# Dedykowane algorytmy diagnostyki medycznej
# Raport cząstkowy
# Detekcja zespołu QRS z wykorzystaniem Dynamic Plosion Index

Przemysław Węgrzynowicz, Agnieszka Żądło

22 grudnia 2016

## 1 Wprowadzenie

### 1.1 Detekcja zespołów QRS

Detektor zespołów QRS w sygnale elektrokardiograficznym ma za zadanie znaleźć charakterystyczne ewolucje związane z faktem depolaryzacji komór serca. Najczęściej detektory zwracają indeksy próbek sygnału, które wyznaczają wystąpienie danego zjawiska. Depolaryzacja komór jest zjawiskiem elektrofizjologicznym powodującym skurcz mięśnia sercowego oraz pompowanie krwi do aorty oraz do pnia płucnego. Algorytmom do detekcji QRS stawia się szereg założeń. Przede wszystkim powinien on oznaczać jedynie zespoły QRS. Co więcej każdy z zespołów QRS powinien być oznaczony dokładnie jeden raz. Dodatkowo wyznaczony przez detektor punkt powinien leżeć w obrębie zespołu QRS oraz dla identycznych dwóch zespołów odległość punktu od początku zespołu QRS powinna być taka sama. Analiza sygnału EKG w kontekście wyznaczania zespołów QRS jest utrudniona ze względu na możliwość wystąpienia zespołów o różnej morfologii. Częstym problemem w analizie sygnału jest również zjawisko pływania izolinii oraz przydźwięki pochodzące od sieci elektrycznej.

### 1.2 Dynamic Plosion Index

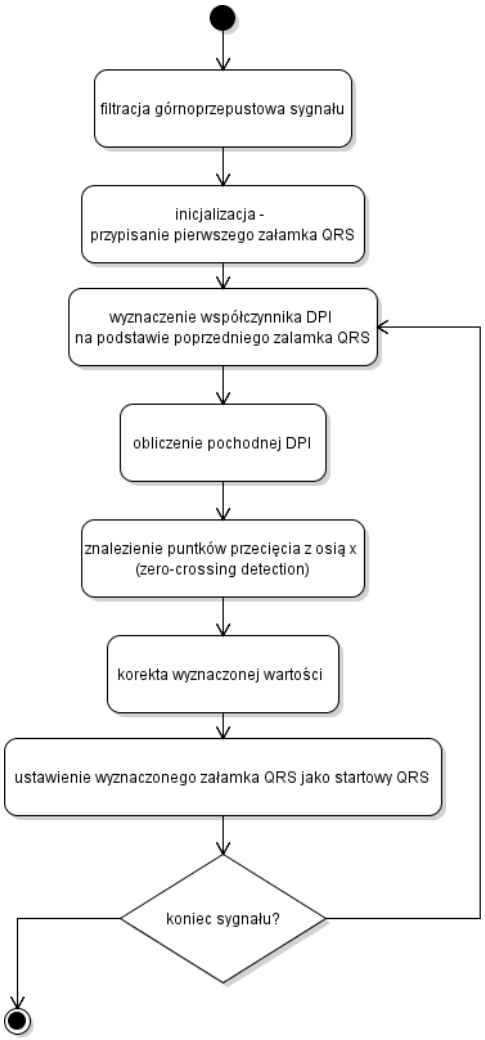
Pierwotnym zastosowaniem metody Dynamic Plosion Index (DPI) była detekcja epok w sygnale mowy, która została opisana w [1]. Metoda pozwala na wyznaczenie lokalnej cechy czasowej sygnału w oparciu o sąsiadujące próbki. Ze względu na podobieństwa dynamiki sygnału mowy oraz elektrokardiograficznego zastosowano modyfikację DPI do wykrywania zespołów QRS. Proponowane rozwiązanie jest metodą detekcji bezprogowej. Jest to szczególnie ważne, gdy algorytm ma działać poprawnie dla różnych urządzeń i w różnych środowiskach akwizycji sygnału EKG.

## 2 Opis algorytmu

Zaproponowana metoda [2] składa się z dwóch głównych etapów. Pierwszym z nich jest wstępne przetwarzanie polegające na przefiltrowaniu EKG filtrem górnoprzepustowym oraz na jednopołówkowym wyprostowaniu sygnału (ang. *half-wave*). Głównym krokiem algorytmu jest sekwencyjna lokalizacja kolejnych zespołów QRS na podstawie Dynamic Plosion Index wyliczanego od punktu należącego do poprzedniego zespołu. Schemat algorytmu przedstawiono na diagramie (Rys.1).

### 2.1 Przetwarzanie wstępne sygnału

Przed operacją właściwej detekcji zespołów QRS zazwyczaj dokonuje się filtracji pasmowo przepustowej z częstotliwościami odcięcia 8 do 20 Hz (30 Hz). Autorzy metody proponują wykorzystanie jedynie filtracji górnoprzepustowej z częstotliwością odcięcia równą 8 Hz. Zaproponowano filtrację w dziedzinie częstotliwości z wykorzystaniem narastającej funkcji cosinus w przedziale 0 do $f_c$. Filtr w dziedzinie




Rysunek 1: Schemat działania bezprogowego algorytmu detekcji zespołów QRS opartego o Dynamic Plosion Index

częstotliwości można przedstawić jako:

$$H(f) = \begin{cases} [0.5 - 0.5 \cdot \cos(\pi f / f_c)] & 0 \leqslant f \leqslant f_c \\ 1 & f_c < f \leqslant f_s/2 \end{cases} \tag{1}$$

Filtrację górnoprzepustową zaimplementowano w środowisku MATLAB z wykorzystaniem wbudowanej funkcji szybkiej transformaty Fouriera oraz transformacji odwrotnej. Drugim etapem wstępnego przetwarzania jest usunięcie składowych o wartościach ujemnych i pozostawienie tylko dodatnich wartości przefiltrowanego sygnału EKG. Prostowanie jednopołówkowe może wprowadzać komplikacje w przypadku ujemnego zespołu QRS, który występuje na przykład dla odprowadzenia aVR. Autorzy zakładają, że amplituda załamka S jest wówczas wystarczająca aby wyznaczyć pozycję zespołu QRS.

## 2.2 Plosion Index

Plosion Index (PI) jest parametrem, który leży u podstawy opisywanej metody. Jego działanie polega na obliczeniu stosunku wartości bezwzględnej aktualnej próbki $s(n_0)$ do wartości średniej próbek z zadanego przedziału $(m_1 - m_2)$ [1]. W przypadku sygnały EKG wysoka wartość współczynnika PI będzie wyznaczać miejsce poszukiwanego zespołu. Wówczas uśredniany przedział $(m_1 - m_2)$ będzie zawierał

próbki izolinii lub część załamka T. Formalną definicję Plosion Index można przedstawić jako:

$$PI(n_0, m_1, m_2) = \frac{|s(n_0)|}{s_{avg}(n_0, m_1, m_2)} \tag{2}$$

$$gdzie \quad s_{avg}(n_0, m_1, m_2) = \frac{\Sigma_{i=n_0+m_1+1}^{i=n_0+m1+m_2} |s(i)|}{m_2} \tag{3}$$

Wartości $m_1$ oraz $m_2$ są dobierane zależnie od jakości sygnału oraz charakteru analizowanych zjawisk. Plosion Index nie jest używany w tej postaci w przedstawianym algorytmie.

## 2.3 Dynamic Plosion Index

Rozwinięciem wcześniej omówionego parametru jest Dynamic Plosion Index (DPI). Można stwierdzić, że algorytm ten to sekwencje wyznaczonych współczynników PI dla następujących po sobie wartości m2, przy stałej wartości m1. Dodatkowo wprowadzono niewielką modyfikację równania (3). Wyliczane wartości sumy są dzielone przez $m_2^{1/p}$, co wprowadza nieliniowe skalowanie współczynników PI w funkcji odległości od punktu startowego $n_0$. Modyfikacja wprowadza wzmocnienie pików znajdujących się bliżej punktu $n_0$.

$$s'_{avg}(n_0, m_1, m_2) = \frac{\Sigma_{i=n_0+m_1+1}^{i=n_0+m1+m_2} |s(i)|}{m_2^{1/p}}, p > 1 \tag{4}$$

## 2.4 Wyznaczanie pozycji załamków QRS

### 2.4.1 Inicjalizacja

Prezentowany detektor QRS działa sekwencyjnie w oparciu o bieżącą pozycję ostatniego zespołu. W pierwszym obrocie pętli zakłada się pozycję QRS w 3 próbce sygnału. Arbitralne przypisanie pozycji pierwszego zespołu QRS jest pod koniec analizy usuwane.

### 2.4.2 Wyznaczenie współczynnika DPI

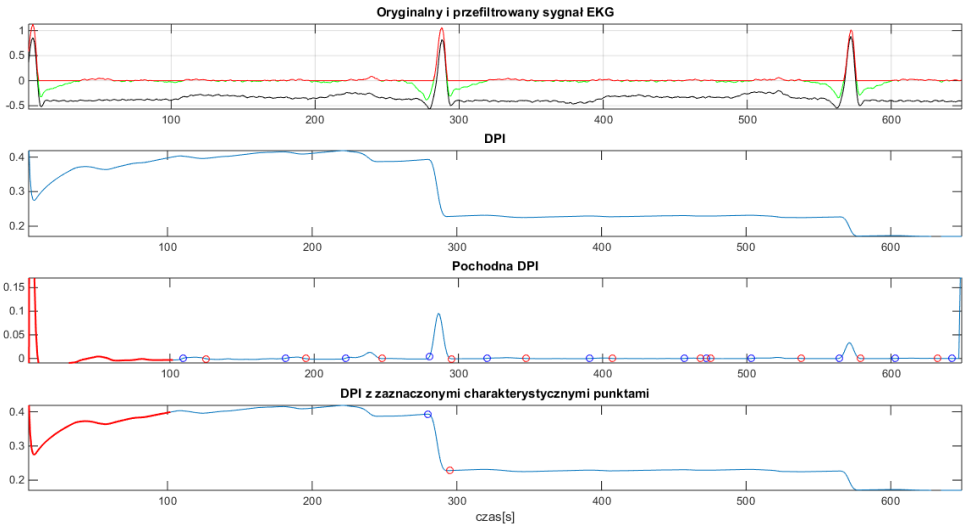
Dla zadanego okna wyliczana jest wartość współczynnika DPI. Wartość $m_1$ ustawiono na wartość -2, zgodnie z propozycją autorów [2]. Oznacza to, że współczynniki DPI będą wyznaczane począwszy od zespołu QRS zawierając sam załamek R (przesunięcie o 2 próbki w lewo). Parametr $m_2$ jest wprowadzany przez użytkownika jako okno obliczeń. W przypadku analizy QRS dobrano wartość 1800ms, co odpowiada w przybliżeniu maksymalnej wartości odstępu R-R (rytm 35 uderzeń na minutę). Współczynniki DPI są wyznaczane zatem dla przedziału 0 do 1800ms na podstawie przefiltrowanego i wyprostowanego jednopołówkowo sygnału EKG. Wynikiem obliczeń jest charakterystyczny przebieg DPI w postaci kaskadowych dolin, co zostało przedstawione na Rys. 2.

### 2.4.3 Analiza pochodnej DPI

Najważniejszy w kontekście znalezienia zespołu QRS jest wyraźny przeskok góra-dolina w DPI (ang. *peak-valley*). W celu wyznaczenia lokalizacji tego obszaru wylicza się prostą pochodną DPI. Następnie wyznacza się punkty przecięcia pochodnej z zero oraz grupuje się te punkty ze względu na polaryzację zbocza (ang. *positive and negative zero-crossing*).

### 2.4.4 Parametr swing

Wyznaczone ekstrema przebiegu DPI są w dalszej części analizowane w celu znalezienia maksymalnego zbocza. Wylicza się dla każdej pary punktów o dodatnim i ujemny zboczu pochodnej prosty parametr - swing. Jest on wartością bezwzględną różnicy odpowiednich wartości DPI. W ten sposób wyznacza się parę góra-dolina, która reprezentuje region występowania zespołu QRS.



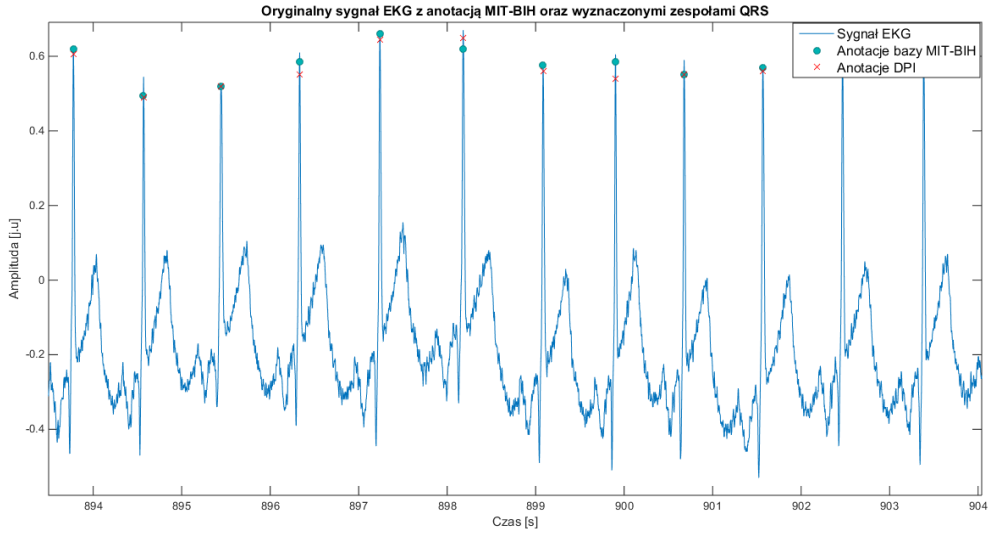

Rysunek 2: Przebieg sygnału, parametru DPI oraz pochodnej DPI dla analizowanego okna

### 2.4.5 Korekta lokalizacji QRS

Ostatnim etapem analizy jest poprawa wyznaczonego położenia zespołu QRS na podstawie wartości bezwzględnej oryginalnego sygnału EKG przefiltrowanego górnoprzepustowo z granicą odcięcia 2 Hz. Zastosowanie wartości bezwzględnej pozwala wyznaczyć położenie załamka R również w przypadku jego odwrócenia. Często w algorytmie wykorzystuje się okno o długości $\pm 285ms$, co odpowiada minimalnemu interwałowi R-R (210 uderzeń na minutę) Lokalizacja zespołu QRS staje się punktem startowym dla kolejnej iteracji algorytmu.

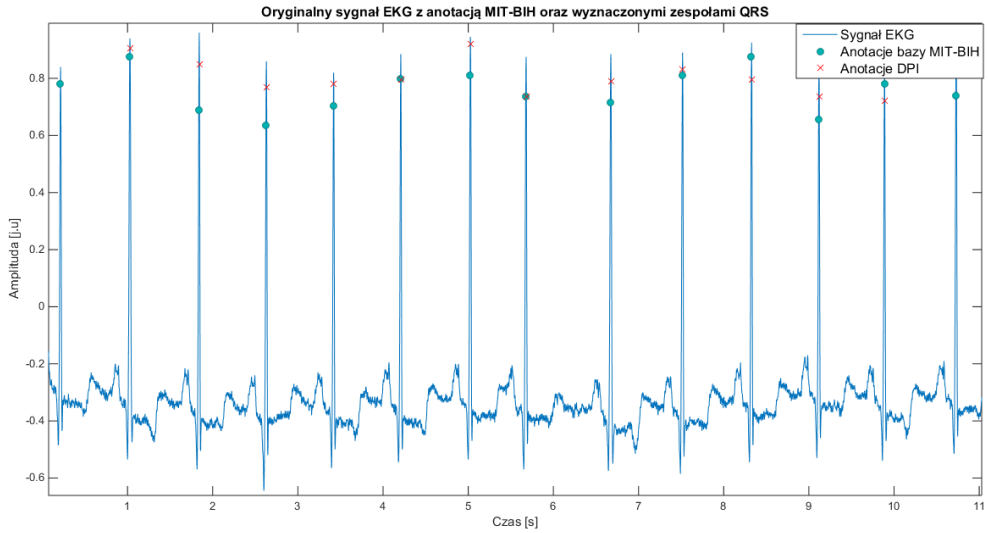
## 3 Otrzymane rezultaty

Zaimplementowany prototyp w środowisku MATLAB został przetestowany na sygnałach pochodzących z arytmicznej bazy MIT-BIH [3]. Porównano wyznaczone punkty detekcji QRS w odniesieniu do dostępnych anotacji (Rys.3). Podczas porównania wykorzystano parametr okna, który określał akceptowalny błąd detekcji. Parametr ustawiono na 40 próbek, co odpowiada około 111ms. Zgodnie z implementacją funkcji walidacyjnej bxb udostępnianej przez PhysioNet wartość ta może być zwiększona nawet do 150ms.



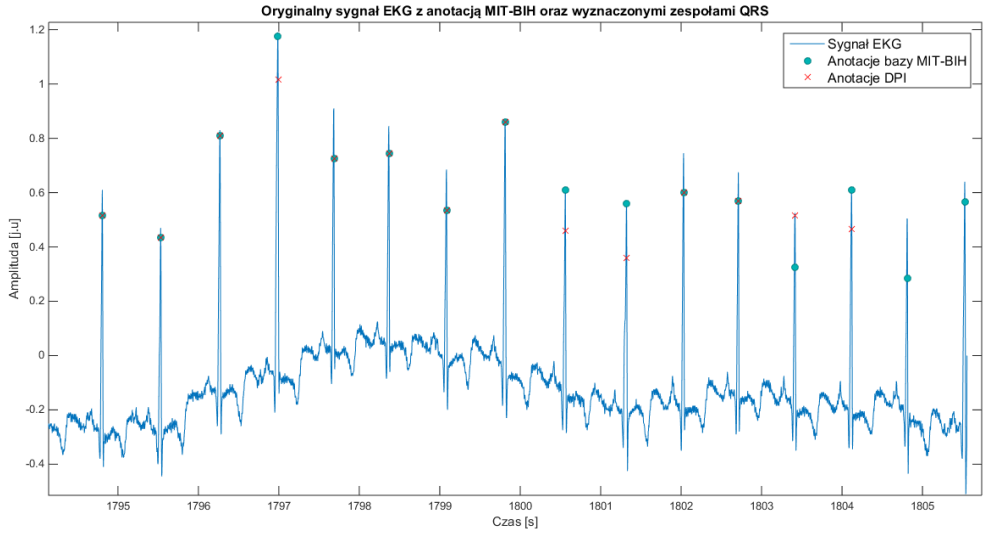

Rysunek 3: Fragment zapisu sygnału EKG wraz z anotacją z bazy MIT-BIH oraz wyznaczonymi zespołami QRS

Ze względu na specyfikę algorytmu DPI utrudnione jest znalezienie pierwszego zespołu QRS w sygnale. Dla przyjętego okna analizy warunki detekcji spełniał dopiero kolejny QRS, co można zaobserwować na Rys.4.




Rysunek 4: Początkowy fragment zapisu sygnału EKG wraz z anotacją z bazy MIT-BIH oraz wyznaczonymi zespołami QRS.

Można także zauważyć, że algorytm nie wykrywał dwóch ostatnich zespołów QRS sygnału elektrokardiograficznego (Rys.5). W założeniu metoda wymaga istnienia określonej liczby próbek w czasie sygnału ze względu na przyjęte okno analizy (1800 ms).




Rysunek 5: Końcowy fragment zapisu sygnału EKG wraz z anotacją z bazy MIT-BIH oraz wyznaczonymi zespołami QRS.

Dokonano również oszacowania skuteczności oraz czułości prototypu dla przykładowych sygnałów przy założeniu akceptowanego błędu równego 40 próbek. Wyniki zebrano w Tabelach 1-2.

Tablica 1: Średnia skuteczność oraz czułość detekcji dla bazy MIT-BIH

| Nagrania MIT-BIH | Skuteczność [%] | Czułość[%] |
|---|---|---|
| 100 - 234 | 96.06 | 96.93 |

Tablica 2: Walidacja detektora QRS dla nagrań bazy MIT-BIH

| Nagranie MIT-BIH | Skuteczność [%] | Czułość[%] |
|---|---|---|
| 100 | 99.82 | 99.82 |
| 101 | 99.31 | 99.41 |
| 102 | 99.73 | 99.73 |
| 103 | 99.62 | 99.62 |
| 104 | 95.37 | 96.19 |
| 105 | 93.10 | 94.72 |
| 106 | 96.14 | 96.23 |
| 107 | 95.94 | 97.06 |
| 108 | 92.22 | 95.50 |
| 109 | 99.61 | 99.61 |
| 111 | 98.93 | 99.34 |
| 112 | 99.49 | 99.49 |
| 113 | 99.83 | 99.83 |
| 114 | 96.20 | 98.99 |
| 115 | 99.54 | 99.54 |
| 116 | 98.36 | 99.13 |
| 117 | 99.48 | 99.68 |
| 118 | 98.91 | 98.91 |
| 119 | 94.84 | 94.84 |
| 121 | 99.20 | 99.20 |
| 122 | 99.76 | 99.76 |
| 123 | 99.80 | 99.80 |
| 124 | 98.96 | 99.02 |
| 200 | 92.78 | 92.98 |
| 201 | 93.51 | 95.39 |
| 202 | 98.06 | 98.70 |
| 203 | 91.30 | 92.54 |
| 205 | 99.03 | 99.03 |
| 207 | 85.92 | 88.26 |
| 208 | 95.47 | 95.76 |
| 209 | 98.46 | 98.46 |
| 210 | 96.36 | 96.54 |
| 212 | 99.38 | 99.38 |
| 213 | 98.45 | 98.45 |
| 214 | 98.22 | 98.30 |
| 215 | 98.71 | 98.74 |
| 217 | 96.54 | 96.62 |
| 219 | 92.94 | 93.43 |
| 220 | 98.89 | 98.89 |
| 221 | 98.30 | 98.42 |
| 222 | 94.02 | 94.23 |
| 223 | 98.15 | 98.15 |
| 228 | 89.61 | 90.61 |
| 230 | 91.40 | 91.40 |
| 231 | 78.07 | 78.07 |
| 232 | 76.34 | 97.91 |
| 233 | 97.46 | 97.46 |
| 234 | 99.53 | 99.53 |

# 4 Bibliografia


[1] A. P. Prathosh, T. V. Ananthapadmanabha i A. G. Ramakrishnan. „Epoch Extraction Based on Integrated Linear Prediction Residual Using Plosion Index". W: *IEEE Transactions on Audio, Speech, and Language Processing* 21.12 (grud. 2013), s. 2471–2480. DOI: 10.1109/TASL.2013.2273717.

[2] A. G. Ramakrishnan, A. P. Prathosh i T. V. Ananthapadmanabha. „Threshold-Independent QRS Detection Using the Dynamic Plosion Index". W: *IEEE Signal Processing Letters* 21.5 (maj 2014), s. 554–558. DOI: 10.1109/LSP.2014.2308591.

[3] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng i H. E. Stanley. „PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals". W: *Circulation* 101.23 (2000), e215–e220.

# Zaimplementowane funkcje w środowisku MATLAB

## 4.1 Filtracja górnoprzepustowa

```
function signal_filtrered = hpf(signal,fc,fs)
%HPF function performs hiph-pass filtereing in the freqency domain
%
% Inputs
% ------
%       signal - one-dimensional vector containing signal form one lead
%       fc - cut off frequency (in Hz)
%       fs - sampling frequency (in Hz)
%
% Outputs
% ------
%       signal_filtered - one-dimensional vector containing filtered signal
%
% Dependencies
% ------------
%
%   This function calls:
%       - fft (MATLAB)
%       - ifft (MATLAB)
%
%   This function is called by:
%       - dpi_based_qrs_detector
%
%
% Authors:    P. Wegrzynowicz, A. Zadlo
% Copyright:  .
% Date:       2016-12-02

if mod(numel(signal),2) ~= 0
    signal=signal(1:end-1);end

freq_point= fs/length(signal);
n_points_to_fc= floor(fc/freq_point);

raised_cosine= 0.5-0.5*cos(pi.*(1:n_points_to_fc)/(n_points_to_fc));
freq_domain_filter  = ones(numel(signal),1);
freq_domain_filter(1:n_points_to_fc) = raised_cosine;
freq_domain_filter(end-n_points_to_fc+1:end) = fliplr(raised_cosine);
freq_domain_filter = freq_domain_filter.^4;

spec= fft(signal);
fft_hpf= freq_domain_filter .* spec;
sig_hpf = ifft(fft_hpf);
signal_filtrered= real(sig_hpf);
end
```

## 4.2 Znajdywanie punktów przecięcia zera

```
function [idx_neg,idx_pos]=zero_crossing(der,th)
%ZERO_CROSSING finds indices of zero-crossing moments
%
% Inputs
% ------
%       der - one-dimensional vector with signal (eg.derivative)
%       th  - threshold used to zero tolerance estimation
%
% Outputs
% ------
%       idx_neg - indices of zero-crossing with negative slope
%       idx_pos - indices of zero-crossing with positive slope
%
% Dependencies
```

```
% ------------
%
%   This function calls:
%
%   This function is called by:
%       - dpi_based_qrs_detector
%
%
% Authors:    P. Wegrzynowicz, A. Zadlo
% Copyright:  .
% Date:       2016-12-02
idx= (1:numel(der)-1);
sign = der(1:end-1).*der(2:end);
moments=sign<=0 ;
positive = der > th;
negative = der < -th;
idx_pos = moments.*positive(1:end-1).*idx;
idx_neg = moments.*negative(1:end-1).*idx;
idx_pos = nonzeros(idx_pos)';
idx_neg = nonzeros(idx_neg)';
end
```

## 4.3 Implementacja detektora QRS w oparciu o DPI

```
function qrs = dpi_based_qrs_detector(signal,fs,window,p,figures)
%DPI_BASED_QRS_DETECTOR function find QRS complexes in ECG signal and
%returns indices of them
%
% Inputs
% ------
%       signal - one-dimensional vector containing signal form one ECG lead
%       fs - sampling frequency (in Hz)
%       window - computation window fo DPI algorithm
%       p - dynamic scaling value for DPI algorithm
%       figures - decision to plot steps of detection (bool)
%
% Outputs
% ------
%       qrs - indices of QRS complexes in ECG signal
%
% Dependencies
% ------------
%
%   This function calls:
%       - hpf
%       - zero_crossing
%       - smooth (MATLAB)
%       - conv (MATLAB)
%
%   This function is called by:
%
%
% Authors:    P. Wegrzynowicz, A. Zadlo
% Copyright:  .
% Date:       2016-12-02
if size(signal,1) == 1
    signal=signal';
end
if mod(length(signal),2) ~= 0
    signal=signal(1:end-1);
end

%% Highpass filtering in the frequency domain (fc = 8 Hz)
signal_filt = hpf(signal,8,fs);
% Highpass filtering in the frequency domain (fc = 2 Hz)
signal_filt_2Hz = hpf(signal,2,fs);
%% DPI
fs_kHz=fs/1000;                         %Sampling frequency in kHz
```

```
ms_285 =floor(285*fs_kHz);               %The period of the highest possible heart rate (210 BPM)
ms_wnd_285=floor((window+285)*fs_kHz);   %Computation window with additional 285 ms
ms_wnd=floor(window*fs_kHz);             %Computation window

%Triangle matrix of DPI denominators:
vec = 1./(1:ms_wnd+1).^(1/p);
dpi_denom = repmat(vec,[ms_wnd+1,1])';
dpi_denom = tril(dpi_denom);


qrs(1) = 5;
m=2;
i=1;
while i< length(signal_filt)-ms_wnd_285
    %% Cut the part of the signal
    sig_part_f8hz= signal_filt(qrs(m-1)-4:qrs(m-1)-4+ms_wnd);

    %% Half wave
    hhecg = sig_part_f8hz.*(sig_part_f8hz>=0);

    %% Dynamic Plosion Index
    dpi= 1./(dpi_denom * hhecg);
    dpi=smooth(dpi);

    %% DPI derivative
    der = conv(dpi,[-1 0 0 0 1],'same')';

    %% Zero crossing points
    [idx_neg,idx_pos]= zero_crossing(der(ms_285:end),0);
    % Shift inidces
    idx_pos=idx_pos+ms_285;
    idx_neg=idx_neg+ms_285;

    % Remove first zero-crossing index if it is positive
    % (current R peak infuence)
    if idx_pos(1) < idx_neg(1)
        idx_pos(1)=[];
    end

    %% Finding pairs
    n_pairs = min(numel(idx_pos),numel(idx_neg));
    % Find pair with maximum swing value
    swing = abs(dpi(idx_neg(1:n_pairs)) - dpi(idx_pos(1:n_pairs)));
    [~,order] = sort(swing,'descend');
    idx_pos_sort = idx_pos(order);
    idx_neg_sort = idx_neg(order);
    idx_pos_sort = idx_pos_sort(idx_pos_sort>ms_285+1);
    if ~isempty(idx_pos_sort)
        idx = idx_pos_sort(1);
    else
        idx=200;
        warning('QRS was artificially shifted - 200 samples')
    end

    %% Absolute indices
    idx = idx+qrs(m-1);
    s = max([1,idx-ms_285]);
    e = min([idx+ms_285,numel(signal_filt_2Hz)]);
    [~,shift] = max(abs(signal_filt_2Hz(s:e)));
    qrs(m)= s+shift;
    i = qrs(m);
    m=m+1;

    if figures
        sig_part_orig = signal(qrs(m-1)-4:qrs(m-1)-4+ms_wnd);
        figure(1);
        clf()
        subplot(4,1,1)
        plot(sig_part_orig,'k')
        hold on
        plot(sig_part_f8hz,'g')
```

```
        plot(hhecg,'r')
        grid on
        axis tight

        subplot(4,1,2)
        plot(dpi)
        axis tight
        ylim([dpi(end),max(dpi(30:end))])
        subplot(4,1,3)
        plot(der)
        hold on
        plot(der(1:ms_285),'r','LineWidth',1.5)
        plot(idx_neg,der(idx_neg),'bo')
        plot(idx_pos,der(idx_pos),'ro')
        axis tight
        ylim([min(der(30:end)),max(der(30:end))])

        subplot(4,1,4)
        plot(dpi)
        hold on
        plot(dpi(1:ms_285),'r','LineWidth',1.5)
        axis tight
        ylim([dpi(end),max(dpi(30:end))])

        plot(idx_neg_sort(1),dpi(idx_neg_sort(1)),'bo')
        plot(idx_pos_sort(1),dpi(idx_pos_sort(1)),'ro')
        pause();
    end

end
% Delete first index, because it was artificial:
qrs(1)=[];
end
```

## 4.4 Funkcja walidująca wyznaczone indeksy

```
function [acc,sen] = validation(r_ind_db,r,wnd)
%VALIDATION performs simple validation prcess using two sets of
%annotations. Computes accuracy and sensitivity with the selected tolerance
%window
%
% Inputs
% ------
%       r_ind_db - indices of true samples (annotation form database)
%       r - computed indices of qrs complexes
%       wnd - tolerance window in samples
%
% Outputs
% ------
%       acc - accuracy
%       sen - sensitivity
%
% Dependencies
% ------------
%
%   This function calls:
%
%   This function is called by:
%       - analysis
%
%
% Authors:    P. Wegrzynowicz, A. Zadlo
% Copyright:  .
% Date:       2016-12-02

    if size(r,1)<size(r,2)
        r = r';
    end
    if size(r_ind_db,1)<size(r_ind_db,2)
        r_ind_db = r_ind_db';
    end

    tp = 0;
    fp = 0;
    fn = 0;
    a = ones(numel(r),1)*(-wnd:1:wnd);
    r_wide = a + repmat(r,[1,2*wnd+1]);
    a = ones(numel(r_ind_db),1)*(-wnd:1:wnd);
    r_ind_db_wide = a + repmat(r_ind_db,[1,2*wnd+1]);
    tp = numel(intersect(r_ind_db,r_wide(:)));
    fn = numel(r_ind_db) - tp;
    fp = numel(r) - numel(intersect(r,r_ind_db_wide));
    acc = tp/(tp+fp+fn)*100;
    sen = tp/(tp+fn)*100;
    display(['Accuracy: ',num2str(acc),'%   sensitivity: ',...
        num2str(sen),'%; window=',num2str(wnd),' samples'])
end
```