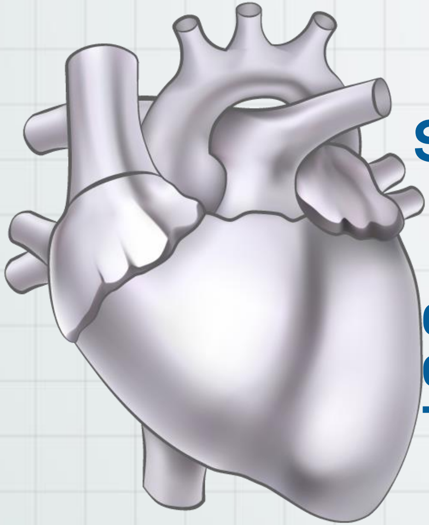


Implementation of TCSC(Threshold Crossing Sample Count) Algorithm for The Detection of Ventricular Fibrillation in Electrocardiogram

Course No- EEE 312

Group No- 02

Team Members- Zafrin Jahan Nikita(1806164)
Sadia Tasnim Mou (1806173)
Sudipto Pramanik (1806172)
Protoye Mohanta (1806179)

Submitted to :

Shahed Ahmed, Lecturer, EEE, BUET
Shafin Bin Hamid, Lecturer, EEE, BUET

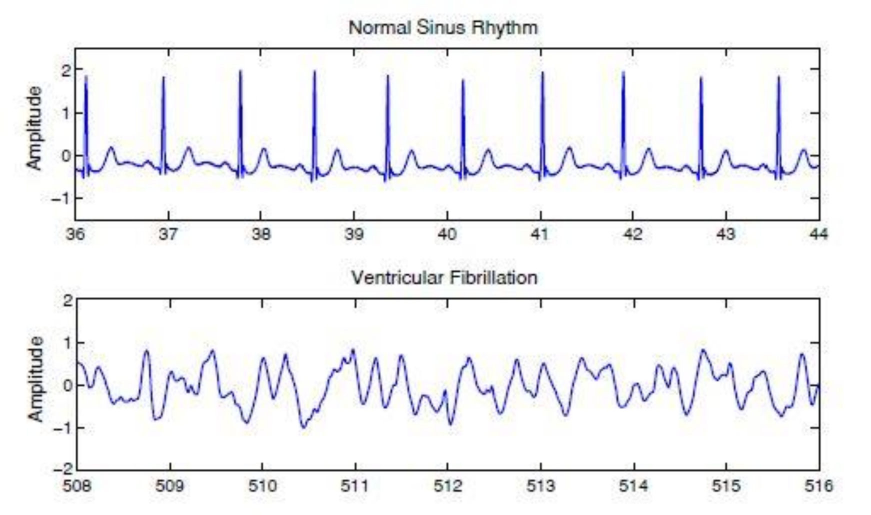
Objectives

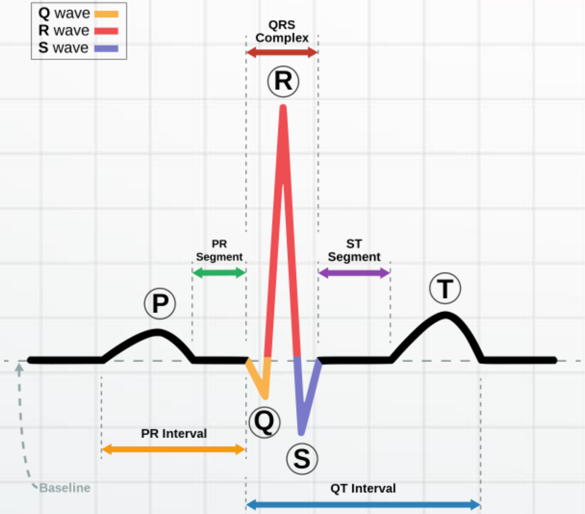
Implementation of TCSC (Threshold Crossing Sample Count) Algorithm 01

Faster and Accurate Detection of Ventricular Fibrillation 02

Ventricular Fibrillation

- Ventricular fibrillation (VF) is a life-threatening cardiac arrhythmia.
- It is due to the random occurrence of many pacemaker cells inside the heart.
- VF can lead to death within minutes if left untreated.





Normal Sinus Rhythm

- In this project, we apply a time domain algorithm called threshold crossing sample count (TCSC),which is an improved version of the threshold crossing interval (TCI) algorithm for VF detection.
- The algorithm is based on an important feature of the VF signal which relies on random behaviour of the electrical heart vector.
- By two simple operations: comparison and count, the technique calculates an effective measure which is used to separate life threatening.

Comparison Between TCI(Threshold Crossing Interval) and TCSC(Threshold Crossing Sample Count)

1. **A 3-s stage is investigated in TCSC instead of 1-s stage.**

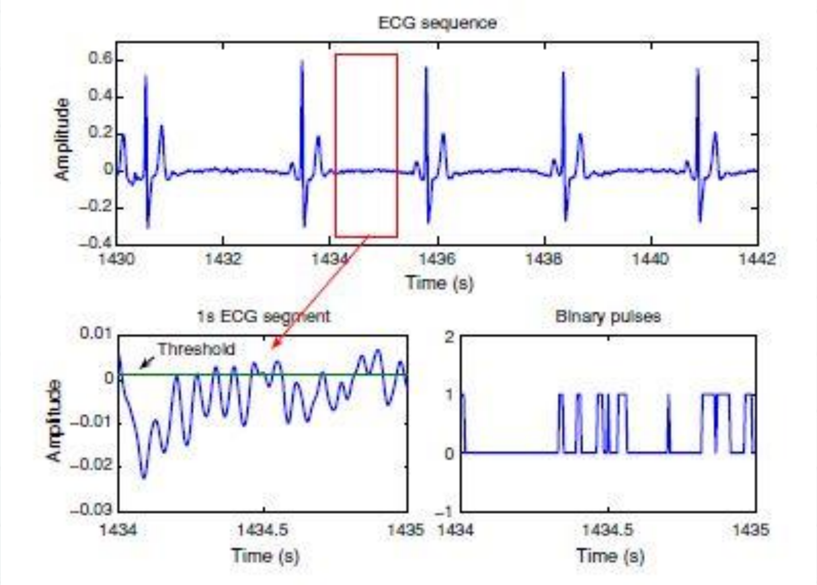


Fig: TCI algorithm for low bit rate

Comparison Between TCI(Threshold Crossing Interval) and TCSC(Threshold Crossing Sample Count)

2. Both positive and negative thresholds are used in TCSC instead of only positive threshold

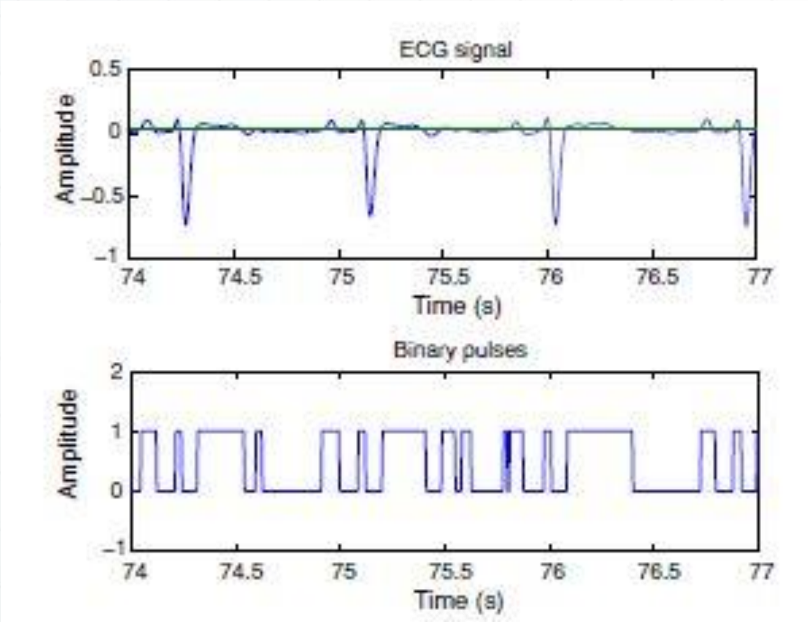


Fig: TCI algorithm for using positive peak to determine the threshold.

Comparison Between TCI(Threshold Crossing Interval) and TCSC(Threshold Crossing Sample Count)

3. Samples above the thresholds are counted in TCSC instead of counting the pulses.

4. In TCSC Moving average filter is applied to make decisions on each Le-second ECG episode (Le > 3).

Dataset Information

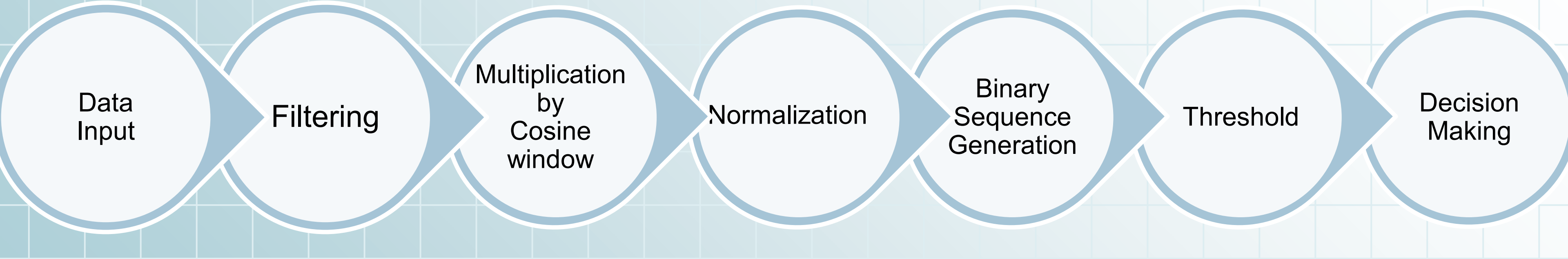
Dataset used in this project is MIT-BIH arrhythmia and CU databases.

Link:

1.“Massachusetts Institute of Technology ,NSR database.” http://www.physionet.org/physiobank/database/mitdb

2. “Massachusetts Institute of Technology, CU database.” http://www.physionet.org/physiobank/database/cudb

Workflow



Data Input

Filtering

Multiplication by Cosine window

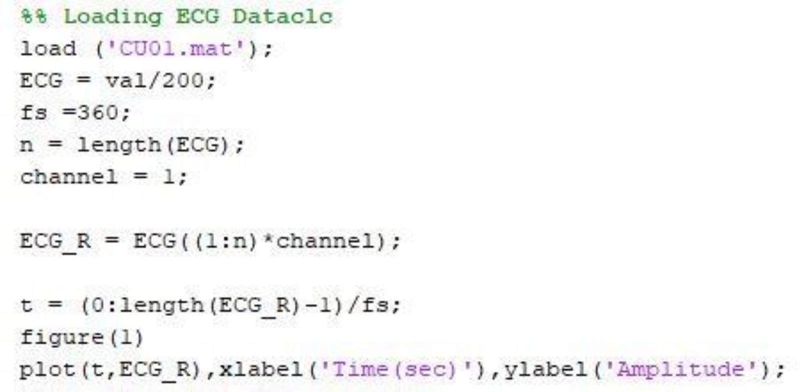
Normalization

Binary Sequence Generation

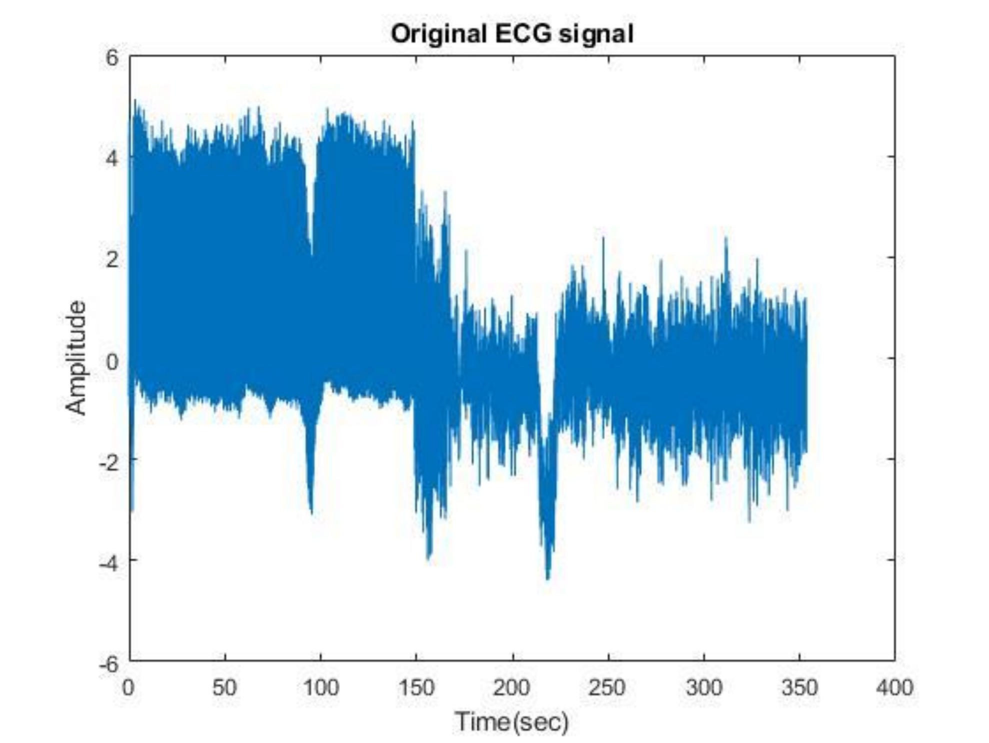
Threshold

Decision Making

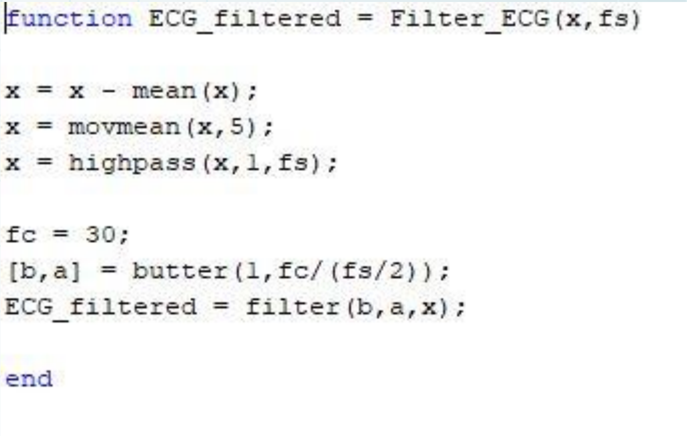
ECG Data Input



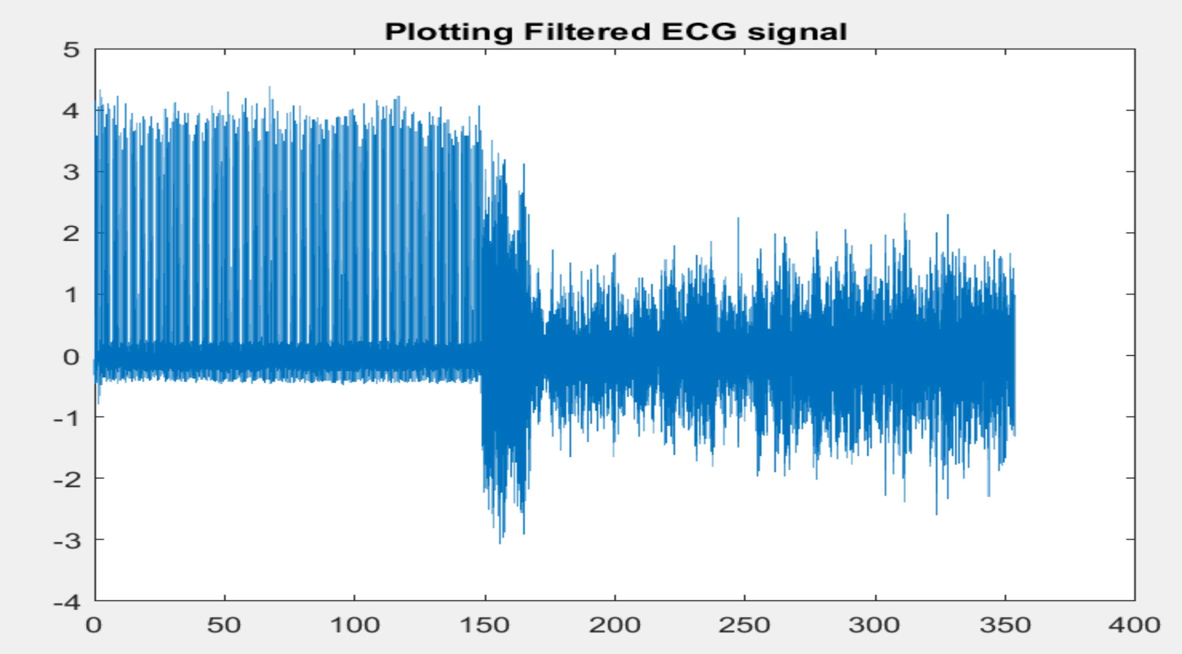
```
%% Loading ECG Dataclc
load ('CU01.mat');
ECG = val/200;
fs =360;
n = length(ECG);
channel = 1;

ECG_R = ECG((1:n)*channel);

t = (0:length(ECG_R)-1)/fs;
figure(1)
plot(t,ECG_R),xlabel('Time(sec)'),ylabel('Amplitude');
```



Filtering

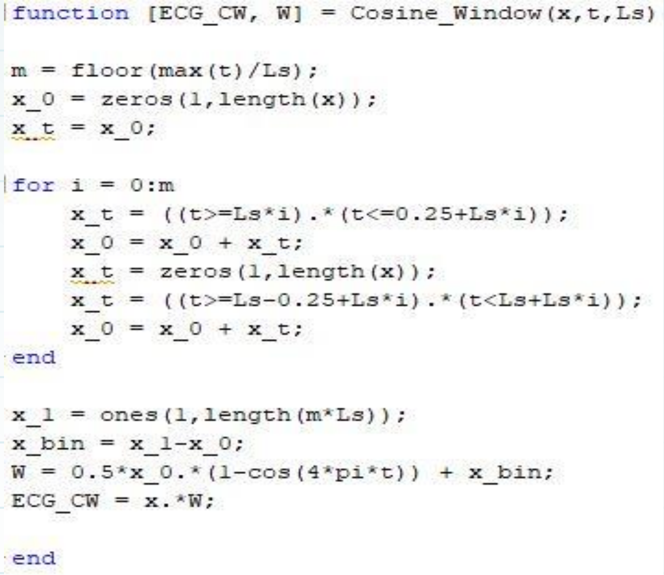


```
function ECG_filtered = Filter_ECG(x,fs)

x = x - mean(x);
x = movmean(x,5);
x = highpass(x,1,fs);

fc = 30;
[b,a] = butter(1,fc/(fs/2));
ECG_filtered = filter(b,a,x);

end
```

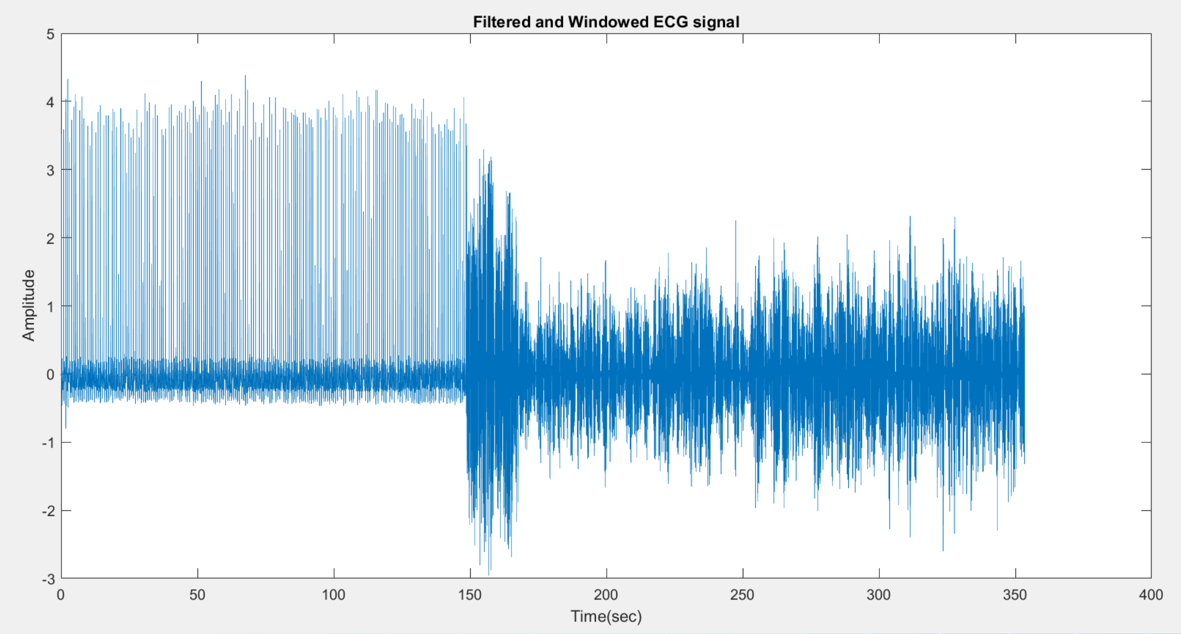


- Drift suppression is carried out by high pass filter of cutoff frequency 1 Hz
- Low pass Butterworth filter of cutoff frequency 30Hz is used to suppress the high frequency component

Cosine Window



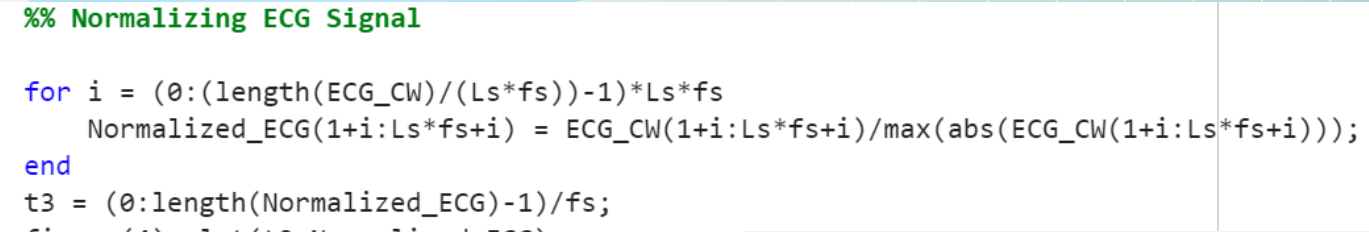
```
function [ECG_CW, W] = Cosine_Window(x,t,Ls)

m = floor(max(t)/Ls);
x_0 = zeros(1,length(x));
x_t = x_0;

for i = 0:m
    x_t = ((t>=Ls*i).*(t<=0.25+Ls*i));
    x_0 = x_0 + x_t;
    x_t = zeros(1,length(x));
    x_t = ((t>=Ls-0.25+Ls*i).*(t<Ls+Ls*i));
    x_0 = x_0 + x_t;
end

x_1 = ones(1,length(m*Ls));
x_bin = x_1-x_0;
W = 0.5*x_0.*(1-cos(4*pi*t)) + x_bin;
ECG_CW = x.*W;

end
```



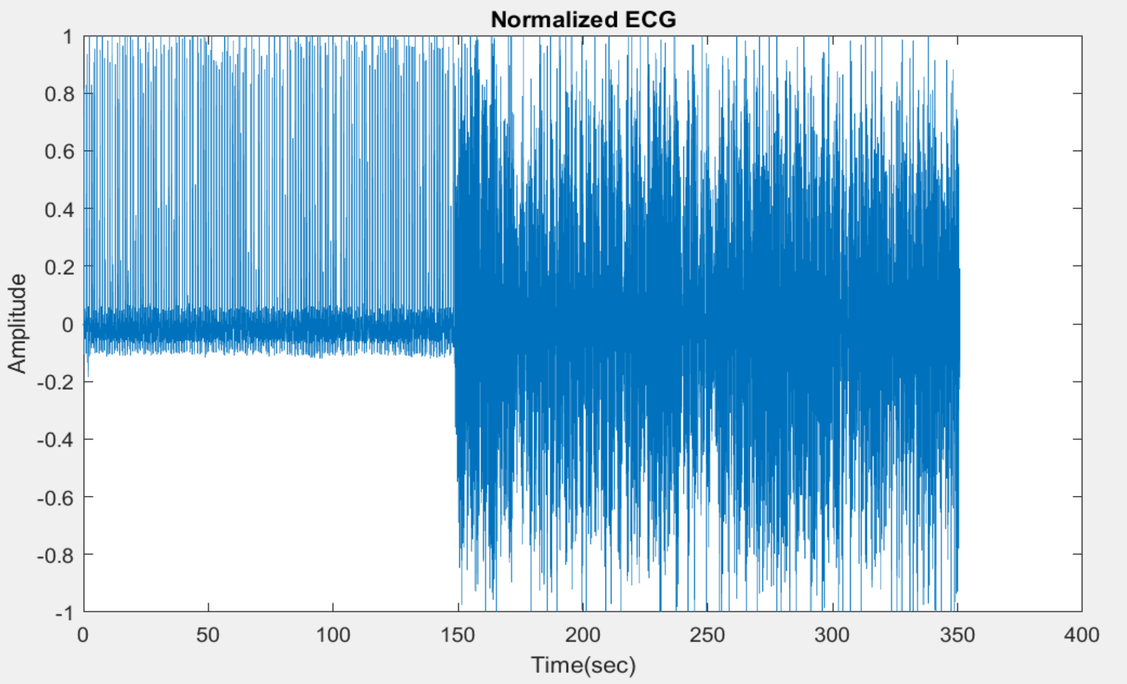
Cosine window is applied on filtered signal, where Ls = 3, length of ECG 3s segment

$$w(t) = \begin{cases} \frac{1}{2}(1-\cos(4\pi t)) & 0 \le t \le \frac{1}{4} \\ 1 & \frac{1}{4} \le t \le L_s - \frac{1}{4} \\ \frac{1}{2}(1-\cos(4\pi t)) & L_s - \frac{1}{4} \le t \le L_s \end{cases}$$

Normalizing ECG Signal

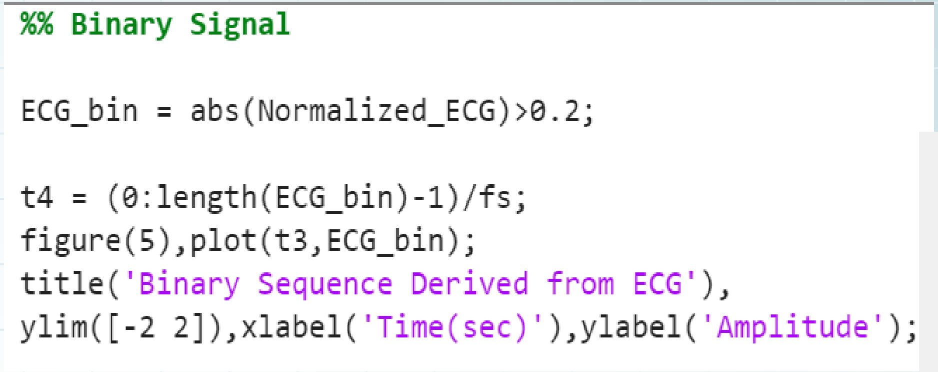


```
%% Normalizing ECG Signal

for i = (0:(length(ECG_CW)/(Ls*fs))-1)*Ls*fs
    Normalized_ECG(1+i:Ls*fs+i) = ECG_CW(1+i:Ls*fs+i)/max(abs(ECG_CW(1+i:Ls*fs+i)));
end
t3 = (0:length(Normalized_ECG)-1)/fs;
figure(4),plot(t3,Normalized_ECG),
xlabel('Time(sec)'),ylabel('Amplitude');
title('Normalized ECG');
```



ECG signal is normalized by absolute maximum value in the segment

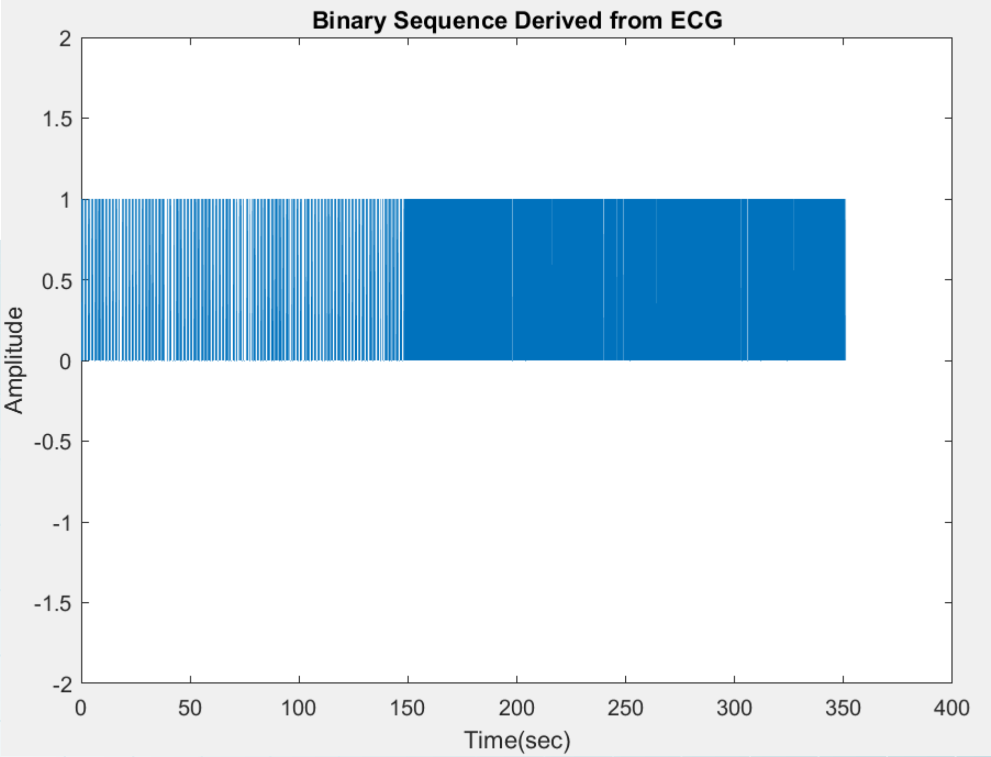
Binary Signal



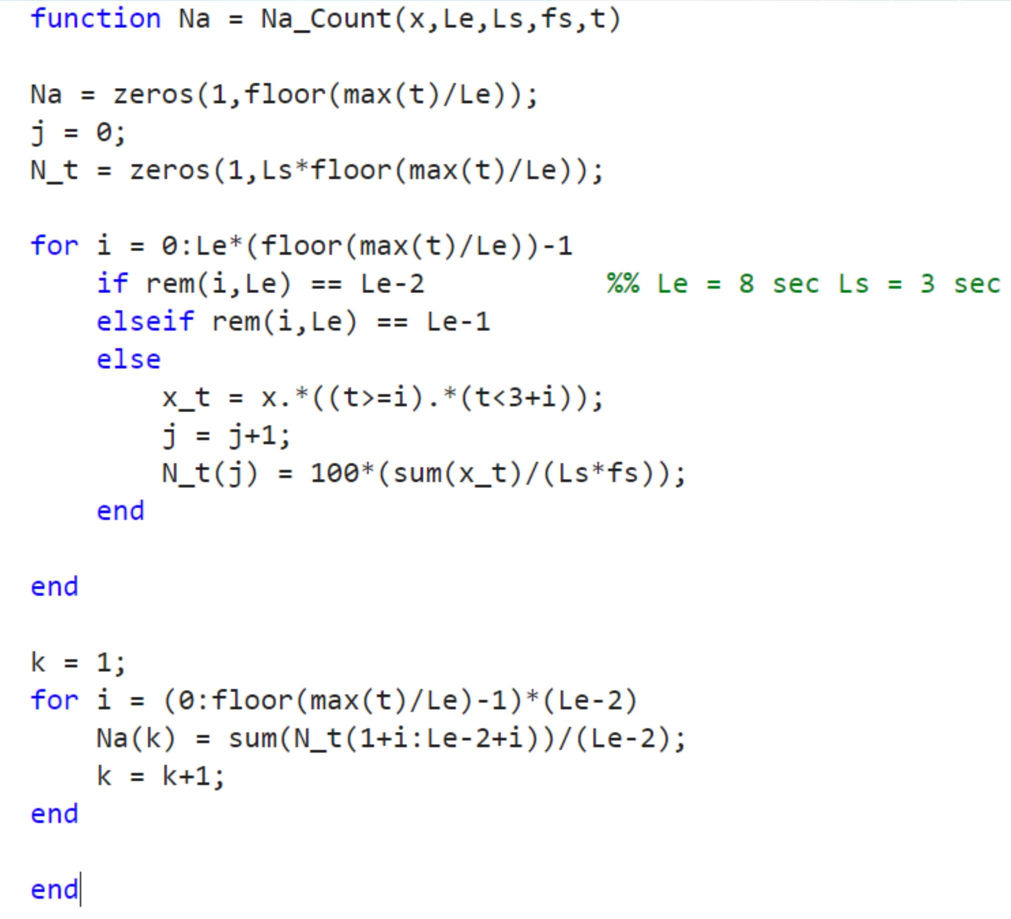
```
%% Binary Signal

ECG_bin = abs(Normalized_ECG)>0.2;

t4 = (0:length(ECG_bin)-1)/fs;
figure(5),plot(t3,ECG_bin);
title('Binary Sequence Derived from ECG'),
ylim([-2 2]),xlabel('Time(sec)'),ylabel('Amplitude');
```

The normalized ECG signal is converted into a binary string of 0–1 by comparing each data sample with threshold absolute(Vo)= 0.2

The value Vo = 0.2 is suitably selected after several empirical studies. The baseline of the preprocessed ECG signal is assumed to go below this threshold. This assumption is valid for VF and works for SR also.



Counting Na

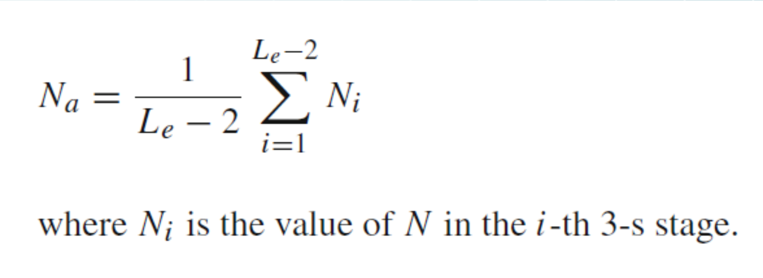


```
function Na = Na_Count(x,Le,Ls,fs,t)

Na = zeros(1,floor(max(t)/Le));
j = 0;
N_t = zeros(1,Ls*floor(max(t)/Le));

for i = 0:Le*(floor(max(t)/Le))-1
    if rem(i,Le) == Le-2           %% Le = 8 sec Ls = 3 sec
    elseif rem(i,Le) == Le-1
    else
        x_t = x.*((t>=i).*(t<3+i));
        j = j+1;
        N_t(j) = 100*(sum(x_t)/(Ls*fs));
    end

end

k = 1;
for i = (0:floor(max(t)/Le)-1)*(Le-2)
    Na(k) = sum(N_t(1+i:Le-2+i))/(Le-2);
    k = k+1;
end

end
```

N is the percent of data samples that cross Vo, obtained by counting the number of 1s in the binary sequence

$$N = \frac{Number\ of\ samples\ that\ cross\ Vo}{Total\ number\ of\ samples} * 100$$

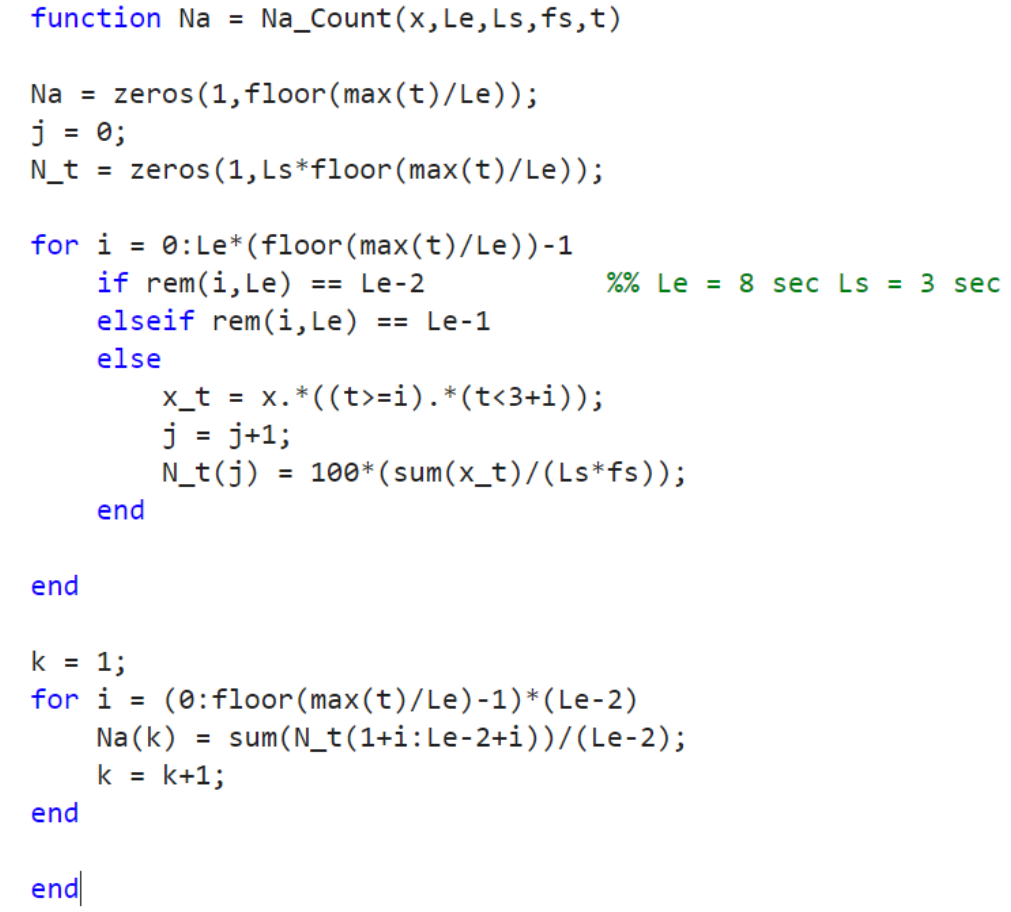
Decision is made on every Le = 8s ECG episode. By averaging Le – 2 = 8–2 = 6 consecutive values of N. The average value of Na in each 8s segment is calculated.



$$N_a = \frac{1}{L_e - 2}\sum_{i=1}^{L_e-2} N_i$$

where N_i is the value of N in the i-th 3-s stage.

Counting Na

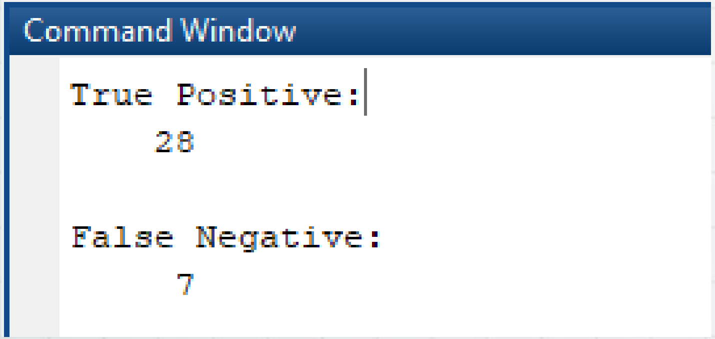


```
function Na = Na_Count(x,Le,Ls,fs,t)

Na = zeros(1,floor(max(t)/Le));
j = 0;
N_t = zeros(1,Ls*floor(max(t)/Le));

for i = 0:Le*(floor(max(t)/Le))-1
    if rem(i,Le) == Le-2           %% Le = 8 sec Ls = 3 sec
    elseif rem(i,Le) == Le-1
    else
        x_t = x.*((t>=i).*(t<3+i));
        j = j+1;
        N_t(j) = 100*(sum(x_t)/(Ls*fs));
    end

end


k = 1;
for i = (0:floor(max(t)/Le)-1)*(Le-2)
    Na(k) = sum(N_t(1+i:Le-2+i))/(Le-2);
    k = k+1;
end

end
```

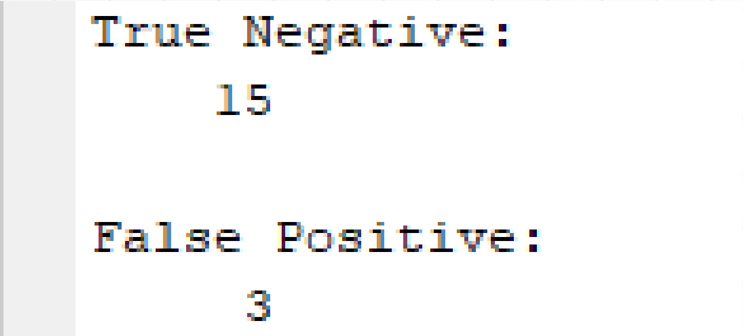
If Na is above Nd = 48, VF is detected

Nd = 48 is selected from probability distribution. The chosen threshold value works well in case of conduction disorder with longer QRS duration or in case of higher heart rate. If we want to put emphasis on high sensitivity, the value Of Nd should fall in the range : 25<Nd<35 *Nd* is the critical threshold parameter (ctp) to obtain the receiver operating characteristic(ROC) curve.

Testing on Database



```
True Positive:
    28

False Negative:
     7
```



```
True Negative:
    15

False Positive:
     3
```

Confusion Matrix

n=53	Predicted No	Predicted Yes	
Actual No	TN= 15	FP=3	18
Actual Yes	FN= 7	TP= 28	35
	22	31	

Calculation

Sensitivity= $\frac{\mathrm{TP}}{\mathrm{TP+FN}} = \frac{28}{28+7} = 80\%$

Specificity= $\frac{\mathrm{T}N}{\mathrm{TN+F}P} = \frac{15}{15+3} = 83\%$

Thank You