fs = fs; It sets the sampling frequency (`Fs`) equal to the previously defined sampling frequency `fs`. This is the frequency at which the signal is sampled.

11. Ts1 = 1/Fs; It calculates the sampling interval (`Ts1`) by taking the reciprocal of the sampling frequency (`Fs`). The sampling interval represents the time between consecutive samples in the signal.
12. bw = 10; It sets the bandwidth (`bw`) of the notch filter to 10 Hz. The bandwidth defines the range of frequencies around the notch frequency that will be suppressed by the notch filter. In other words, the filter will attenuate frequencies within this bandwidth around the specified notch frequency.

These variables (`Fs`, `Ts1`, and `bw`) will likely be used in the design of a notch filter to remove a specific frequency component from a signal(f2), but the actual filter design and implementation are not provided in this snippet. The values assigned to `Fs`, `Ts1`, and `bw` are crucial parameters for setting up the notch filter.

```
% Design a digital notch filter to remove the chosen frequency component
Fs = fs;
Ts1 = 1/Fs;
bw = 10;
```

13. After that, we used the standard pole calculation and zero calculation methods and applied the filter on the signal as shown in our Lab Manual.

```
% Zero-calculation
Fr = f2;
thetal = 2*pi*(Fr/Fs);
z = exp(1j*thetal);
zp = exp(-1j*thetal);

% Pole-calculation
Fp1 = f2 - 5;
Fp2 = f2 + 5;
thetap1 = 2*pi*(Fp1/Fs);
thetap2 = 2*pi*(Fp2/Fs);
r = 1-(bw/Fs)*pi;
p1 = r*exp(1j*thetap1);
p1p = r*exp(-1j*thetap1);
p2 = r*exp(1j*thetap2);
p2p = r*exp(-1j*thetap2);
num = poly([z,zp]);
den = poly([p1, p1p, p2, p2p]);
[h, f] = freqz(num, den, 256, Fs);

% Apply the filter to the signal
filtered_signal = filter(num, den, signal);
```

14. Now as per requested on our objective, we plot all the data in respective manner using necessary plot data.

```
% Plotting
% Time domain and Frequency domain plots of generated signal
figure(1);
subplot(2,1,1);
plot(t, signal);
title('Generated Signal (Time Domain)');
subplot(2,1,2);
plot(freq, abs(fftshift(fft(signal))));
title('Generated Signal (Frequency Domain)');

% Impulse response and frequency response of the designed filter
figure(2);
subplot(2,1,1);
impz(num, den);
title('Impulse Response of the Filter');
subplot(2,1,2);
freqz(num, den, fs);
title('Frequency Response of the Filter');
```

```
% Time domain and Frequency domain plots of the filtered signal
figure(3);
subplot(2,1,1);
plot(t, filtered_signal);
title('Filtered Signal (Time Domain)');
subplot(2,1,2);
plot(freq, abs(fftshift(fft(filtered_signal))));
title('Filtered Signal (Frequency Domain)');
```

Outputs:

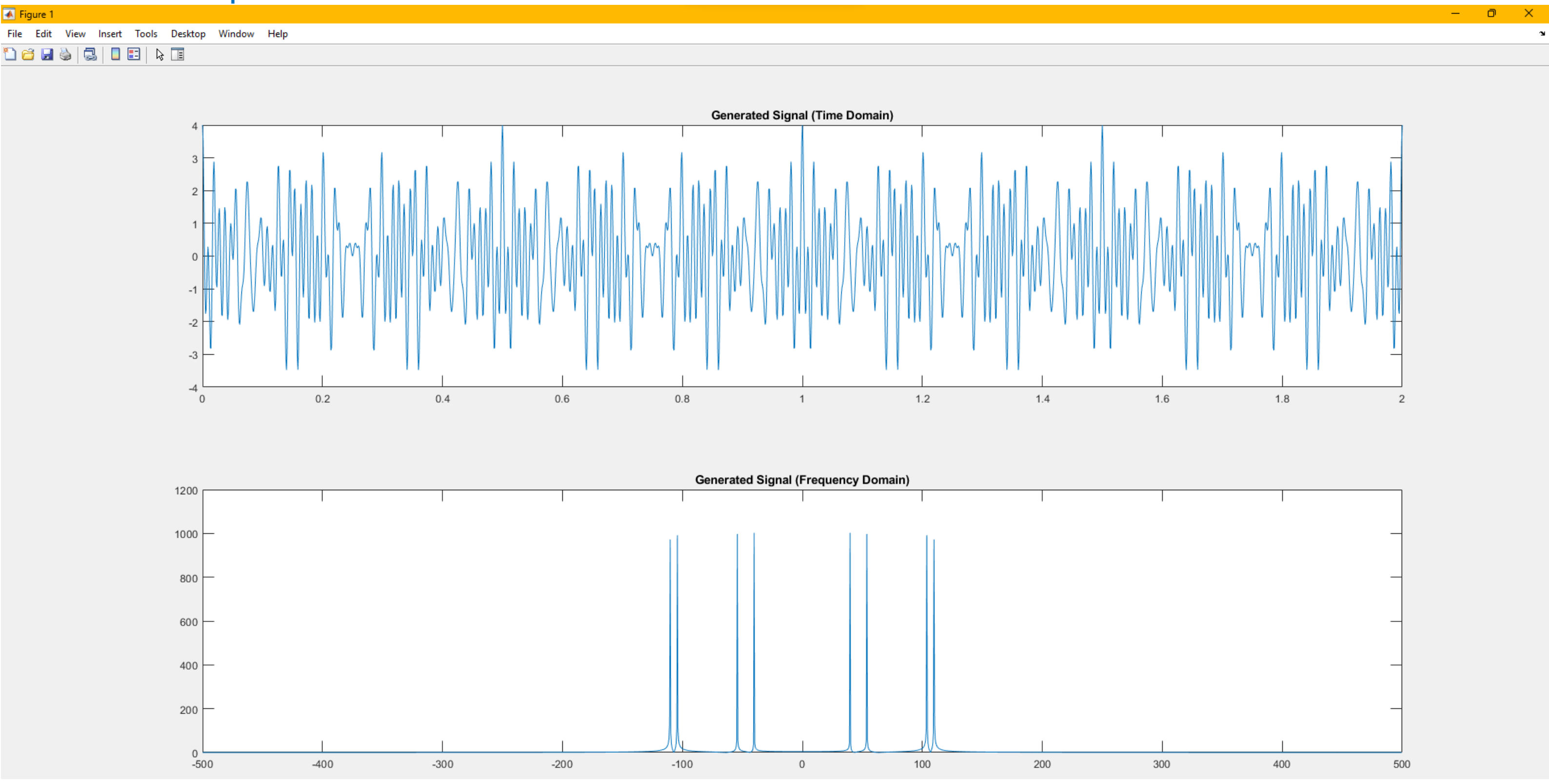


Figure 1
File Edit View Insert Tools Desktop Window Help
Generated Signal (Time Domain)
Generated Signal (Frequency Domain)

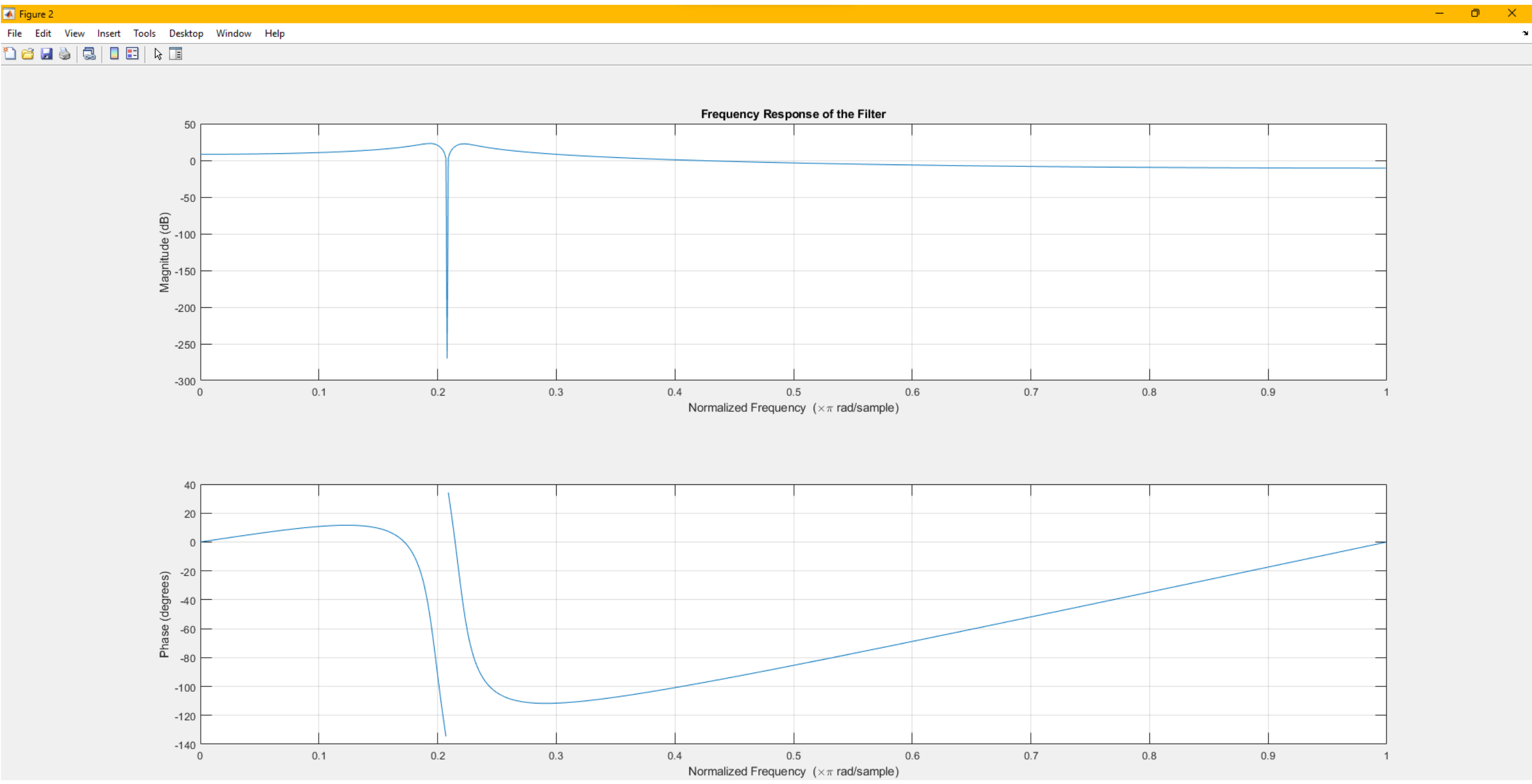


Figure 2
File Edit View Insert Tools Desktop Window Help
Frequency Response of the Filter
Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)

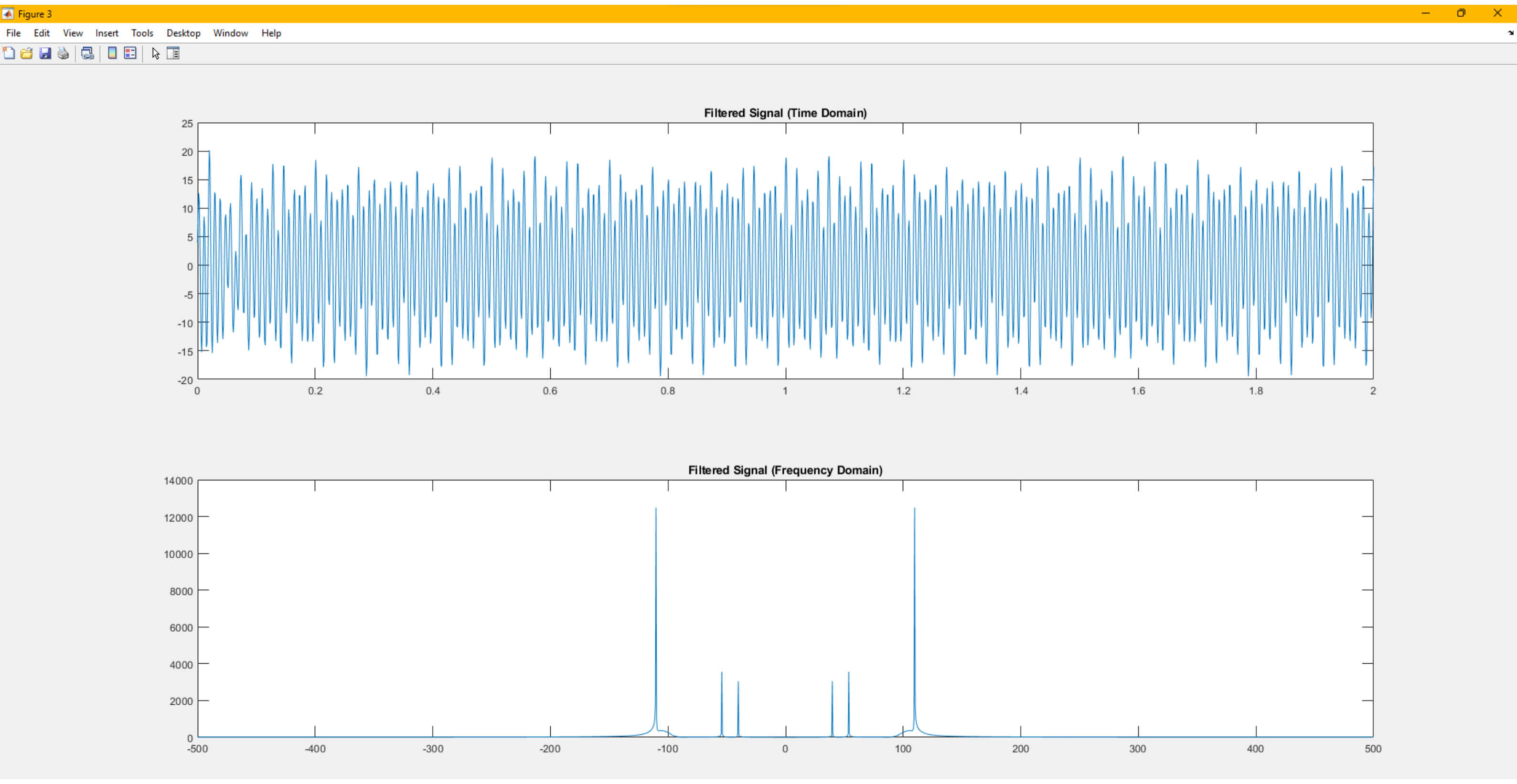


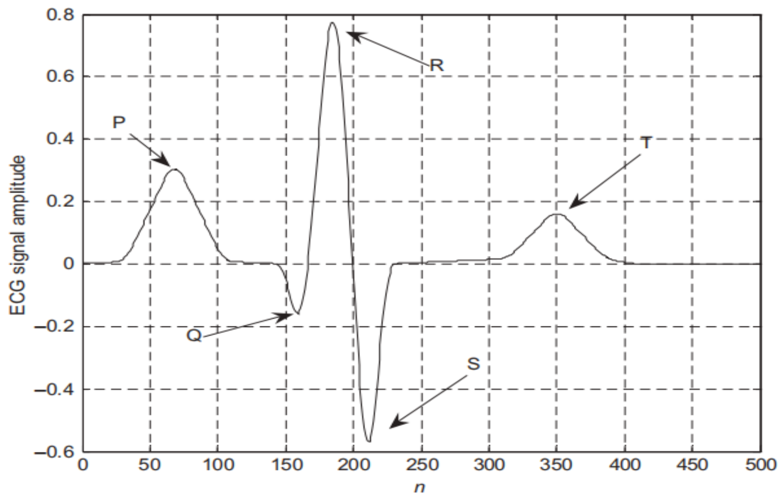
Figure 3
File Edit View Insert Tools Desktop Window Help
Filtered Signal (Time Domain)
Filtered Signal (Frequency Domain)

Discussion:

Here we can see that from the generated signal from figure (1) to the filtered signal in Figure (3), we see f2 has been deducted using our designed filter. And our designed filter was successful.

NOISE REMOVING FROM ECG SIGNAL

The ECG is a small electrical signal captured from an ECG sensor. The ECG signal is produced by the activity of the human heart, thus can be used for heart rate detection, fetal monitoring, and diagnostic purposes.
The single pulse of the ECG is depicted in the figure, which shows that an ECG signal is characterized by five peaks and valleys The highest positive wave is the R wave. Shortly before and after the R wave are negative waves called Q wave and S wave. The P wave comes before the Q wave, while the T wave comes after the S wave. The Q, R, and S waves together are called the QRS complex. The properties of the QRS complex, with its rate of occurrence and times, highs, and widths, provide information to cardiologists concerning various pathological conditions of the heart



IN ECG SIGNAL 3 TYPES OF NOISE CAN BE FOUND

i. Powerline interference noise
ii. Baseline wander noise
iii. High-frequency noise

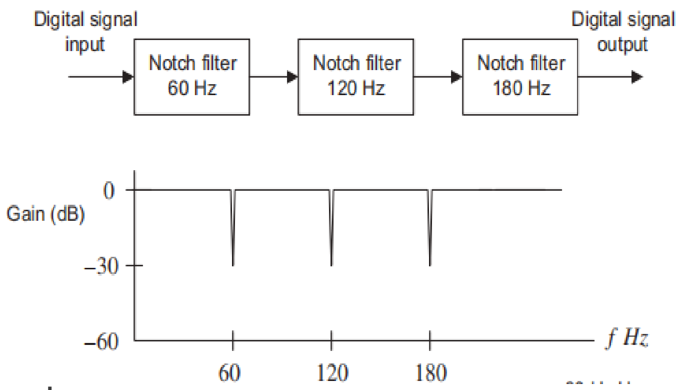
Powerline interference noise:

Powerline interference noise, also known as mains hum or electrical interference, is a type of noise that can affect electronic signals and is commonly encountered in audio and biomedical signal processing, including electrocardiogram (ECG) and electromyogram (EMG) recordings. This interference is caused by the presence of alternating current (AC) power lines, typically operating at 50 Hz or 60 Hz, depending on the region.

- In regions where the power grid operates at 50 Hz, the interference frequency is typically around 50 Hz. So in Bangladesh it is 50 hz.
- Harmonics of the fundamental frequency (e.g., 100 Hz, 150 Hz, etc.) may also be present in the signal.

WE WILL USE NOTCH FILTER TO REMOVE THIS NOISE. WE WILL USE THREE NOTCH FILTER IN TOTAL. ONE TO REMOVE THE FUNDAMENTAL HARMONIC 50 HZ. ANOTHER ONE TO REMOVE THE SECOND HARMONICS AT 100 HZ.AND ANOTHER ONE TO REMOVE THE THIRD HARMONIC AT 150 HZ.

This picture is given for the power line interference of 60 hz.



Baseline wander noise:

Baseline wander is a type of noise or interference that can affect physiological signals, such as electrocardiogram (ECG) signals. It refers to the low-frequency variations in the baseline of the signal, which can be caused by various factors. Baseline wander can make it challenging to accurately analyze and interpret the underlying physiological information in the signal.

Causes of Baseline Wander:

1.Respiration Artifacts: Changes in chest impedance during respiration can lead to baseline wander in ECG signals. This is particularly common in ambulatory recordings.

2.Body Movement: Patient movement or changes in body position can introduce baseline wander. For example, shifting body weight or adjusting posture can affect the baseline of the signal.

3.Muscle Activity: Electromyographic (EMG) activity from muscle contractions, especially in non-cardiac muscles, can contribute to baseline wander.

4.Electrode Contact Issues: Poor electrode-skin contact or movement of electrodes can introduce baseline variations.

High Frequency Noise:

High-frequency noise refers to rapid and short-duration variations in a signal that occur at frequencies higher than the frequency of the primary signal of interest. This type of noise can be particularly problematic in electronic signals, including physiological signals like electrocardiograms (ECG), where it can obscure important details and interfere with accurate signal analysis

Causes of High Frequency Noise:

Electromagnetic Interference (EMI): EMI from electronic devices, power lines, or other electronic equipment can introduce high-frequency noise into signals.

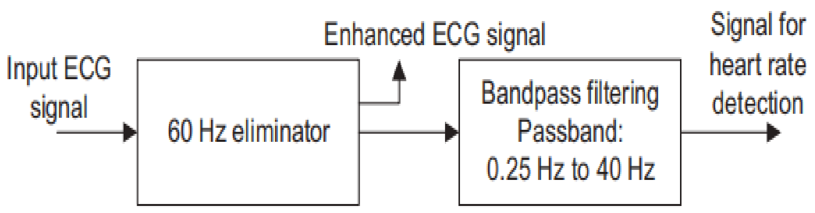
Radio Frequency Interference (RFI): Signals from radio frequency sources, such as radio or television broadcasts, can cause interference in electronic signals.

Switching Noise: Rapid switching of electronic components, such as digital circuits or power supplies, can generate high-frequency noise.

Ambient Electrical Noise: Environmental factors, such as electrical appliances or fluorescent lights, can contribute to high-frequency noise

IN ORDER TO REMOVE THE HIGH FREQUENCY AND BASELINE WANDERER NOISE WE WILL USE A PASSBAND FILTER WITH LOWER CUTOFF FREQ .25HZ AND HIGHER CUTOFF FREQ OF 40HZ.

Considering the lowest heart rate as 30 beats per minute (BPM), the corresponding frequency is 0.5 Hz. To ensure that the bandpass filter retains the relevant heart rate information, a lower cutoff frequency of 0.25 Hz is chosen. The upper cutoff frequency is set at 40 Hz to eliminate muscle noise or any higher frequency term. The range chosen is from 0.25 to 40 Hz.


Input ECG signal
60 Hz eliminator
Enhanced ECG signal
Bandpass filtering Passband: 0.25 Hz to 40 Hz
Signal for heart rate detection

The ECG signal after bandpass filtering with a passband from 0.25 to 40 Hz is no longer valid for general ECG applications, since the original ECG signal occupies the frequency range from 0.01 to 250 Hz (diagnostic-quality ECG). The enhanced ECG signal from the 60-Hz hum eliminator can serve for general ECG signal analysis.

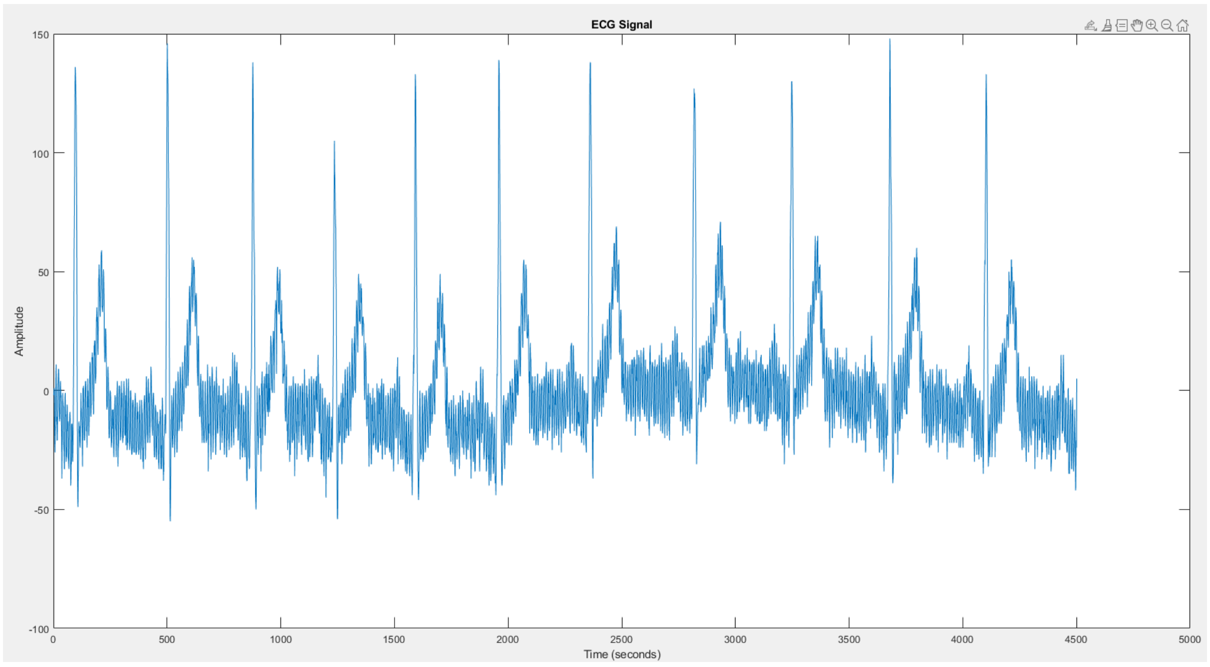
All the information has been collected from the book: Digital Signal Processing (Fundamentals and Applications by Li-Tang

Part-2:

Choose the data of the row number (1+0) = 1 and 5*(6+0) = 30 of the given data set. We use the row number as 1 then again as 30. So, for row number = 1:

```
x = load('combine.mat');
w = x.val(1, 500:5000);
t = linspace(0,20,4501);

figure(1)
plot(w);
title('ECG Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');
```

1. We load the combine.mat dataset in the variable x. Then we set the stage for w. And take t as the length since 500 to 5000 are 4501 steps.
2. Then we plot the loaded ECG Signal in figure 1.



3. We make Notch filters of the 50hz, 100hz, 150hz eliminators. And for that first we set the variables for the frequency that is 500hz (sampling) and bandwidth (bw) = 5.

```
%50 Hz eliminator
Fs = 500;
bw =5;
```

```
%100 hz eliminator
Fs = 500;
Tsl = 1/Fs;
bw =5;
```

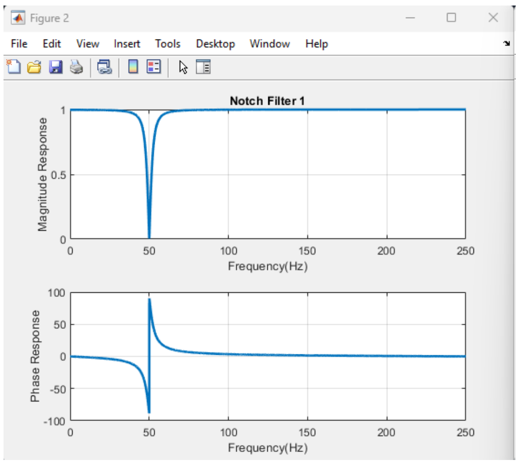
```
%150 hz eliminator
Fs = 500;
Tsl = 1/Fs;
bw =5;
```

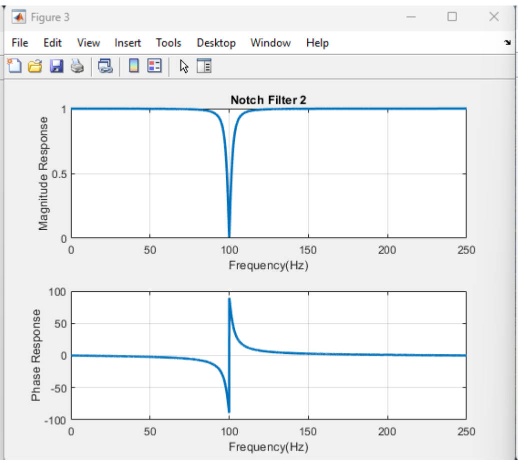
4. Then we use Pole, zero calculations and plot their magnitude and phase response curves on the figure 2, figure 3 and figure 4 respectively.

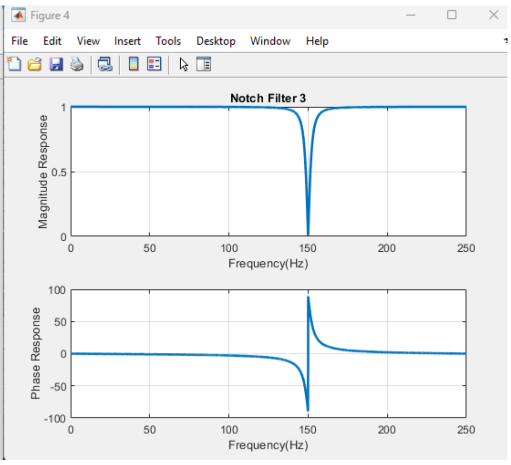
```
%Zero-calculation
Fr1 =50;
theta1 = 2*pi*(Fr1/Fs);
z1 = exp(1j*theta1); z2 = exp(-1j*theta1);

%Pole-calculation
Fp = 50;                    % passband frequency
thetap = 2*pi*(Fp/Fs); r = 1-(bw/Fs)*pi;
p1 = r*exp(1j*thetap); p2 = r*exp(-1j*thetap);

%Numerator vector & denominator vector
num0 = poly([z1, z2]);
den0 = poly([p1, p2]);
[h0, f] = freqz(num0, den0, 4501, Fs);

figure(2);
subplot(211);
plot(f, abs(h0)/max(abs(h0)), 'linewidth', 2);
grid on;
title('Notch Filter 1');
xlabel('Frequency(Hz)');
ylabel('Magnitude Response');
subplot(212);
plot(f, angle(h0)*180/pi, 'linewidth', 2);
grid on;
xlabel('Frequency(Hz)');
ylabel('Phase Response');
```

```
%Zero-calculation
Fr1 =100;
theta1 = 2*pi*(Fr1/Fs);
z1 = exp(1j*theta1); z2 = exp(-1j*theta1);

%Pole-calculation
Fp = 100;
thetap = 2*pi*(Fp/Fs); r = 1-(bw/Fs)*pi;
p1 = r*exp(1j*thetap); p2 = r*exp(-1j*thetap);

%Numerator vector & denominator vector
num1 = poly([z1, z2]);
den1 = poly([p1, p2]);
[h1, f] = freqz(num1, den1, 4501, Fs);

figure(3);
subplot(211);
plot(f, abs(h1)/max(abs(h1)), 'linewidth', 2);
grid on;
title('Notch Filter 2');
xlabel('Frequency(Hz)');
ylabel('Magnitude Response');
subplot(212);
plot(f, angle(h1)*180/pi, 'linewidth', 2);
grid on;
xlabel('Frequency(Hz)');
ylabel('Phase Response');
```

```
%Zero-calculation
Fr1 =150;
theta1 = 2*pi*(Fr1/Fs);
z1 = exp(1j*theta1);
z2 = exp(-1j*theta1);

%Pole-calculation
Fp = 150;                   % passband frequency
thetap = 2*pi*(Fp/Fs);
r = 1-(bw/Fs)*pi;
p1 = r*exp(1j*thetap);
p2 = r*exp(-1j*thetap);

%Numerator vector & denominator vector
num2 = poly([z1, z2]);
den2 = poly([p1, p2]);
[h2, f] = freqz(num2, den2, 4501, Fs);

figure(4);
subplot(211);
plot(f, abs(h2)/max(abs(h2)), 'linewidth', 2);
grid on;
title('Notch Filter 3');
xlabel('Frequency(Hz)');
ylabel('Magnitude Response');
subplot(212);
plot(f, angle(h2)*180/pi, 'linewidth', 2);
grid on;
xlabel('Frequency(Hz)');
ylabel('Phase Response');
```

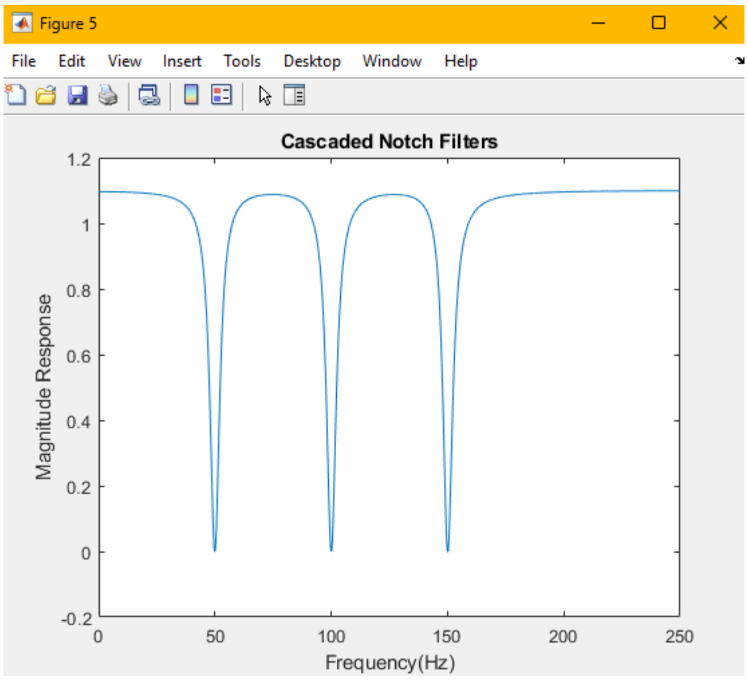




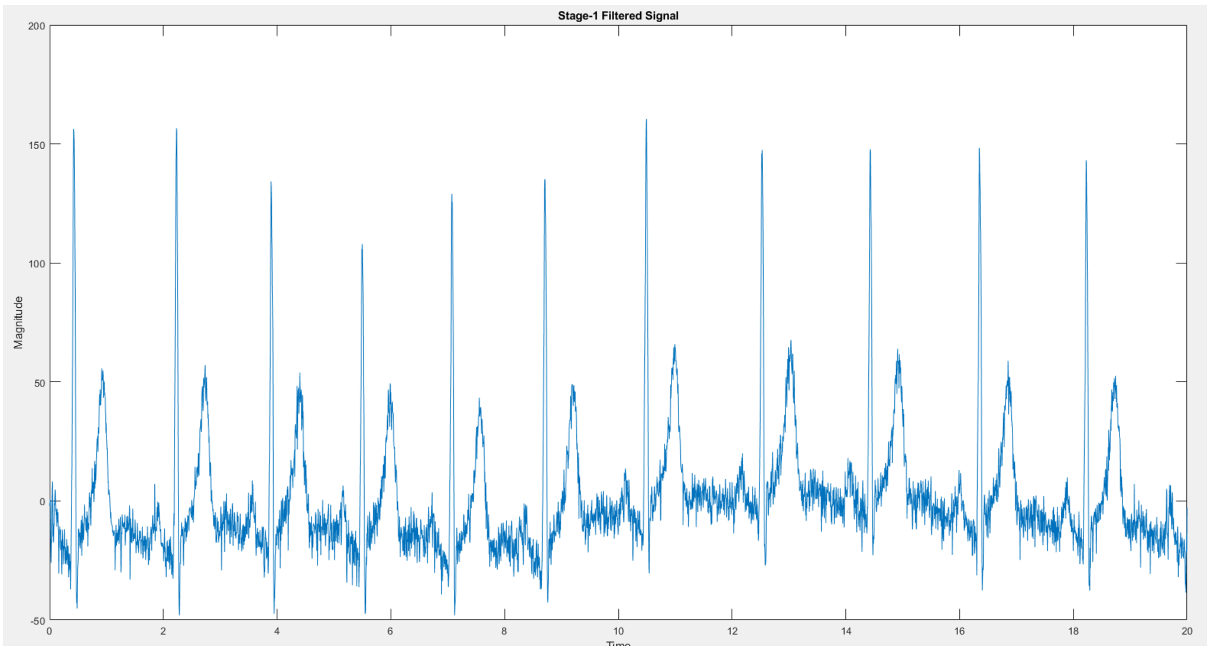


5. We make the cascaded notch filters to put them in series for applying them altogether. And plot that magnitude response in figure (5).

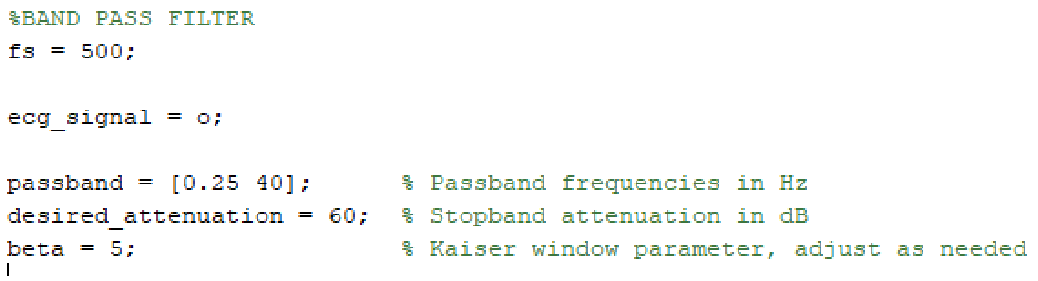
```
%Cascading
NUM1 = conv(num0, num1);
NUM = conv(NUM1, num2);

DEN1 = conv(den0, den1);
DEN = conv(DEN1, den2);

o = filter(NUM, DEN, w);

figure(5)
H = h0.*h1.*h2;
plot(f,H);
title('Cascaded Notch Filters');
xlabel('Frequency(Hz)');
ylabel('Magnitude Response');
```



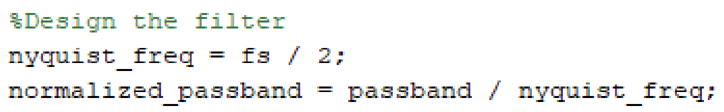
6. We now plot the signal passed through the filter in figure (6).



7. We now use the Band pass filter.



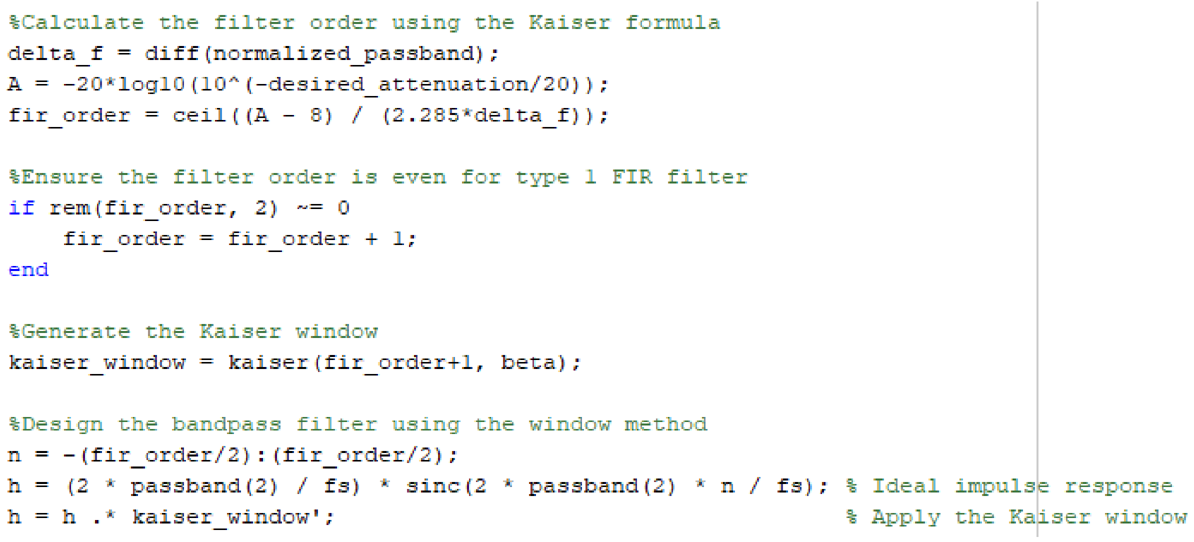
```
%BAND PASS FILTER
fs = 500;

ecg_signal = o;

passband = [0.25 40];        % Passband frequencies in Hz
desired_attenuation = 60;    % Stopband attenuation in dB
beta = 5;                    % Kaiser window parameter, adjust as needed
```



```
%Design the filter
nyquist_freq = fs / 2;
normalized_passband = passband / nyquist_freq;
```

8. The Kaiser window is used in this code to design a bandpass filter using the window method. The Kaiser window is chosen for its ability to control the trade-off between the main lobe width and the side lobe levels in the frequency domain.

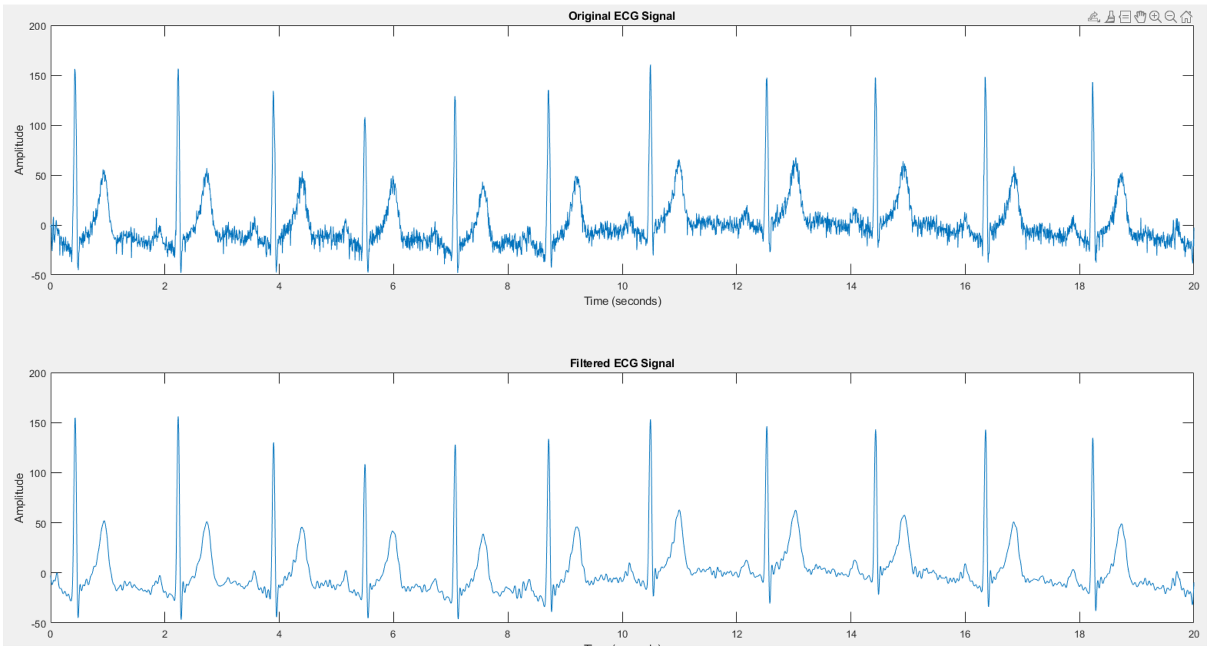
i. `delta_f = diff(normalized_passband);`: It calculates the width of the passband in the normalized frequency domain.
ii. `A = -20*log10(10^(-desired_attenuation/20));`: It calculates the desired stopband attenuation in decibels.
iii. `fir_order = ceil((A - 8) / (2.285*delta_f));`: It uses the Kaiser formula to calculate the filter order based on the desired attenuation and passband width.
iv. `if rem(fir_order, 2) ~= 0...`: It ensures that the filter order is even, as the type 1 FIR filter requires an even order.
v. `kaiser_window = kaiser(fir_order+1, beta);`: It generates a Kaiser window of length `fir_order+1` with a shape parameter `beta`. The Kaiser window will be used to window the ideal impulse response.
vi. `n = -(fir_order/2):(fir_order/2);`: It defines a time vector for the ideal impulse response.
vii. `h = (2 * passband(2) / fs) * sinc(2 * passband(2) * n / fs);`: It calculates the ideal impulse response of the bandpass filter.
viii. `h = h .* kaiser_window';`: It applies the Kaiser window to the ideal impulse response, effectively windowing the response to obtain the final filter coefficients.



```
%Calculate the filter order using the Kaiser formula
delta_f = diff(normalized_passband);
A = -20*log10(10^(-desired_attenuation/20));
fir_order = ceil((A - 8) / (2.285*delta_f));

%Ensure the filter order is even for type 1 FIR filter
if rem(fir_order, 2) ~= 0
    fir_order = fir_order + 1;
end

%Generate the Kaiser window
kaiser_window = kaiser(fir_order+1, beta);

%Design the bandpass filter using the window method
n = -(fir_order/2):(fir_order/2);
h = (2 * passband(2) / fs) * sinc(2 * passband(2) * n / fs); % Ideal impulse response
h = h .* kaiser_window';                                      % Apply the Kaiser window
```

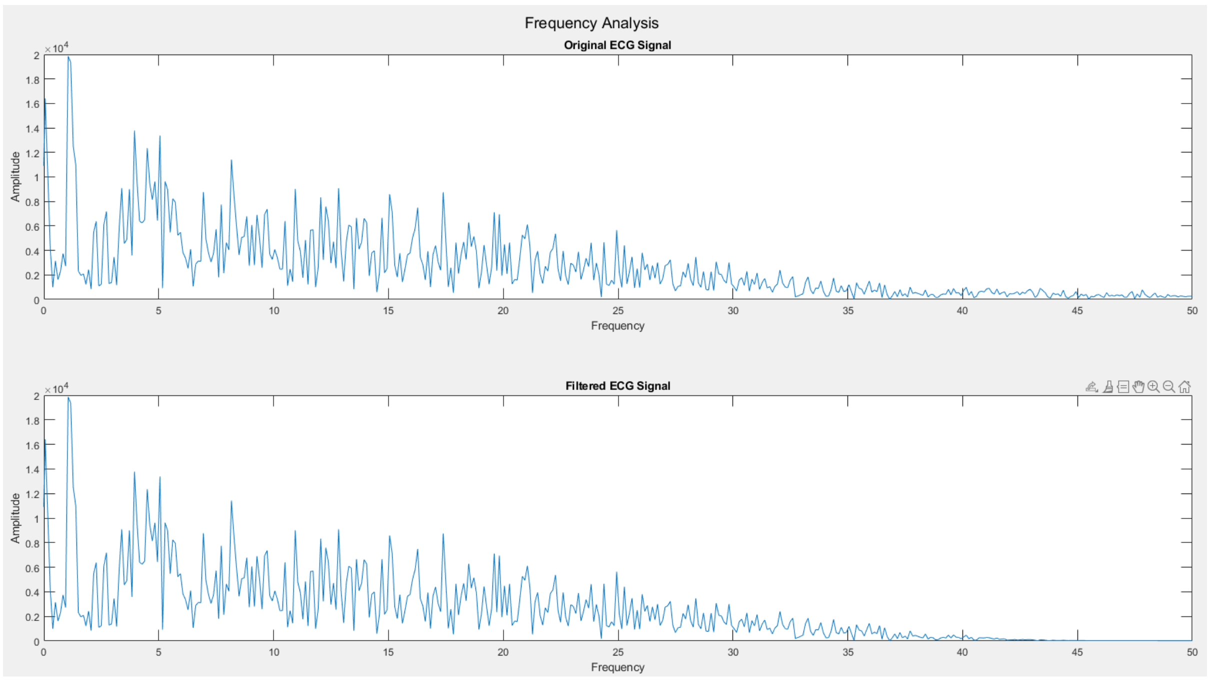
9. We now apply the designed filter on our ECG signal and plot the response in figure(7).

```
%Apply the filter to the ECG signal
filtered_ecg = conv(ecg_signal, h, 'same');

%Plot original and filtered signals
figure(7);
subplot(2,1,1);
plot(t, ecg_signal);
title('Original ECG Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');

subplot(2,1,2);
plot(t, filtered_ecg);
title('Filtered ECG Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');
```



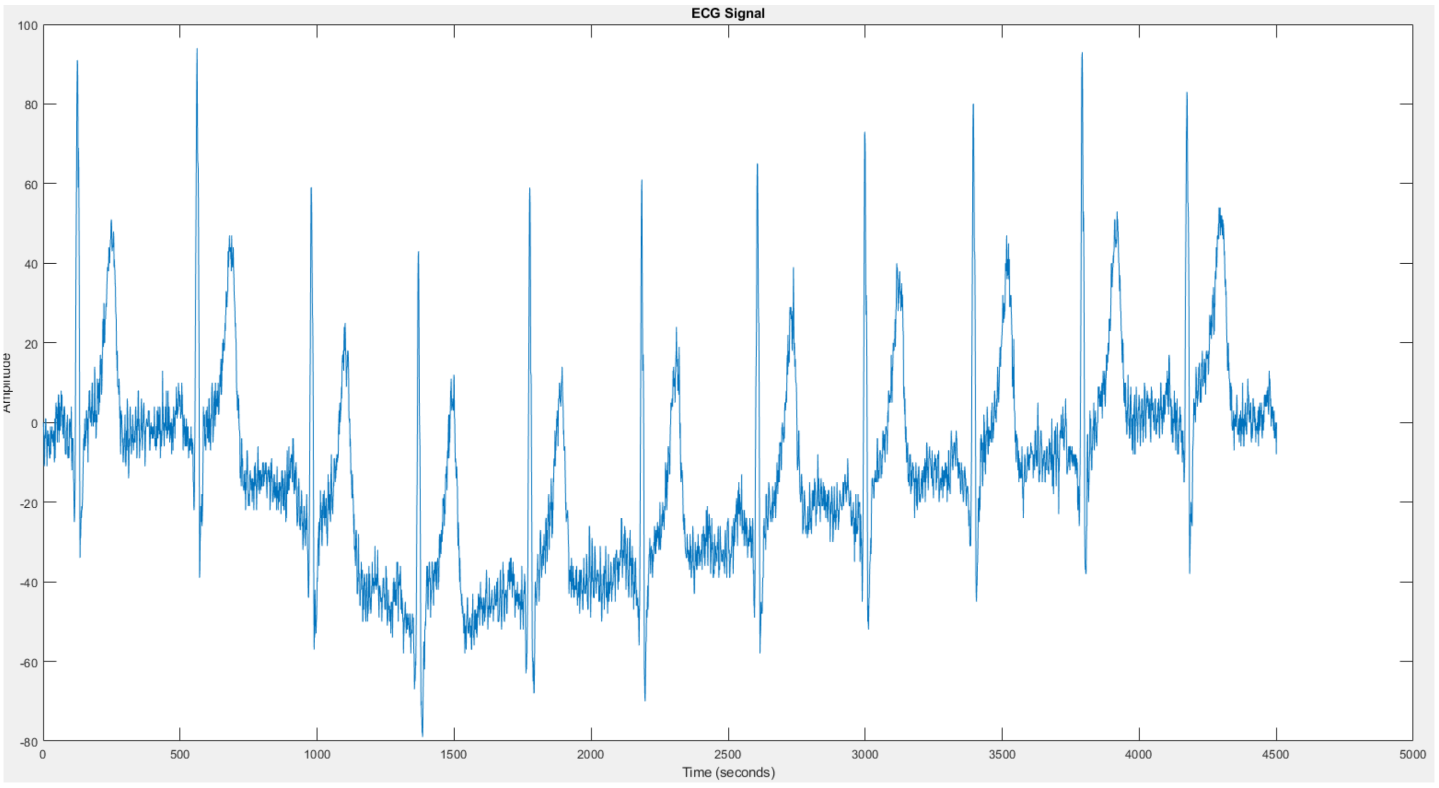
10. Now we do frequency analysis using fft in figure (8).

```
%Frequency Plotting
freq = -fs/2: fs/4501: (fs/2)-(fs/4501);
figure(8);
subplot(2,1,1);
plot(freq, abs(fftshift(fft(ecg_signal))));
title('Original ECG Signal');
xlabel('Frequency');
ylabel('Amplitude');
xlim([0,50]);
subplot(2,1,2);
plot(freq, abs(fftshift(fft(filtered_ecg))));
title('Filtered ECG Signal');
xlabel('Frequency');
ylabel('Amplitude');
xlim([0,50]);
sgtitle('Frequency Analysis')
```

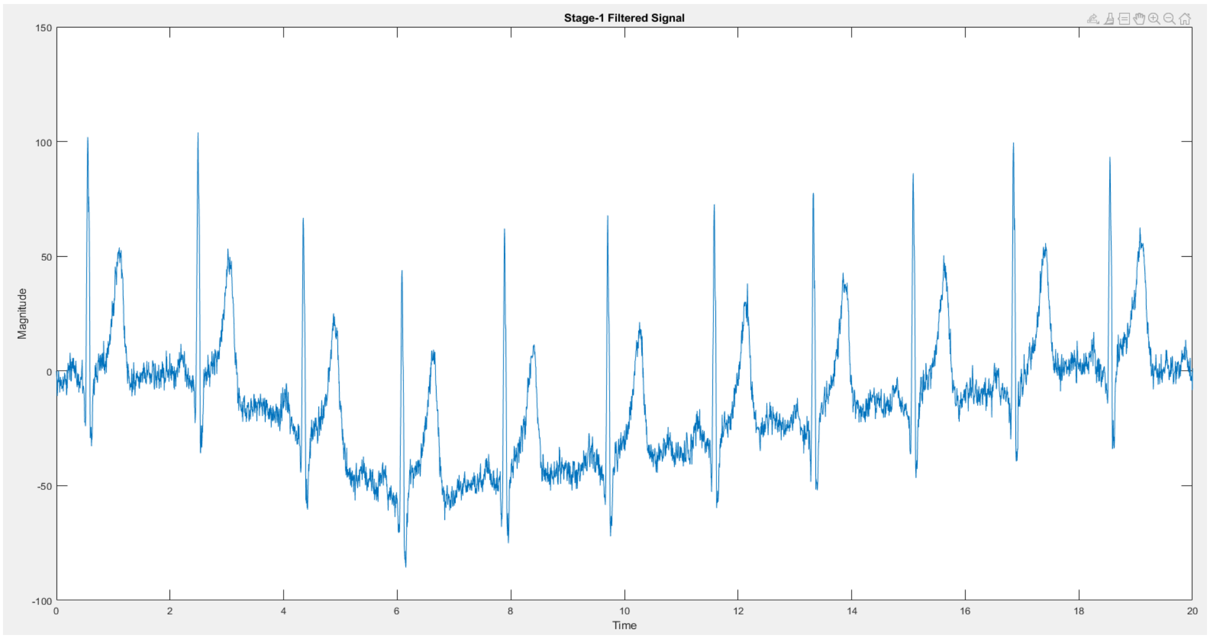


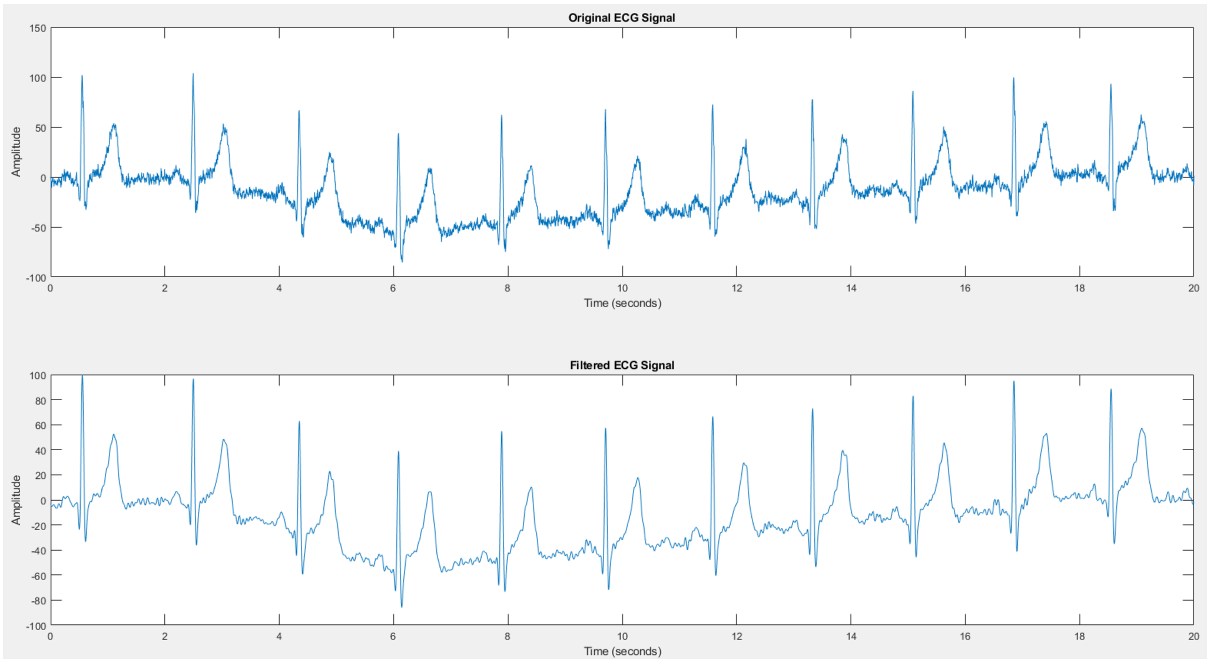
11. Now using the row number = 30, we do all of it similarly using similar logic.

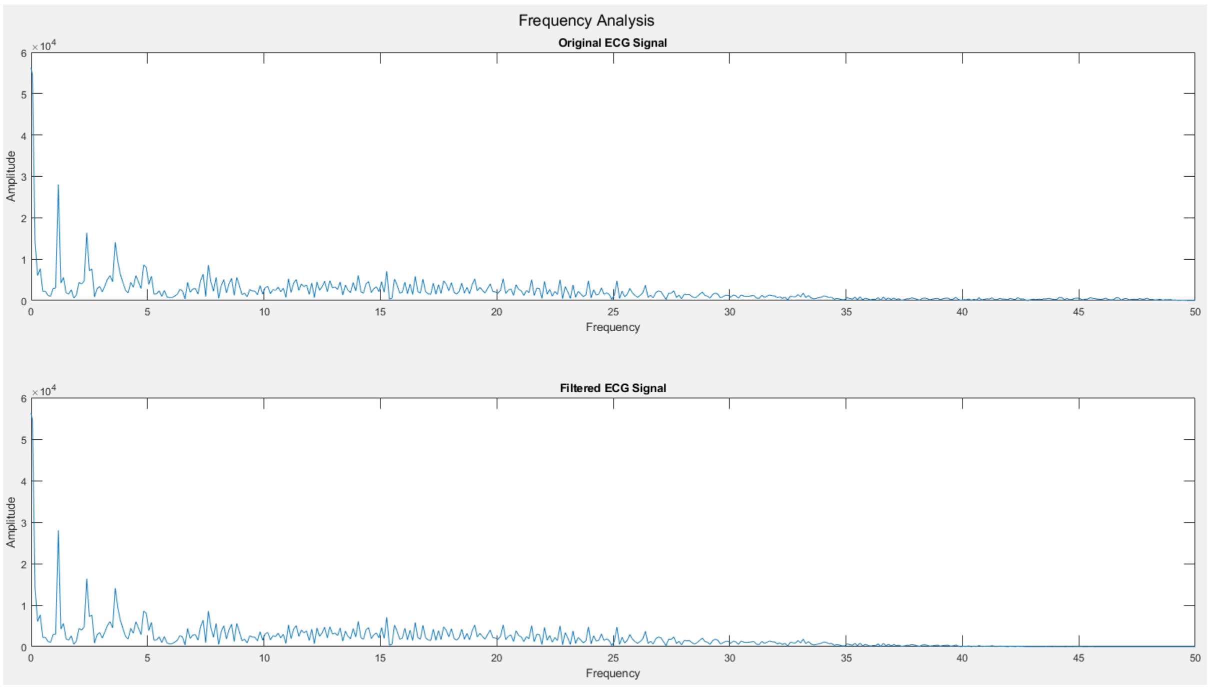
```
% For another value of row that is 30
x = load('combine.mat');
w = x.val(30, 500:5000);
t = linspace(0,20,4501);

figure(9)
plot(w);
title('ECG Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');
```



12. Now finally we plot its necessary curves in figure 10, 11, 12 respectively.







Results and Discussion:

Data:

The objective of this experimental investigation was to improve the quality of an electrocardiogram (ECG) signal by addressing the problem of unwanted noise. This was accomplished via the experiment. As the basis for our inquiry, the dataset that was derived from electrocardiogram recordings served as the cornerstone. The raw electrocardiogram (ECG) data was put into the MATLAB environment, and early analysis indicated the existence of a unique frequency component, which is most likely indicative of noise interference. In order to be successful in following noise reduction attempts, it was essential to recognize and comprehend this undesired frequency component.

Data Analysis:

A thorough data analysis strategy, in particular one that included frequency-domain investigation, was essential to the fulfillment of our approach's potential for success. Several calculations were carried out in order to ascertain the characteristics of the notch filter, which included the filter order, the notch frequency, and the bandwidth. For the purpose of computing the filter order, the Kaiser formula was used after the necessary stopband attenuation and passband width were taken into consideration. We were able to see the properties of the noise thanks to the frequency vector, which ultimately led to a better-informed design of the notch filter. When it came to structuring the impulse response of the filter, the Kaiser window, which is well-known for its adaptability in balancing the width of the main lobe and the levels of the side lobes, played a significant role.

We were able to develop a more individualized strategy for noise reduction thanks to our application of frequency analysis. The undesired frequency component in the electrocardiogram signal was efficiently targeted and suppressed by the use of the notch filter, which was created with accuracy. A frequency-domain knowledge is essential for building filters for biomedical signal processing, and this approach shows the significance of this understanding.

Conclusion:

Through the use of a notch filter, we were able to successfully accomplish our goal of lowering the amount of noise present in the electrocardiogram (ECG) data. A refined electrocardiogram (ECG) signal was achieved by the combination of careful data analysis, which included frequency analysis, and the use of a filter that was thoughtfully constructed at the same time. In order to effectively suppress the undesired frequency component, the notch filter, with its characteristics that were given, proved to be successful. The findings of this work highlight the necessity of using modern signal processing methods, particularly in the field of biomedicine, where the precision and dependability of physiological signal interpretation are of the utmost importance. It is possible that future research paths may investigate the wider applicability of such approaches and further evaluate their effectiveness in a variety of biological signal processing settings.