GROUPWORK

Coursework Declaration and Feedback Form

Course Code:	**ENG 5027**	Course Name:	**Digital Signal Processing**		
Course Co-Ordinator:	**Scott Watson**	Mentor:		Group Number Lab Group Tutorial Group	**9**
Title of Assignment:	**Assignment 2 (Fourier Transform – FFT)**				
Date of Submission:	**18-11-2024**				
Declaration of Originality and Submission Information		*I affirm that this submission is my own / the groups original work in accordance with the University of Glasgow Regulations and the School of Engineering Requirements* ***All students should sign this form***			
Student Name:	**Basav Prasad (21961731)**	Student Name:	**Cem Cetinkaya (2961798)**	Student Name:	**Mingxuan Sun (3025864)**
Student Name:	**Syafiq Fathullah (2830739)**	Student Name:	**Kabir Sani Muhammad (2973267)**	Student Name:	**Abdulrazaq Muhammad Sani (2962438)**

Feedback from Lecturer to Student – to be completed by Lecturer or Demonstrator	
Grade Awarded: Feedback (as appropriate to the coursework which was assessed)	
Lecturer/Demonstrator	Date returned to the Teaching Office

Assignment 2 (Finite Impulse Response – FIR)

November 18, 2024

Assignment report submitted in partial fulfilment of the requirements for the credit of course -

Digital Signal Processing

Introduction

This report provides the design and implementation of FIR (Finite Impulse Response) filters applied to ECG signal processing using the generic FIR filter, adaptive FIR and matched FIR filter methods. The main objectives were to filter the ECG signal, remove noise such as 50 Hz powerline interference, and obtain a clean output. Efforts were also made to optimize the filter for efficiency. The methods, parameters, and justifications for various choices are explained in detail, along with observations from the filtering results.

Methods

- **Task 1: Coefficient Calculation and FIR Filter Design**

Determining Filter Order (M) and Frequency Resolution: To design the FIR filter, the filter order M was determined based on the desired frequency resolution Δf. This is the cut-off frequency for the highpass filter to remove the DC component of the noisy signal. Both 0.5 Hz and 1 Hz resolutions were tested. **0.5 Hz Resolution**, in theory, was supposed to give higher precision. However, not only was there not a noticeable difference both 0.5 and 1 Hz, but it also introduced a much longer delay, resulting in two fewer pulses than the original signal and information loss. For this reason, a **1 Hz Resolution** was used in this design.

- The frequency resolution Δf is calculated using the equation:

$$\Delta f = \frac{fs}{M}$$

To handle division by zero when $n = 0$, different coefficient expressions were used for the stopband and high-pass filters. For the stop-band filter, the coefficient at $n = 0$ is defined as $1-(2f_1-2f_2)$, while for the high-pass filter, it is $1-2f_{highpass}$. It should be noted that the normalized frequencies were used inside these expressions.

Single Pulse Limitation to M - For the single pulse graph, the signal length was limited to M, the number of filter taps. In FIR filtering, this limitation ensures that only a finite segment of the signal (up to M) is processed, as FIR filters have finite responses, not infinite.

For filtering out the 50 Hz noise, a stopband between 45 Hz and 55 Hz was set up, which is typical for removing powerline interference. The frequencies f_highpass, f1, and f2 represented the highpass, cutoff frequency, bandstop start frequency and bandstop stop frequency respectively. These frequencies were normalized by dividing them by the sampling rate **(f_s)** to ensure the correct response.

To remove the 50 Hz interference, a 45-55 Hz stop-band was chosen instead of a narrower range like 48-52 Hz to ensure complete noise removal. A narrower notch might miss fluctuating interference and require sharper filter responses, leading to ringing effects. The broader 45-55 Hz range effectively filters noise without compromising signal quality. Moreover, no noticeable differences between narrower and broader notches were observed, so 45-55 Hz was used to ensure effective noise removal and signal integrity.

The next step was to determine the coefficients. The FIR filter coefficients were generated using the sinc functions associated with the high-pass and band-stop filters, as specified in the task requirements, although at first the **m** method was considered.

Initially, a manual ring buffer was used to implement the FIR filter, which resulted in a computation time of 11.28 seconds. To optimize performance, **np.roll** was introduced to shift the buffer efficiently, reducing the computation time to 0.6 seconds. This enhancement makes the filter useful for real-time applications by demonstrating the substantial effect that optimized code structures have on execution time.

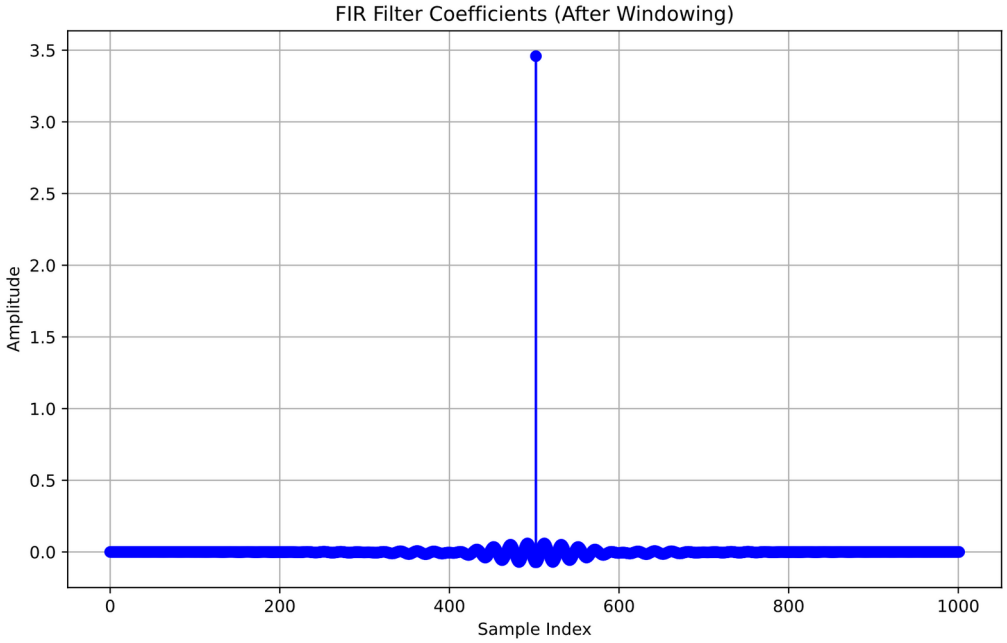

Figure 1 – Impulse response/filter coefficients

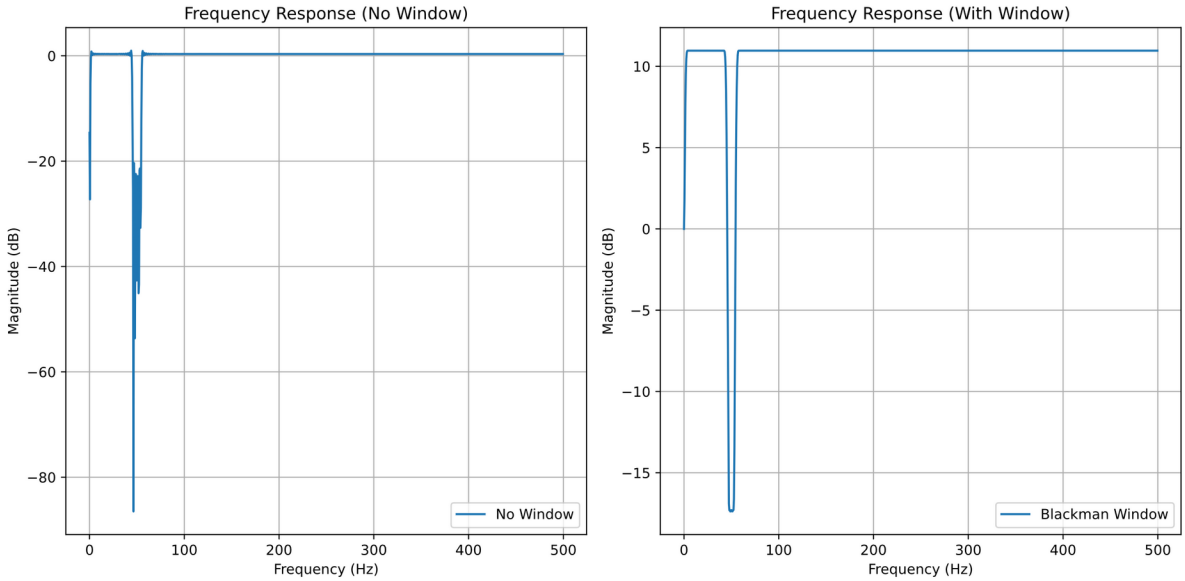

Figure 2 - comparison with and without window function

- **Task 2: Implementation of FIR Filtering**

Single-Sample Input Requirement: The FIR filter was set up to take in one sample at a time (scalar), which is necessary for real-time signal processing as FIR filters only work with causal signals. In the class, each sample gets processed individually, with a rolling buffer to store the last M

samples. This design is important because it lets the filter handle each incoming sample as it arrives, rather than working with the whole signal at once.

Filtering Mechanism - The FIR filter operates by computing a dot product between the filter coefficients and the current window of samples. If there's more than one set of coefficients, convolution is used to combine them, giving a smooth output. The number of pulses was also ensured to remain consistent, which was a good sign that the filter was doing its job without distorting the signal. Here, the filter coefficients are designed over a symmetric range from **−M/2 to M/2 + 1**. This step is what makes the filter casual.

Causality and Delay - One side effect of FIR filters is they introduce a delay proportional to M/2. This is expected, and it's worth mentioning because it affects the timing of the filtered output. The delay doesn't break causality, but it does mean the output is slightly time-shifted.

Window Function - Moreover, a window function is applied to smooth the output response. While all types of window functions are effective at removing ripples, for stop-band rejection, **the Blackman window** gave the best results. The frequency response can be seen in Figure 2, which compares the responses without and with Blackmann window. It can be observed that the ringing effect has vanished with the application of the window effect. The transition width has also improved.

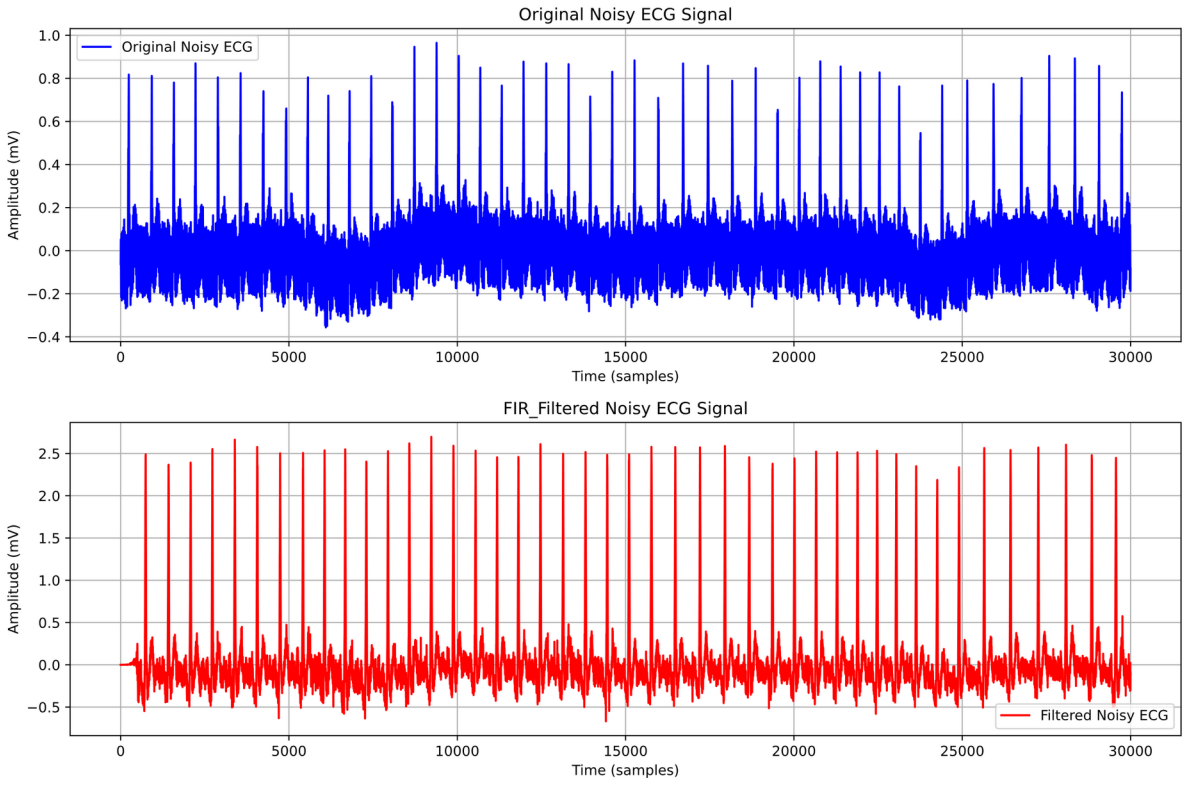

Figure 3 – Comparison between original and FIR filtered noisy ECG signal

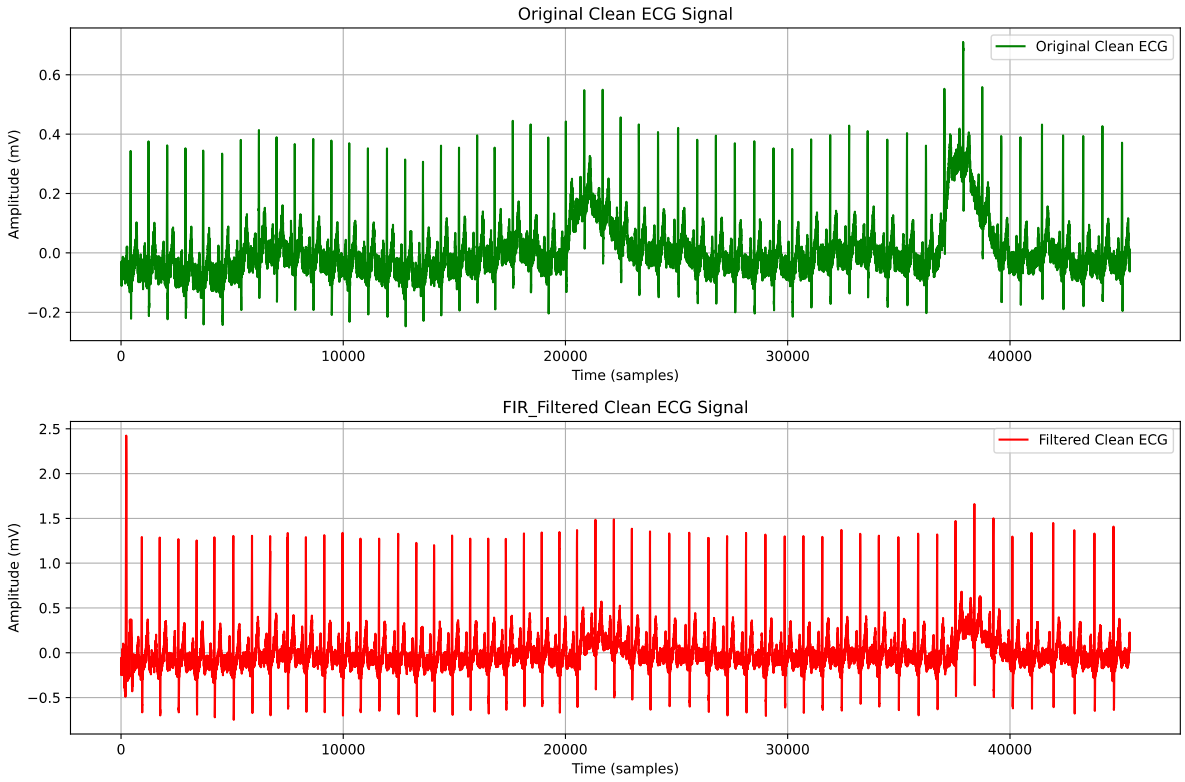

Figure 4 – Comparison between original and FIR filtered noisy ECG signal

- ## Task 3: Adaptive LMS Filter

This task aimed to use an adaptive LMS (Least Mean Squares) filter to remove the 50 Hz powerline interference and a DC offset from the noisy ECG signal. The LMS filter is a good choice here because it's designed to adjust its coefficients over time, making it adapt to the signal in real-time. The remainder of the second will summarize how it was set up and why each part was needed.

Setting Up the FIR Filter Class and LMS Update

The FIR_filter class was built with two main functions:

1. **Filter Method**: This handles the filtering by taking a new sample and using a buffer to store the last few inputs (specifically, NTAPS worth). It shifts the buffer, adds the new sample, and then calculates the filtered output with the dot product of the buffer and the filter coefficients.
2. **LMS Method**: This is where the LMS adaptation happens. It updates the filter's coefficients based on the error (the difference between the desired output and the actual filtered output). For each coefficient, it adjusts based on the error and the buffer content, using a learning rate (mu). This lets the filter "learn" from each sample, gradually adjusting to better match the noise characteristics in the signal.

Main Adaptive Filtering - The main script sets up the filter and runs it on each sample in the ECG signal to filter out noise in real time. Here, the **learning rate** is chosen as **0.001** after testing. If a relatively high value is chosen, the filter will adjust too fast and might overshoot, causing oscillations. If it's too low, the filter adjusts too slowly.

Noisy Reference Signal - A 50 Hz sine wave plus a DC offset of 0.1 is created. The 50 Hz sine wave

simulates powerline noise, while the DC offset represents a baseline shift. Initially, the code was altered in such it could work out the DC component of the noisy ECG signal and generate a noisy reference signal with the same DC component to match them for better DC removal. However, the DC component of the noisy ECG signal turned out to be quite low and the output was not oscillating around 0. So different numbers were experimented with and 0.1 was the optimal choice. Numbers other than this would result in a signal which has a DC offset to some degree.

Loop Through Each Sample:

1. **Reference Noise**: A reference noise signal is generated with both a 50 Hz sine wave and a small DC offset (set to 0.1 for a flat response).
2. **Noise Estimation**: The filter method runs on this reference noise to estimate the noise in the ECG signal. The LMS filter gradually improves its noise estimation as it adjusts the coefficients.
3. **Noise Cancellation**: The estimated noise (canceller) is then subtracted from the original ECG signal, leaving a cleaner output.
4. **Coefficient Update**: The lms method updates the filter coefficients based on the output error to refine the noise estimate on the next sample.
5. **Store Filtered Output**: The cleaned ECG sample is stored in y, which holds the final output.

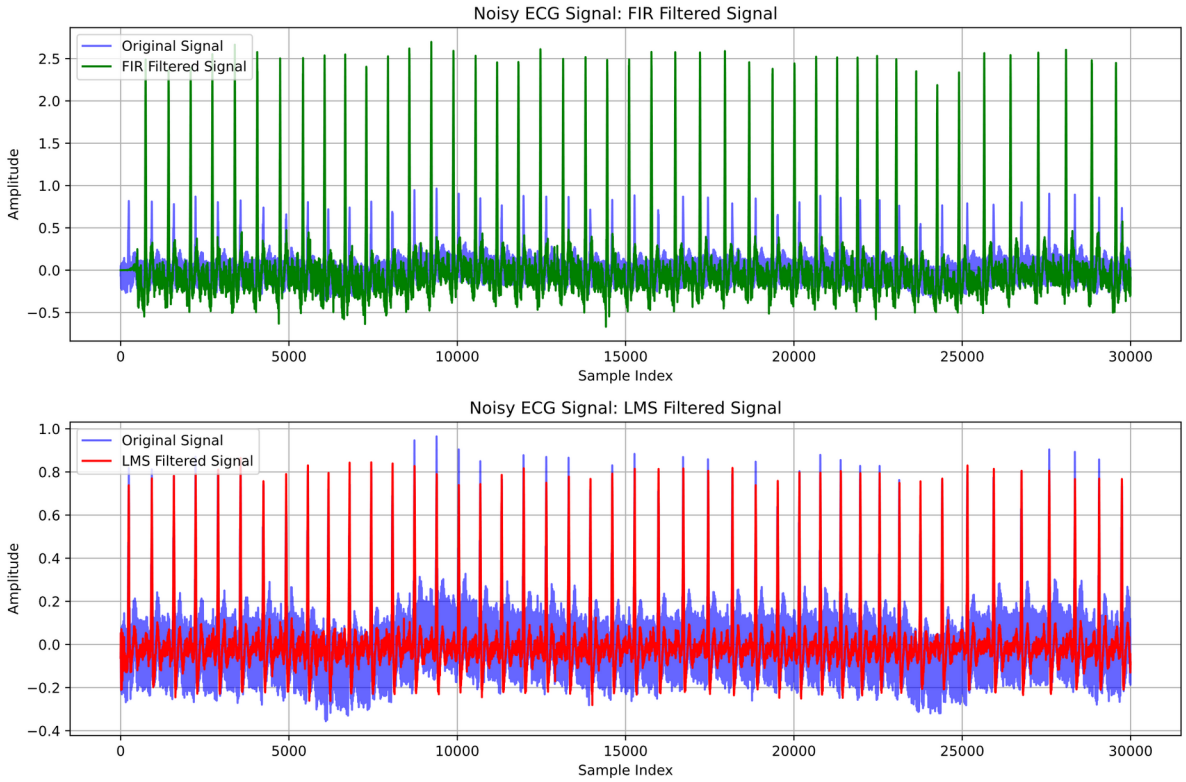

Figure 5 - Comparing LMS filter vs FIR filter to noisy signal

It can be observed that the output is not shifted, and the magnitudes are not reduced=means good output. Also, the Adaptive filter response has a flatter response than the initial FIR filter design.

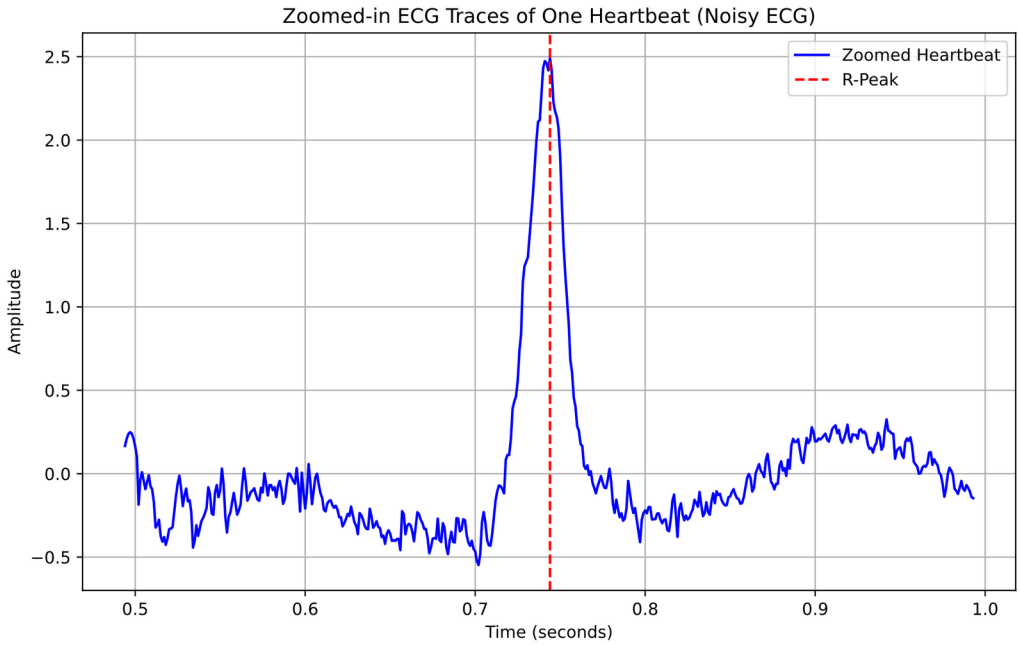

Figure 6 - Single heartbeat of a filtered noisy signal to check the pqrst intact.

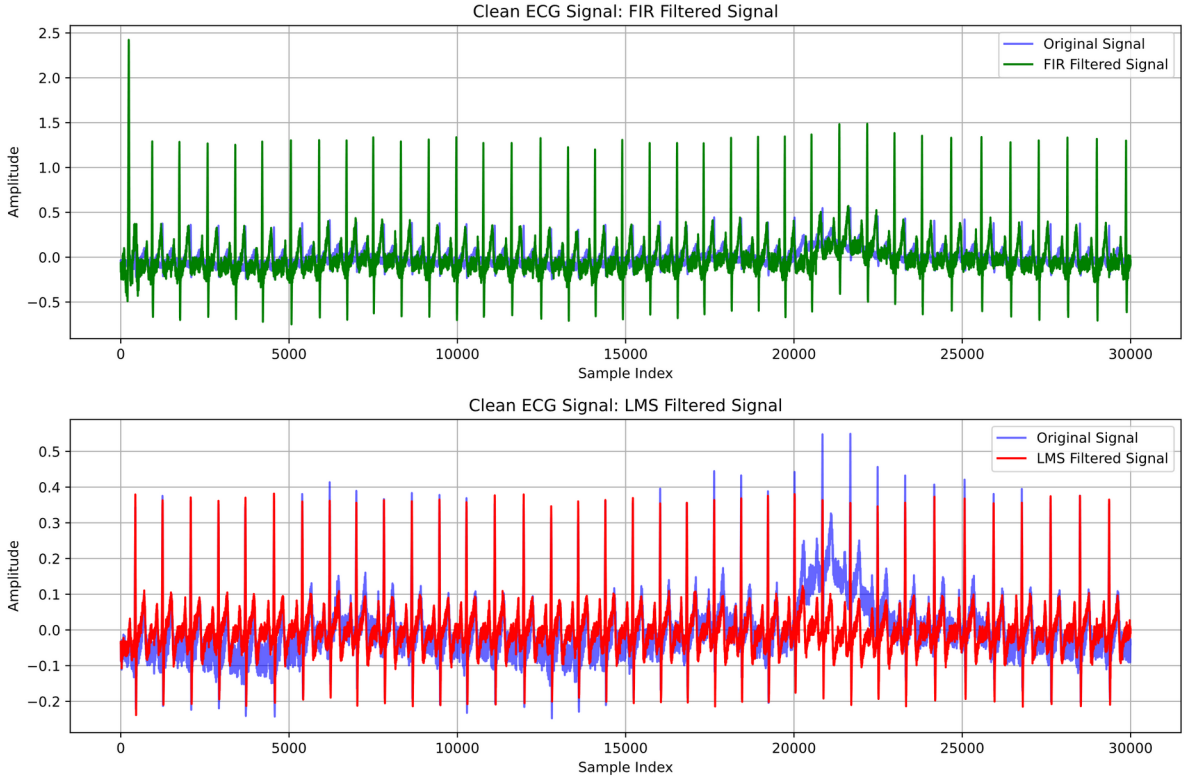

Figure 7 - Comparing LMS filter vs FIR filter to clean (standing ECG) signal

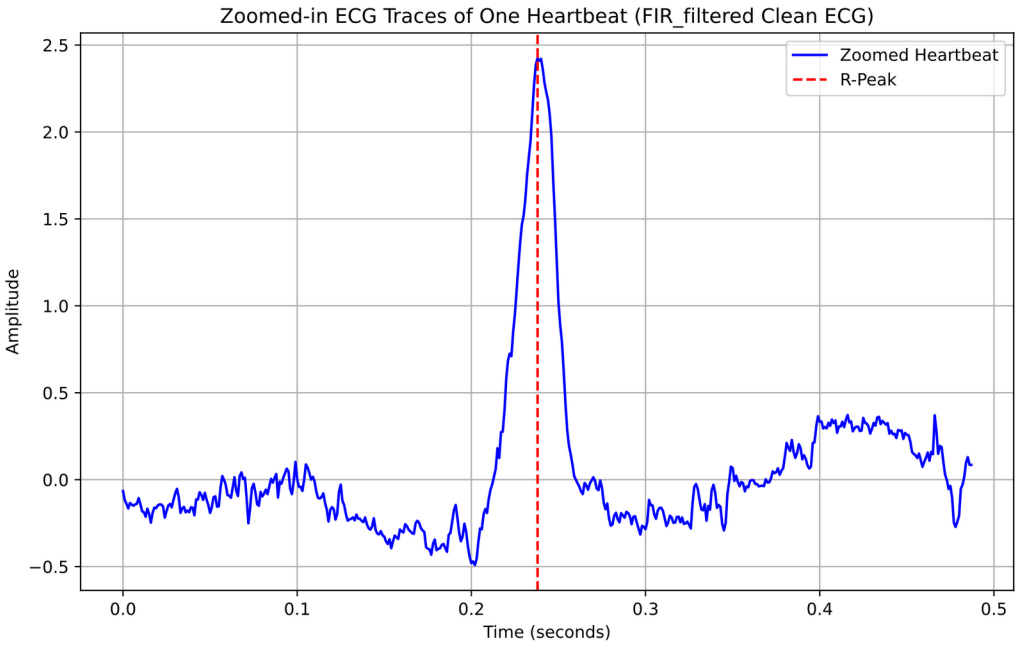

Figure 8 - Single heartbeat of a filtered clean signal to check the pqrst intact.

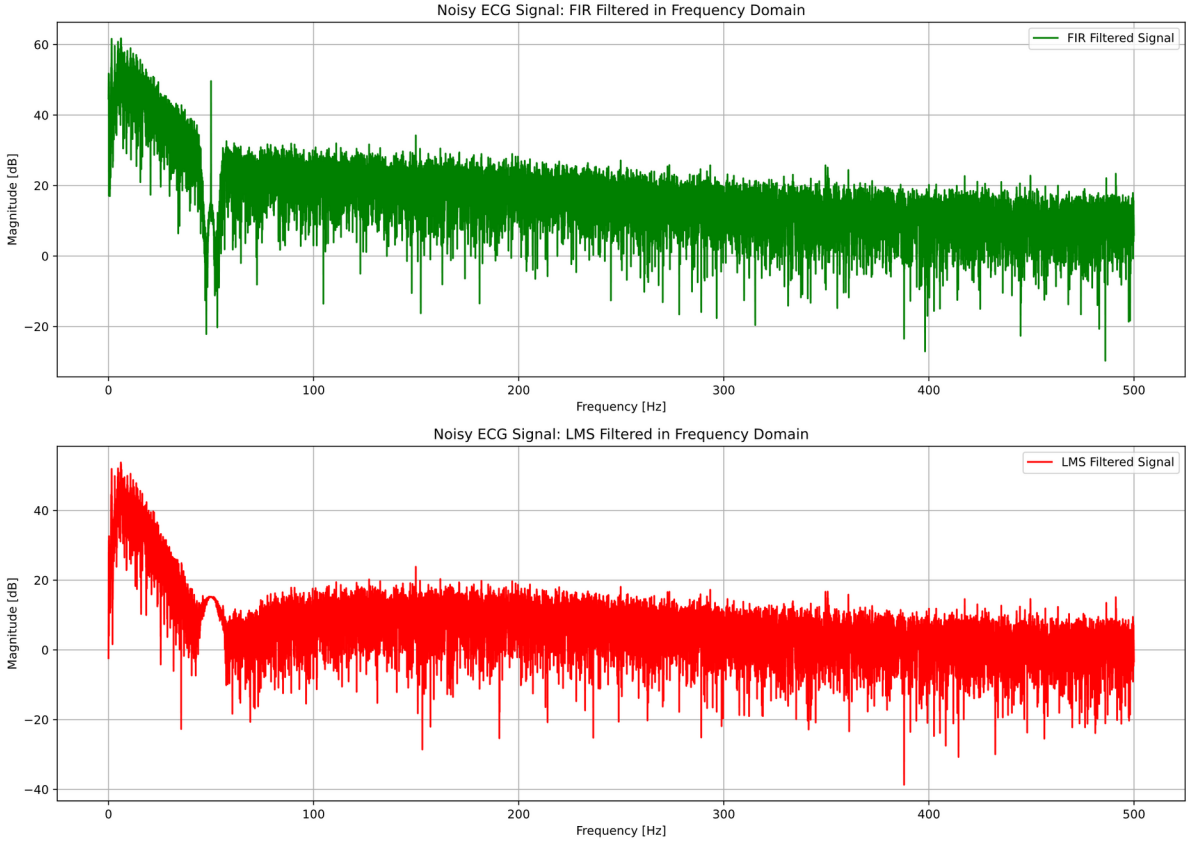

Figure 9 - Comparing frequency response of FIR filtered vs LMS filtered noisy signal

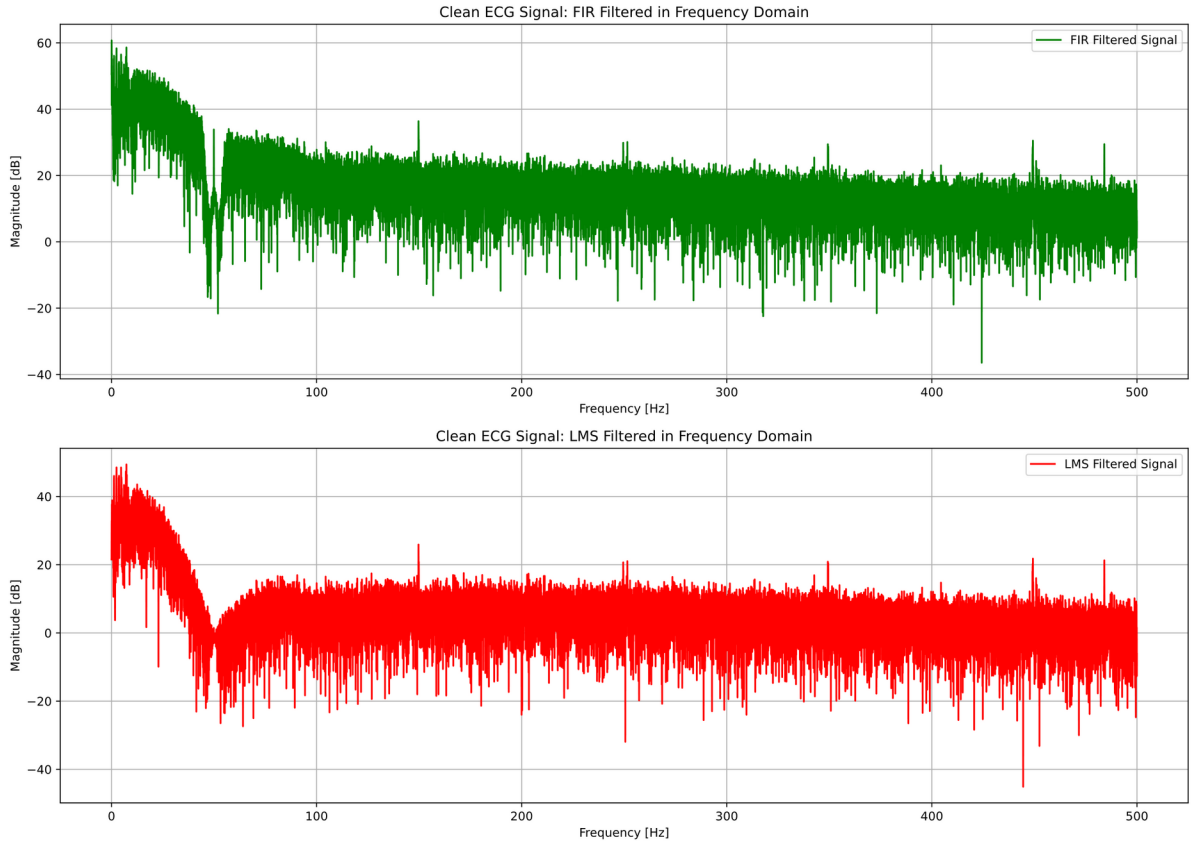

Figure 10 - Comparing frequency response of FIR filtered vs LMS filtered clean signal

- **Task 4: ECG heart beat detection**

The matched filter works by correlating an input signal to a known template signal (usually is one sample of what is to be detected) to detect the known template in the input signal. For example, to detect a heartbeat in an ECG signal, the matched filter will correlate the ECG signal with a known template of a heartbeat signal to detect a heartbeat in the ECG. To perform a heartbeat detection, the signal is first prefiltered to remove any DC and 50Hz noise components. A template heartbeat is then obtained by estimating a perfect heartbeat signal that will match. Next the template is reversed to obtain our FIR coefficients. The ECG signal is then filtered with the time reversed template. The result is usually squared to improve SNR value.

Choosing the Best Template - Different R-peaks in the clean signal were experimented with to find the best template. After testing a few, the one that had a good amplitude, minimal noise, and a shape was chosen for the matched filter to detect. This was important since using too large a template introduces noise, while too short a template doesn't capture the full shape of the peak. The length had to be just right to capture the essential part of the R-peak.

Reversing the Template for Matched Filtering - To make the matched filter work, the template was time-reversed. The reason for this is that a matched filter needs to align with incoming signals, so the template must be flipped. With the reversed template, the filter was able to catch similar patterns in the signal, giving clear peaks in the output.

Comparison of Template and Reversed - The original template was plotted against the reversed template to make sure they matched up as expected. This check was important because any mismatch would mean the filter might miss the pattern. The reversed template aligned well, giving a strong

response in the matched filter output.

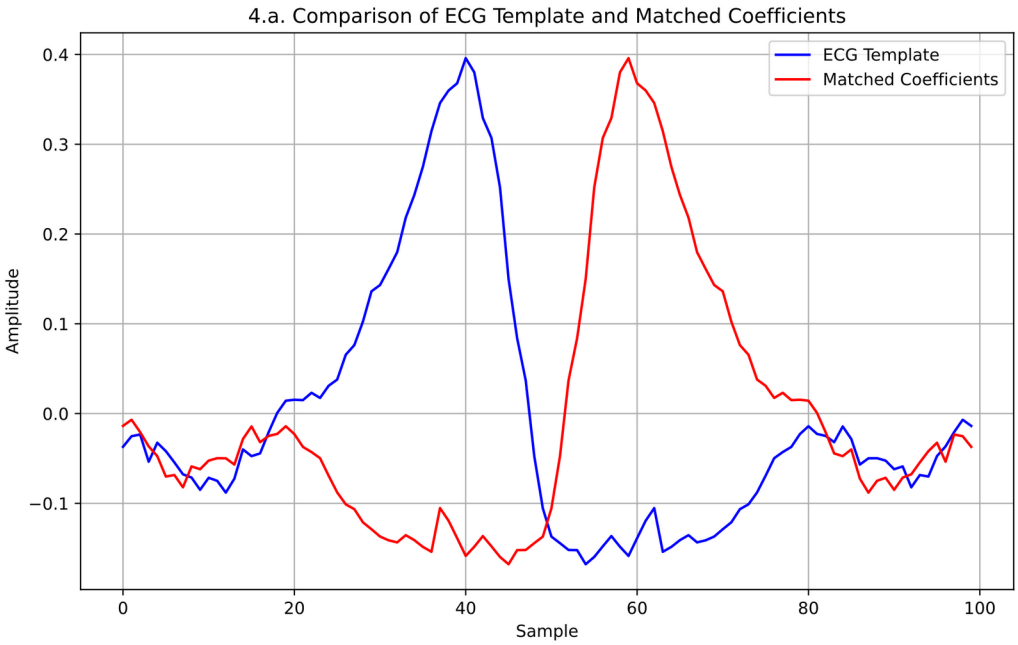

Figure 11 – Comparing the template and the reversed to ensure mirroring

Enhancing Signal-to-Noise Ratio through Squaring - After applying the matched filter, it was observed that some noise and amplitude imperfections remained. The waveforms were not as clear and sharp as desired. To improve the signal-to-noise ratio (SNR), the filtered data was multiplied by itself, effectively squaring the signal.

This technique works because noise amplitudes are typically small (close to zero). Squaring these small values reduces their magnitude even further, effectively suppressing noise. Furthermore, the R-peaks have larger amplitudes. Squaring these values increases their magnitude significantly, making the peaks more pronounced. The resulting output is given in Figure 12 below.

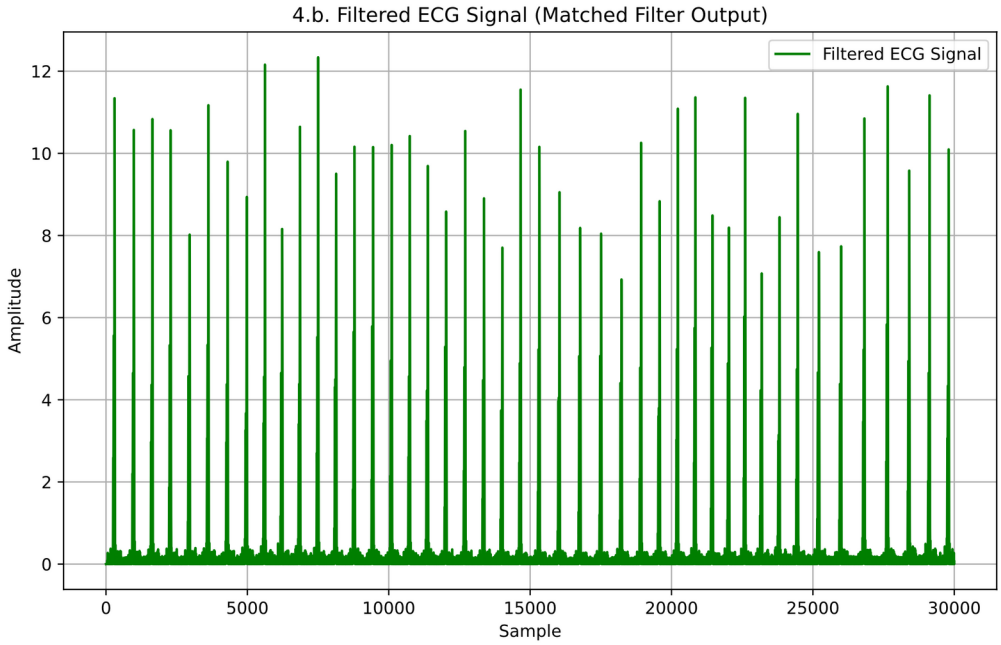

Figure 12 – Matched filter output

Momentary Heartbeat Detection - To analyse the momentary heart rate, beats per minute (BPM) were calculated from the filtered ECG signal over a duration of 30 seconds (duration of the ECG). R-peaks were detected using a threshold-based method and computed the RR intervals, which represent the time between consecutive R-peaks. The RR intervals were then converted to BPM using the formula $BPM = 60 / Rinterval$ (in seconds). The graph (see Figure R) showed a consistent range of BPM values, approximately between 73 and 105 BPM, further confirming the accuracy of our approach**.**

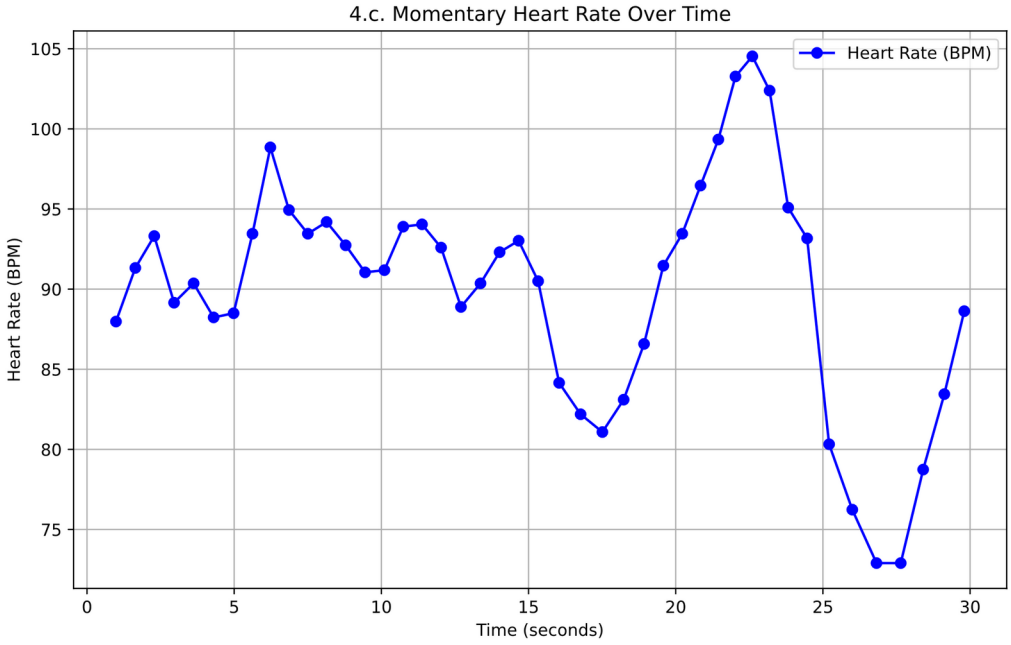

Figure 13 – Heart rate over time

Conclusion

In this assignment, FIR filters were effectively designed and implemented for ECG signal processing tasks, including noise removal and feature detection. By carefully selecting the filter order based on frequency resolution and optimizing the code with **np.roll**, real-time processing capabilities were achieved. High-pass and band-stop filters successfully **removed the DC offset and 50 Hz powerline interference**. The matched filter, utilizing a time-reversed template of the R-peak, detected these features within the noisy signal. Further enhancement of the signal-to-noise ratio was accomplished by squaring the filtered output and amplifying the R-peaks while suppressing noise. The methods and parameter selections were justified based on theoretical considerations and practical observations, resulting in **improved ECG signal clarity and processing efficiency**.

GitHub Repository

The full code and additional lab assignments for this Digital Signal Processing (DSP) course can be found on the GitHub repository: DSP Lab Assignments. The repository contains all tasks and scripts, for FIR and FFT.

References

- **Python Libraries**: numpy, matplotlib, wave, scipy.signal (used just to find the peak and nothing else)