Appendix A

Code for Matlab Program

```
clear all;

clc;

close all;

%*****************************************************************
% Get Data & User Inputs
%*****************************************************************
Fileloc = 'C:\MATLAB7\Work\';

Filename = input('Enter ECG File Name = ','s'); % Input Filename

Headerfile = strcat(Filename,'.hea');           % Header In TXT
                                                  Format

load (Filename);              % .mat file for Data

%*****************************************************************
% Load Header Data
%*****************************************************************
fprintf(1,'\nK> Loading Data from Header File %s ...\n',
Headerfile);

signalh = fullfile(Fileloc, Headerfile);

fid1 = fopen(signalh,'r');

z = fgetl(fid1);

A = sscanf(z, '%*s %d %d',[1,2]);

nosig = A(1);                                % Number Of Signals

sfreq = A(2);

clear A;

z = fgetl(fid1);

A = sscanf(z, '%*s %*d %d %d %d %d',[1,4]);

gain = A(1);    % Integers Per mV

clear A;
```

```
S = sfreq*60;
counter1=0;
counter2=0;
counter3=0;
for n = 0:5
    tic
    j = S*n+1:1:S*(n+1);
    D = val(j);
    dat = length (D);
    k = 1:1:dat;
    D = D(k)/gain;
%****************************************************************
%Signal filter and Base line wander correction
%****************************************************************
D= transpose (D);
windowSize = 5;
filsig = filter (ones(1,windowSize)/windowSize,1,D);
y = medfilt1(filsig,200); % 1st median filter
s1 = y;
clear y;
y = medfilt1(s1,600); % 2nd median filter
D = filsig - y;
clear s1;
clear y;
pack;

%****************************************************************
% Manipulate Data So We Only Look At What The User Wants
%****************************************************************
D = transpose (D);
```

```
D = cwt (D, 1:4, 'bior2.4'); %Performing Continuous Wavelet
                              Transform using Biorthogonal
                              Wavelet to ECG_1

D = transpose (D);

x = D (:,4);

clear D;

%**************************************************************
% R-Peak Detection
%**************************************************************
thresh = 0.6;

% create time axis

len = length (x);

tt = 1/sfreq:1/sfreq:ceil(len/sfreq);

t = tt(1:len);

max_h = max (x(round(len/4):round(3*len/4)));%segment search
                                              area first find the
                                              highest bumps in
                                              the ECG_1

poss_reg = x>(thresh*max_h); %then build an array of segments to
                              look in
                             %find indices into boudaries of
                              each segment

left  = find(diff([0 poss_reg'])==1); % remember to zero pad at
                                        start

right = find(diff([poss_reg' 0])==-1); % remember to zero pad at
                                         end

%loop through all possibilities

for(i=1:length(left))

   [maxval(i) maxloc(i)] = max( x(left(i):right(i)) );
```

```
    [minval(i) minloc(i)] = min( x(left(i):right(i)) );
    maxloc(i) = maxloc(i)-1+left(i); % add offset of present
                                       location
    minloc(i) = minloc(i)-1+left(i); % add offset of present
                                       location
end
R_index = maxloc;
R_t    = t(maxloc);
R_amp = maxval;
%****************************************************************
% Heart Rate Calculation
%****************************************************************
for j = 2:length (R_t)
HR(j)= R_t(j)-R_t(j-1);
end
H_R = 60/(mean (HR));
fprintf (1,'\nK> Heart Rate is %d \n',H_R);
%****************************************************************
% S-Point Detection
%****************************************************************
R_len= length (R_index);
for j = 1:R_len
    IR1 = R_index(j);
    for i = IR1:IR1+ (round(sfreq*0.03)*(H_R/72))
        if i == length (x)|i==0
            S_index(j)= 1;
            S_amp(j)  = x(1,1);
            S_t(j)  = t(1,1);
            break
        end
```

```
        if x(i,1)< x(i+1,1) && x(i,1)< x(i-1,1)
            S_index(j)= i;
            S_amp(j) = x(i,1);
            S_t(j) = t(1,i);
            break
        end
    end
end
%****************************************************************
% Q-Point Detection
%****************************************************************
for j = 1:R_len
    IR1 = R_index(j);
    for i = IR1:-1:IR1- (round(sfreq*0.03 *(H_R/72)))
        if i == 0|i==length (x)
            Q_index(j)= 1;
            Q_amp(j) = x(1,1);
            Q_t(j) = t(1,1);
           break
        end
        if x(i,1)< x(i+1,1) && x(i,1)< x(i-1,1)
            Q_index(j)= i;
            Q_amp(j) = x(i,1);
            Q_t(j) = t(1,i);
            break
        end
    end
end
```

```
%*****************************************************************
% J-Point Detection
%*****************************************************************
S_len = length (S_index);
for j = 1:S_len
    IS1= S_index(j);
    for i=IS1:IS1+ (round(sfreq*0.03) *(H_R/72))
        if i==0
            J_index(j)=1;
            J_amp(j)= x(1,1);
            J_t(j)= t(1,1);
            break
        end
        if i> length (x)
            break
        end
        if x(i,1)>=0
            J_index(j)=i;
            J_amp(j)= x(i,1);
            J_t(j)= t(1,i);
            break
        end
    end
end
%*****************************************************************
% T-Peak Detection
%*****************************************************************
J_len = length (J_index);
for j = 1: J_len
    P1 = R_index(j)+ (round(sfreq*0.4) *(H_R/72));
```

```
    P2 = J_index (j)+ (round(sfreq*0.08) *(H_R/72));
    if P1> length (x)|P2> length (x)
        break
    end
    if P1 > P2
        [T_peak(j),T_peak_index(j)] = max(x(P2:P1));
        T_peak_index(j) = T_peak_index(j)+ P2;
    else
        [T_peak(j),T_peak_index(j)] = max(x(P1:P2));
        T_peak_index(j) = T_peak_index(j)+ P1;
    end
    T_peak_t (j)= t(T_peak_index (j));
end



%****************************************************************
% T-Point (T onset) Detection via T-peak
%****************************************************************
T_len = length (T_peak_index);
for j = 1:T_len
    IT1 = T_peak_index(j);
    for i = IT1:-1:IT1-(round(sfreq*0.035) *(H_R/72))
        TP_index(j)=i;
        TP_amp(j)=x(i,1);
        TP_t(j)=t(1,i);
    end
end
```

```
%***************************************************************
% K-Point
%***************************************************************
Q_len = length (Q_index);

for j = 1:Q_len

    IQ1 = Q_index(j);

    for i=IQ1:-1:IQ1- (round(sfreq*0.03) *(H_R/72))

        if i == 0

           K_index(j) = 1;

           K_amp(j)= x(1,1);

           K_t(j)=t(1,1);

           break

        end

        if x(i,1)>=0

           K_index(j) = i;

           K_amp(j)= x(i,1);

           K_t(j)=t(1,i);

           break

        end

    end

    if i == 0

           K_index(j) = 1;

           K_amp(j)= x(1,1);

           K_t(j)=t(1,1);

           break

        end

    if x(i,1)< x(i+1,1) && x(i,1)< x(i-1,1)

           K_index(j) = i;

           if K_index(j)==0
```

```
              K_index(j)=1;

           end

           K_amp(j)= x(i,1);

           K_t(j)=t(1,i);

    end


end


%*****************************************************************
% P-Point Detection via K+80ms
%*****************************************************************
K_len = length (K_index);
for j = 1:K_len

    IK1 = K_index(j);

    for i=IK1:-1:IK1- (round(sfreq*0.08) *(H_R/72))

        if i ==0

            P_index(j) = 1;

            P_amp(j)= x(1,1);

            P_t(j)=t(1,1);

            break

        end

        P_index(j) = i;

        P_amp(j)= x(i,1);

        P_t(j)=t(1,i);

    end

end
```

```
%*************************************************************
% Calculation of Isoelectric Line
%*************************************************************
j = 1:1:K_len;
ISO(j) = mean(x(P_index(j):K_index(j)));

%*************************************************************
% Calculation of ST-segment
%*************************************************************
a = length (J_index);

b = length (TP_index);

if a==b;

    j = 1:1:J_len;

    ST(j) = mean(x(J_index(j):TP_index(j)));

end

if a>b

   j = 1:1:b;

    ST(j) = mean(x(J_index(j):TP_index(j)));

end

if a<b

    j = 1:1:a;

   ST(j) = mean(x(J_index(j):TP_index(j)));

End

%*************************************************************
% Comparison of ISO and ST
%*************************************************************
a = length (ISO);

b = length (ST);

if a==b

    for j = 1:a

        counter1=counter1+1;

       if (ISO(j))>= (ST(j)+0.0001) && ISO(j)>=ST(j)-
0.0001|ISO(j)==ST(j)
```

```
            counter2=counter2+1;
        else
            counter3=counter3+1;
        end
    end
end

if a<b
    for j=1:a
        counter1=counter1+1;
        if ISO(j)>=ST(j)+0.0001 && ISO(j)>=ST(j)-
0.0001|ISO(j)==ST(j)
            counter2=counter2+1;
        else
            counter3=counter3+1;
        end
    end
end

if a>b
    for j=1:b
        counter1=counter1+1;
        if ISO(j)>=ST(j)+0.0001 && ISO(j)>=ST(j)-
0.0001|ISO(j)==ST(j)
            counter2=counter2+1;
        else
             counter3=counter3+1;
        end
```

```
    end
end
clear ISO;
clear ST;
toc
fprintf(1,'\nK> %d loop completed %n \n',n);
end
fprintf (1,'\nK> total number of signals evaluated is %d
\n',counter1)
fprintf (1,'\nK> total number of signals without MI is %d
\n',counter2)
fprintf (1,'\nK> total number of signals with MI is %d
\n',counter3)
if counter3/counter1>=0.95
    fprintf(1,'\nK>WARNING: MI\n');
else
    fprintf(1,'\nK>No MI\n');
end
%***************************************************************
%Plotting Function
%***************************************************************
figure
subplot(2,1,1)
plot(t,x), grid on;
title('Level 4 2^4 Biorthogonal Wavelet Transformed ECG Signal')
ylabel('ECG')
subplot(2,1,2)
plot(t,x,'b');hold on;
plot(S_t,S_amp,'+r'), grid on; hold on;
plot(R_t,R_amp,'+k'); hold on;
```

```
plot (Q_t, Q_amp, '+g'); hold on;

plot (T_peak_t,T_peak, '+y');hold on;

plot (TP_t,TP_amp, '+m');hold on;

plot (J_t,J_amp,'+c');hold on;

plot (K_t,K_amp,'*r');hold on;

plot (P_t,P_amp,'*m');

title('Biorthogonal Wavelet Transformed ECG Signal with Q-Peaks
(green), R-Peaks (black),S-Peaks (red)')

ylabel('ECG+S+R+Q+P+J')

hold off;

fprintf(1,'\nK> Analysis Complete \n');

%****************************************************************
% End of Code
%****************************************************************
```

Appendix B

he Graphical user inerface for the software code developed is as follows:

```
function varargout = GUISTchange(varargin)

gui_Singleton = 1;

gui_State = struct('gui_Name',       mfilename, ...

                   'gui_Singleton',  gui_Singleton, ...

                   'gui_OpeningFcn', @GUISTchange_OpeningFcn,
...

                   'gui_OutputFcn',  @GUISTchange_OutputFcn, ...

                   'gui_LayoutFcn',  [] , ...

                   'gui_Callback',   []);

if nargin && ischar(varargin{1})

    gui_State.gui_Callback = str2func(varargin{1});

end

if nargout

    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});

else

    gui_mainfcn(gui_State, varargin{:});
```

```
end

%*************************************************************

% Get Data & User Inputs

%*************************************************************

function edit1_Callback(hObject, eventdata, handles)

Fname = get(hObject,'string');

handles.Filename = Fname;

guidata(hObject, handles);

function edit2_Callback(hObject, eventdata, handles)

handles.Filename = '';

handles.Result = '';

handles.output = hObject;

guidata(hObject, handles);

function varargout = GUISTchange_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;

Fileloc = 'C:\MATLAB7\Work\';

Headerfile = strcat(handles.Filename,'.hea');           % Header
In TXT Format
```

```
load (handles.Filename);                % .mat file for Data

%*****************************************************************
% Load Header Data
%*****************************************************************

signalh = fullfile(Fileloc, Headerfile);

fid1 = fopen(signalh,'r');

z = fgetl(fid1);

A = sscanf(z, '%*s %d %d',[1,2]);

nosig = A(1);                                   % Number Of Signals

sfreq = A(2);

clear A;

z = fgetl(fid1);

A = sscanf(z, '%*s %*d %d %d %d %d',[1,4]);

gain = A(1);   % Integers Per mV

clear A;

S = sfreq*60;

counter1=0;

counter2=0;
```

```
counter3=0;

for n = 0:4

    j = S*n+1:1:S*(n+1);

    D = val(j);

    dat = length (D);

    k = 1:1:dat;

    D = D(k)/gain;

%*****************************************************************

%Signal filter and Base line wander correction

%*****************************************************************

D= transpose (D);

axes(handles.axes1)

plot (D,'g');

title ('\it{Original Signal for ECG}');

ylabel ('Amplitude in Volts');

xlabel ('# of Samples');

%set(handles.axes1,'XMinorTick','on');

set(handles.axes1,'Color','k');
```

```
drawnow;

windowSize = 5;

filsig = filter (ones(1,windowSize)/windowSize,1,D);

y = medfilt1(filsig,200); % 1st median filter

s1 = y;

clear y;

y = medfilt1(s1,600); % 2nd median filter

D = filsig - y;

axes(handles.axes2)

plot (D,'g');

title ('\it{Filtered Baseline Wander Corrected Signal for
ECG}');

ylabel ('Amplitude in Volts');

xlabel ('# of Samples');

set(handles.axes2,'Color','k');

drawnow;

clear s1;

clear y;
```

```
pack;

%****************************************************************

% Manipulate Data So We Only Look At What The User Wants

%****************************************************************

D = transpose (D);

D = cwt (D, 1:4, 'bior2.4'); %Performing Continuous Wavelet
Transform using Biorthogonal Wavelet to ECG_1

D = transpose (D);

D_1 = D (:,1);

D_2 = D (:,2);

D_3 = D (:,3);

x = D (:,4);

clear D;

%****************************************************************

% R-Peak Detection

%****************************************************************

thresh = 0.6;

% create time axis

len = length (x);
```

```
tt = 1/sfreq:1/sfreq:ceil(len/sfreq);

t = tt(1:len);

max_h = max (x(round(len/4):round(3*len/4)));%segment search
area first find the highest bumps in the ECG_1

poss_reg = x>(thresh*max_h); %then build an array of segments to
look in

left  = find(diff([0 poss_reg'])==1); % remember to zero pad at
start

right = find(diff([poss_reg' 0])==-1); % remember to zero pad at
end

for(i=1:length(left))

    [maxval(i) maxloc(i)] = max( x(left(i):right(i)) );

    [minval(i) minloc(i)] = min( x(left(i):right(i)) );

   maxloc(i) = maxloc(i)-1+left(i); % add offset of present
location

   minloc(i) = minloc(i)-1+left(i); % add offset of present
location

end

R_index = maxloc;
```

```
R_t    = t(maxloc);

R_amp = maxval;

%****************************************************************

% Heart Rate Calculation

%****************************************************************

for j = 2:length (R_t)

HR(j)= R_t(j)-R_t(j-1);

end

H_R = 60/(mean (HR));

%****************************************************************

% S-Point Detection

%****************************************************************

R_len= length (R_index);

for j = 1:R_len

    IR1 = R_index(j);

    for i = IR1:IR1+ (round(sfreq*0.03*(H_R/72)))

        if i == length (x)|i==0

            S_index(j)= 1;

            S_amp(j)  = x(1,1);
```

```
            S_t(j) = t(1,1);
            break
        end
        if x(i,1)< x(i+1,1) && x(i,1)< x(i-1,1)
            S_index(j)= i;
            S_amp(j) = x(i,1);
            S_t(j) = t(1,i);
            break
        end
    end
end
%****************************************************************
% Q-Point Detection
%****************************************************************
for j = 1:R_len
    IR1 = R_index(j);
    for i = IR1:-1:IR1- (round(sfreq*0.03*(H_R/72)))
        if i == 0|i==length (x)|i-1==0
```

```
            Q_index(j)= 1;

            Q_amp(j) = x(1,1);

            Q_t(j) = t(1,1);

           break

        end

        if x(i,1)< x(i+1,1) && x(i,1)< x(i-1,1)

            Q_index(j)= i;

            Q_amp(j) = x(i,1);

            Q_t(j) = t(1,i);

            break

        end

    end

end

%****************************************************************

% J-Point Detection

%****************************************************************

S_len = length (S_index);

for j = 1:S_len
```

```
IS1= S_index(j);

foundj = 0;

for i=IS1:IS1+ (round(sfreq*0.03*(H_R/72)))

    if i==0

        J_index(j)=1;

        J_amp(j)= x(1,1);

        J_t(j)= t(1,1);

        foundj = 1;

        break

    end

    if i > length (x)

        break

    end

    if x(i,1)>=0

        J_index(j)=i;

        J_amp(j)= x(i,1);

        J_t(j)= t(1,i);

        foundj = 1;
```

```
            break
        end
    end
    if foundj == 0
        J_index(j)=1;
        J_amp(j)= x(1,1);
        J_t(j)= t(1,1);
    end
end
%*****************************************************************
% T-Peak Detection
%*****************************************************************
J_len = length (J_index);
for j = 1: J_len
    P1 = R_index(j)+ (round(sfreq*0.4*(H_R/72)));
    P2 = J_index (j)+ (round(sfreq*0.08*(H_R/72)));
    if P1> length (x)|P2> length (x)
        break
```

```
    end
    if P1 > P2
        [T_peak(j),T_peak_index(j)] = max(x(P2:P1));
        T_peak_index(j) = T_peak_index(j)+ P2;
    else
        [T_peak(j),T_peak_index(j)] = max(x(P1:P2));
        T_peak_index(j) = T_peak_index(j)+ P1;
    end
    T_peak_t (j)= t(T_peak_index (j));
end


%****************************************************************
% T-Point (T onset) Detection via T-peak
%****************************************************************
T_len = length (T_peak_index);
for j = 1:T_len
    IT1 = T_peak_index(j);
    for i = IT1:-1:IT1-(round(sfreq*0.035*(H_R/72)))
```

```
        TP_index(j)=i;

        TP_amp(j)=x(i,1);

        TP_t(j)=t(1,i);

    end

end

%*****************************************************************

% K-Point

%*****************************************************************

Q_len = length (Q_index);

for j = 1:Q_len

    IQ1 = Q_index(j);

    foundk = 0;

    for i=IQ1:-1:IQ1- (round(sfreq*0.03*(H_R/72)))

        if i == 0

           K_index(j) = 1;

           K_amp(j)= x(1,1);

           K_t(j)=t(1,1);

           foundk = 1;
```

```
            break
        end
        if x(i,1)>=0
            K_index(j) = i;
            K_amp(j)= x(i,1);
            K_t(j)=t(1,i);
            foundk = 1;
            break
        end
    end
    if foundk == 0
          K_index(j)=1;
          K_amp(j)= x(i,1);
          K_t(j)=t(1,i);
    end
end
```

```
%*****************************************************************
% P-Point Detection via K+80ms
%*****************************************************************
K_len = length (K_index);
for j = 1:K_len
    IK1 = K_index(j);
    for i=IK1:-1:IK1- (round(sfreq*0.08*(H_R/72)))
        if i ==0
            P_index(j) = 1;
            P_amp(j)= x(1,1);
            P_t(j)=t(1,1);
            break
        end
        P_index(j) = i;
        P_amp(j)= x(i,1);
        P_t(j)=t(1,i);
    end
end
```

```
%****************************************************************

% Calculation of Isoelectric Line

%****************************************************************

j = 1:1:K_len;

ISO(j) = mean(x(P_index(j):K_index(j)));

%****************************************************************

% Calculation of ST-segment

%****************************************************************

a = length (J_index);

b = length (TP_index);

if a==b;

    j = 1:1:J_len;

    ST(j) = mean(x(J_index(j):TP_index(j)));

end

if a>b

   j = 1:1:b;

    ST(j) = mean(x(J_index(j):TP_index(j)));

end

if a<b
```

```
    j = 1:1:a;

   ST(j) = mean(x(J_index(j):TP_index(j)));

end

%****************************************************************

% Comparison of ISO and ST

%****************************************************************

a = length (ISO);

b = length (ST);

if a==b

    for j = 1:a

        counter1=counter1+1;

       if (ISO(j)>= ST(j)+0.0001 && ISO(j)>=ST(j)-
0.0001)|ISO(j)==ST(j)

           %fprintf(1,'\nK>No MI\n');

           counter2=counter2+1;

       else

           counter3=counter3+1;

        end
```

```
    end
end
if a<b
    for j=1:a
        counter1=counter1+1;
        if (ISO(j)>=ST(j)+0.0001 && ISO(j)>=ST(j)-
0.0001)|ISO(j)==ST(j)
           counter2=counter2+1;
        else
           counter3=counter3+1;
        end
    end
end


if a>b
    for j=1:b
        counter1=counter1+1;
```

```
        if (ISO(j)>=ST(j)+0.0001 && ISO(j)>=ST(j)-
0.0001)|ISO(j)==ST(j)
           counter2=counter2+1;
        else
            counter3=counter3+1;
        end
    end
end
clear ISO;
clear ST;
if counter3/counter1>=0.90
    fprintf(1,'\nK>WARNING: Possible MI\n');
else
    fprintf(1,'\nK>No MI\n');
end
axes(handles.axes3)
plot(t,x,'g');hold on;
plot(S_t,S_amp,'+r'), hold on;
```

```
plot(R_t,R_amp,'+w'); hold on;

plot (Q_t, Q_amp, '+b'); hold on;

plot (T_peak_t,T_peak, '+y');hold on;

plot (TP_t,TP_amp, '+m');hold on;

plot (J_t,J_amp,'+c');hold on;

plot (K_t,K_amp,'Marker','+','color',[1,0.4,0.6]);hold on;

plot (P_t,P_amp,'Marker','+','color',[0.4,0,0.6]);

title('Biorthogonal Wavelet Transformed ECG Signal with Q-Peaks
(green), R-Peaks (black),S-Peaks (red)')

ylabel('ECG+S')

hold off;

ylabel ('Amplitude in Volts');

xlabel ('Time in seconds');

set(handles.axes3,'Color','k');

drawnow;

end

if counter3/counter1>=0.90

    fprintf(1,'\nK>WARNING: Call 9-1-1 & Get Help\n');
```

```
    handles.Result = 'WARNING: Call 9-1-1 & Get Help';

else

    fprintf(1,'\nK>No ST-Changes Detected\n');

    handles.Result = 'No ST-Changes Detected';

end

guidata(hObject, handles);

set(handles.edit2,'String',handles.Result);

function edit1_CreateFcn(hObject, eventdata, handles)

if ispc

    set(hObject,'BackgroundColor','white');

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundC
olor'));

end

function edit2_CreateFcn(hObject, eventdata, handles)

if ispc

    set(hObject,'BackgroundColor','White');
```

```
else
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundC
olor'));
end
function figure1_ResizeFcn(hObject, eventdata, handles)
%*****************************************************************
% End Of Code
%*****************************************************************
```