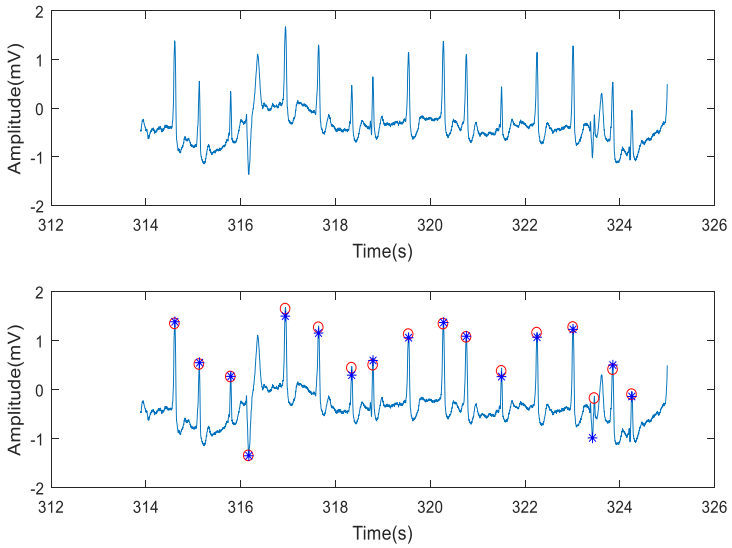
Digital Signal Processing Case
By: Nezar Assawiel
==========

This docment shows the code and sample results of an "optimized" MatLab implimentation of:

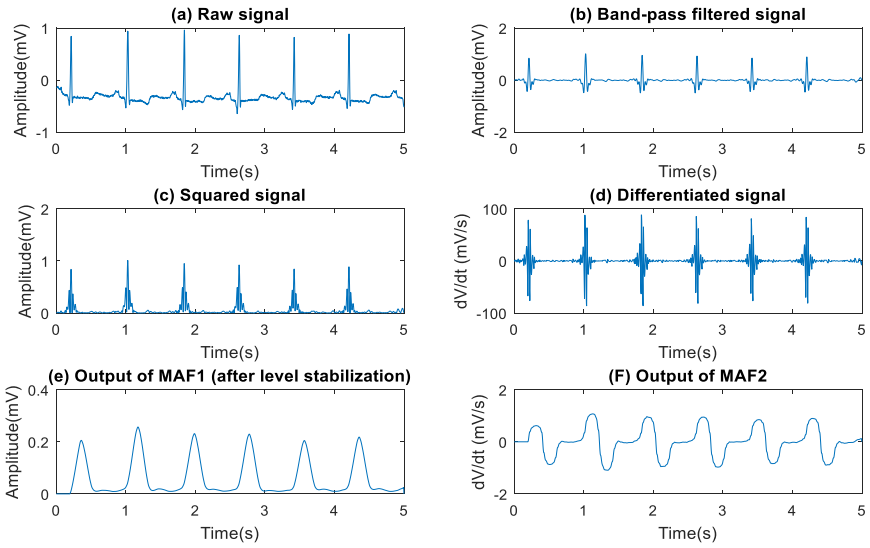
Jinkwon Kim and Hangsik Shin, "Simple and Robust Realtime QRS Detection Algorithm based on Spatiotemporal Characteristic of the QRS Complex", PlosOne, 2016.

Sample Result:



Results from this algorithm (*) compared to the Pan Tompkings algorithm (O) using a segment of record 105 from the RMIT-BIH arrhythmia database.

Processing blocks of the alogrithm:



```
function [loc, time] = Kim_Shin(data,Fs,rtime)
%%
% This code is an implementation for:
% Jinkwon Kim and Hangsik Shin, "Simple and Robust Realtime QRS Detection Algorithm based on
% Spatiotemporal Characteristic of the QRS Complex", PlosOne, 2016 (Note:
% this implimentation uses slightly different filter and slightly different pramaters values)

%Inputs:
% data: ecg data as one vector (one channel ecg)
% Fs: smapling freq
% rtime: reference time

%output:
% loc: indices of QRS
% time: time location of QRS

% The alogrithm detects QRS withing ECG in three steps: 1) energy level detection 2) Energy variation detection
% and 3)Adjusting weights. Please refer to the paper to understand how the algorithm works


%-----------------------------------
% Written By: Nezar Assawiel
% nezar.assawiel@mail.utoronto.ca
% date: March 3, 2017
% Version: 1.0
%-----------------------------------


%% Parameters

r_a = 0.1;    %application rate for weight adjustment
r_b = 0.05;   %application rate for weight adjustment
r_nr = 1.75;  %application rate of noise level (for signal threshold)
r_s = 0.001;  %application rate of signal level (for noise threshold)
r_d = 0.09;   %decay rate for adaptive threshold
r_n = 0.03;   %application rate of noise level (for signal threshold)

size_maf1 = 0.15;     %size of 1st MAF
size_maf2 =0.2;       %size of 2nd MAF
refract_time = 0.15;  %refractory period length
thEL0 = 0.1;          %threshold for Energy Level Detection
stabLvel = 0.5;       %pre-defined stable level reference for level stability
Weight = 1;           %weight for level stability adjustment


%% window parameters
win_size_EL = round(size_maf1*Fs);          %window size for Energy Level(EL) detection
win_size_EV = round(size_maf2*Fs);          %window size for Energy Variation (EV) detection
diff_win_size = win_size_EV - win_size_EL;  %diff in window size between EL and EV
refract_p = round(refract_time*Fs);         %refractory period

thEVlimit = 1*Fs/(0.2*Fs*20);               %threshold limit for EV
thEV_ub = 0.45*Fs/(0.2*Fs*20);              %EV threshold upper bound
thEV_lb = -0.45*Fs/(0.2*Fs*20);             %EV threshold lower bound
thEV_ub2 = 20*Fs/(0.2*Fs*20);               %EV threshold upper bound 2
thEV_lb2 = -20*Fs/(0.2*Fs*20);              %EV threshold lower bound 2

AryLen = 5;
decayF = 1-1/( (0.40-refract_time)*Fs );    % decay factor


%% Pre-processing
% data -> fsig -> ssig -> dsig as shown below:

% IIR filter
f1=5;                   %cuttoff low frequency to get rid of baseline wander
f2=25;                  %cuttoff frequency to discard high frequency noise
Wn=[f1 f2]*2/Fs;        %cutt off based on fs
N = 4;                  %order of 4 (less processing)
[a,b] = butter(N,Wn);   %bandpass filtering

fsig = filtfilt(a,b,data);   %filtered siganl

ssig=sqrt(fsig.^2);         %squared signal

dsig=[0; diff(ssig)]*Fs;    %differentiated signal
```

```
%% Memory Allcation
sig_len = length(ssig);

ELqrs = zeros(sig_len,1);
EVqrs = zeros(sig_len,1);

thEL = zeros(sig_len,1) * thEL0;
thEV = zeros(sig_len-1,1);

max_buf = zeros(AryLen,1);
min_buf = zeros(AryLen,1);

BUF1=zeros(win_size_EL,1);
BUF2=zeros(win_size_EV,1);

thN = zeros(sig_len,1);
max_v_arry = zeros(AryLen,1);

%% Processing


large_win = win_size_EV;
max_v=1;
count_qrs = 0;
checker2=0;
loc=[];
is_begin=0;

for q=large_win:sig_len-1

    % MAF with weight
    BUF1(q,1)=sum( ssig(q-win_size_EL+1:q,1) )/ win_size_EL;
    BUF2(q,1)=sum( dsig(q-win_size_EV+1:q,1) )/ win_size_EV;

    ELqrs(q,1) = sum( BUF1(q-win_size_EL+1:q,1) )/win_size_EL;
    EVqrs(q,1) = sum( BUF2(q-win_size_EV+1:q,1) )/win_size_EV;

    % STEP 1: ENERGY LEVEL DETECTION
    if (is_begin==0) && (ELqrs(q) >= thEL(q))
        thEL(q)=ELqrs(q);
        max_v=ELqrs(q);
        maxP=q;
        is_begin = 1;
    end
    if(ELqrs(q)<thN(q))
        thN(q)=ELqrs(q);
    end
    if(is_begin)
        if (ELqrs(q)>=max_v)
            thEL(q)=ELqrs(q);
            max_v=ELqrs(q);
            maxP=q;
            timer=refract_p;
        else
            timer=timer-1;
            thEL(q)=max_v;
            if(timer==0)
                is_begin=0;
                checker2=1;
                time_peak=win_size_EV-(refract_p-win_size_EL);
                max_p_buf=maxP;
                max_v_buf=max_v;
            end
        end
    end

    % STEP 2: ENERGY VARIATION DETECTION
    if(checker2==1)
        time_peak=time_peak-1;
        if(time_peak==0)
            checker2=0;
            if max_p_buf-win_size_EL<1
                BufStartP2=1;
            else
                BufStartP2=max_p_buf-win_size_EL;
            end
            if max_p_buf+2*diff_win_size>sig_len
```

```
                buf_end=size(M,1);
            else
                buf_end=max_p_buf+2*diff_win_size*2;
            end

            DiffSumCheck1=max(EVqrs(BufStartP2:max_p_buf+diff_win_size,1));
            DiffSumCheck2=min(EVqrs(max_p_buf+diff_win_size:buf_end,1));

            if ( isempty(loc) || (DiffSumCheck1-DiffSumCheck2>thEVlimit)&&(DiffSumCheck1*DiffSumCheck2<0)&&...

(DiffSumCheck1>thEV_ub)&&(DiffSumCheck2<thEV_lb)&&(DiffSumCheck1<thEV_ub2)&&(DiffSumCheck2>thEV_lb2) )
                count_qrs = count_qrs + 1;

                loc=[loc; max_p_buf-win_size_EL+2];

                % STEP 3: WEIGHT ADJUSTMENT
                max_v_arry(mod(count_qrs,AryLen)+1,1)=max_v_buf;
                max_buf(mod(count_qrs,AryLen)+1,1)=max(EVqrs(BufStartP2:buf_end,1));
                min_buf(mod(count_qrs,AryLen)+1,1)=min(EVqrs(BufStartP2:buf_end,1));
                if stabLvel>mean(max_v_arry)
                    adujR=min((stabLvel-median(max_v_arry))*r_a, stabLvel*r_b);
                else
                    adujR=min((stabLvel-median(max_v_arry))*r_a, stabLvel*-r_b);
                end
                Weight=Weight+adujR;
            end
        end
    end
    thN(q+1)=thN(q)+r_s*ELqrs(q);
    if(exist('maxV_Buf','var'))
        thEL(q+1)=thEL(q)* ( decayF  * (1-r_d*(thN(q)/max_v_buf))) + r_n*thN(q);
    else
        thEL(q+1)=thEL(q)* ( decayF);
    end
    if thEL(q+1)<r_nr*thN(q)
        thEL(q+1)=r_nr*thN(q);
    end
end

time=rtime(loc);
end
```