Marella MORAD - 103076428

Further explanation of Project Code – ENG10004 – Projects A & B

Project A:

MinPeakProminence:

This property of the findpeaks() command, allowed me to identify the R-peaks more accurately than using a yLimit to find them as we were taught in the lab sessions. As shown in figure 2, the peak prominence is indicated by the purple arrows.

```
%% Finding the peaks of the filtered data and plotting it
subplot(2,1,2);
plot(data);
findpeaks(data, 'MinPeakProminence',140);
xlabel('Samples (360 samples/s)');
ylabel('Amplitude (mV)');
title('Filterd ECG Signal');
ylim([miny-100 maxy+100]);
xlim([0 3650]);
hold off;
```

Figure 1: MinPeakProminence code

"The prominence of a peak measures how much the peak stands out due to its intrinsic height and its location relative to other peaks" (Mathworks, 2020). The 140 min peak prominence I am using was obtained among analysis of the data and testing numbers. This value differs however in project B, since the filtering (passband frequency) changes as well.

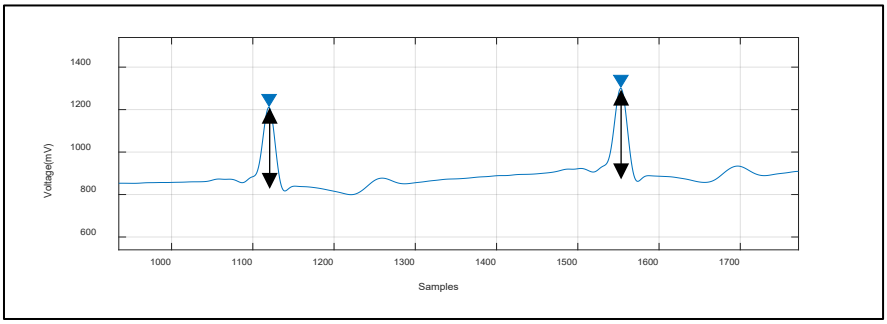


Figure 2: MinPeakProminence explained

Project B:

Filtering Choice:

```
%% Filtering data using a lowpass filter
data = lowpass(Orig_Sig,0.07); % 0.07 is the passband frequency, different from Project A!
```

For part B of the project, the previous filtering (the one used in project A) did not work out, since it made the signal too smooth. In that filtering, I used 0.01 as my passband frequency. I did not use the FFT function to find the frequency, however, I experimented on the data with several different passband frequencies, and was able to find the one that suited the purpose of the project, that is 0.07. This gave 143 peaks for Sample_1. Using this I was able to construct some code that allowed me to find all the required attributes. Other passband frequencies altered with the position of the required attributes, as show in figure 3.

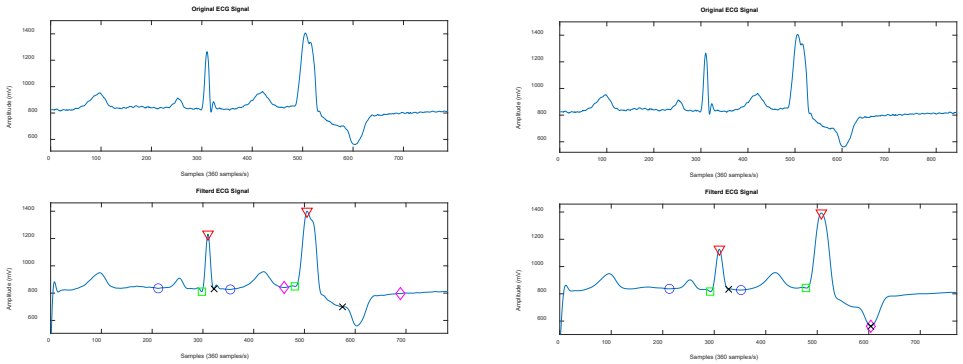


Figure 3: comparison between two passband frequencies, 0.07 to the left & 0.01 to the right

Finding and plotting the Q peaks (similar approach for S peaks):

```
%% Find and plot Q-peaks
current_Q = locs_R(n); %assigning a varible for the R-peak that will be used a reference point
previous_Q = current_Q - 2; %the Q-peak is before the R-peak, so lowering the value by 2
% looping through the data based on the indicies current_Q and
% previous_Q until finding a point where the current_Q is lower than
% the previous_Q
while data(current_Q) >= data(previous_Q)
    current_Q = previous_Q;
    previous_Q = previous_Q - 2;
end
%plotting the Q peaks, with a green squared marker
plot(current_Q,data(current_Q),'s','MarkerSize', 7.5, 'MarkerEdgeColor','g');
hold on;
```

As explained in the comments, the Q-peaks are found with reference to the R-peaks. For example, choosing n = 1, will result in the first R-peak. This will first be assigned to current_Q, so that it can be compared to previous_Q, which is 2 less than current_Q. After assigning the values they are then compared to find the lowest point (i.e. minimum turning point, that represents the Q peak).
**The same approach is used for finding the S-peak, the initial point of the P-wave, and the ending point for the T-wave.*

Finding the P-wave starting point (similar approach for T-wave):

```
%% Finding the P wave starting point
for k = 1:3
    p_peaks(k) = locs(R_Peaks_Match(n) - k);
    p_max = max(data(p_peaks));
    p_max_index = find(data==p_max);
end
```

I learnt that the P-wave, is the turning point of the highest peak, between the three previous peaks, with a reference to the Q-peak. Therefore, I implemented this statement as code. Firstly, I found the three peaks before the R-peak (which is the same thing as the S-peak). I then found the maximum peak out of them and got the index value for that peak. I then implemented the same

code that I used for the Q peaks (see previous section), to find the turning point of this peak, which represents the starting point of the P-wave.
A similar approach was used to find the T-wave ending point.

Finding the time intervals (PR, QT and QRS complex):

I used three matrices to save the multiple values for each interval as the loop is running and n changing. I then used these values to calculate the average value for each intervalto then display it to the user and compare it to the normal range.

```
    %% Recording the requested values
    PR_Interval(n) = current_Q - current_TP_P;

    QT_Interval(n) = current_TP_T - current_Q;

    QRS_Duration(n) = current_S - current_Q;
end

%% Calculating Average Values
%Average PR and PR inteval in sec
Ave_PR = mean(PR_Interval);
PR_s = Ave_PR*10/3600;

%Average QT and QT inteval in sec
Ave_QT = mean(QT_Interval);
QT_s = Ave_QT*10/3600;

%Average QRS and QRS Complex in sec
Ave_QRS = mean(QRS_Duration);
QRS_s = Ave_QRS*10/3600;
```

Necessary adjustments:

1- After trying on the other 7 samples, I found some errors in finding the requested attributes. These main erro was that: *The index exceeds the number of elements*. I fixed this problem using conditional if statements as shown in the figure below.

```
current_S = locs_R(n);
if current_S < 3598
    next_S = current_S + 2;
    while data(current_S) >= data(next_S)
        if current_S < 3597
            current_S = next_S;
            next_S = next_S + 2;
        else
            break;
        end
    end
    plot(current_S,data(current_S),'x','MarkerSize', 7.5, 'MarkerEdgeColor','k');
    hold on;
end
```

a- The first if statements, `if current_S < 3598`
Upon experiment, I discovered that since next _S = current_S + 2, current_S cannot exceed 3598. This is because, if we take current_S as 3599, next_S will be 3601, which is over the indices of our data (3600).

b- The second if statement, inside the while loop, `if current_S < 3597`
Again, upon experiment, I have noticed that since current_S is assigned to whatever the values is for next_S after the while condition is check, current_S cannot exceed 3597. The case where it exceeds would result in an incompatibility error (index number greater than 3600)

2- I also faced the same error while finding the T-waves, however, this time the index number was related to the R-peaks, not the whole entire data.

```
%% Finding the T-wave ending point
if R_Peaks_Match(n) <= length(locs) - 4
    for l = 1:4
        t_peaks(l) = locs(R_Peaks_Match(n) + l);
        t_max = max(data(t_peaks));
        t_max_index = find(data==t_max);
    end
end
```

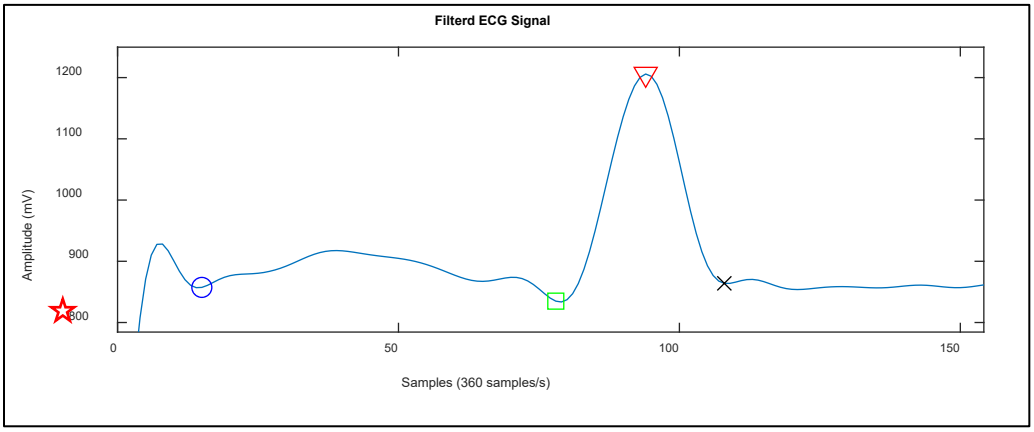
Note: in finding the T-wave, and after experimenting on the data, I decided that 4 peaks after the R-peak, instead of 3 peaks that I used in finding the p-waves, was more suitable to this situation.
Since I am looking for 4 peaks after the current R-peak, the index number for the R-peak needed to be less than or equal to the total number of R-peaks minus 4.

3- No data present for analysis

```
%Find and plot turning point of P_wave
if p_max_index < 10
    current_TP_P = p_peaks(2);
else
    current_TP_P = p_max_index;
end

previous_TP_P = current_TP_P - 2;
while data(current_TP_P) >= data(previous_TP_P)
    if current_TP_P > 4
        current_TP_P = previous_TP_P;
        previous_TP_P = previous_TP_P - 2;
    else
        break;
    end
end
```

This error, I noticed while running the code on sample_3. The problem was that the 3rd peak before the first R-peak was very far from the R-peak which led to errors in the calculations of the time intervals.



So instead of having the starting point of the P-wave (blue circle) where it is in the above graph, it was set to the red star also shown in the above graph. Therefore, I examined the numbers and was able to determine the correct condition that would give me the position of the blue circle as the p-wave starting point.

I also faced another challenge, because of sample_3, however with the use of the second if statement (inside the while loop), I was able to overcome it. This problem is very similar to what I faced in finding the T-wave and the S-peak and not having the index number > 3600. This time the index number was getting smaller and eventually, less than 0. Therefore, the condition that the value of current_TP_P has to be grater than 4, resolved the issue.

ismember command:

I used the ismember function just to find the indices of the R-peaks in the data. For example, in sample 1, I had 10 R-peaks and a total of 143 peaks, so finding where exactly these peaks occur (index number), really helped me in finding the P and the T waves, as can be seen in the project code.

```
%Match find the R-peaks indicies in the matrix that holds all the peaks to
%be able to use it as a reference point
matchIdx = ismember(peaks, peaks_R); %logical varible that returns 1 or 0, 1 if the value is found and 0 if not found
R_Peaks_Match = find(matchIdx); %collecting the values that are equal to 1 and storing them in R_Peaks_Match
```