PERM - sprawozdanie z laboratorium nr 3
“Splot. Wprowadzenie do filtrów.”
Michał Kolankiewicz, 303765

1. Cel zadania:

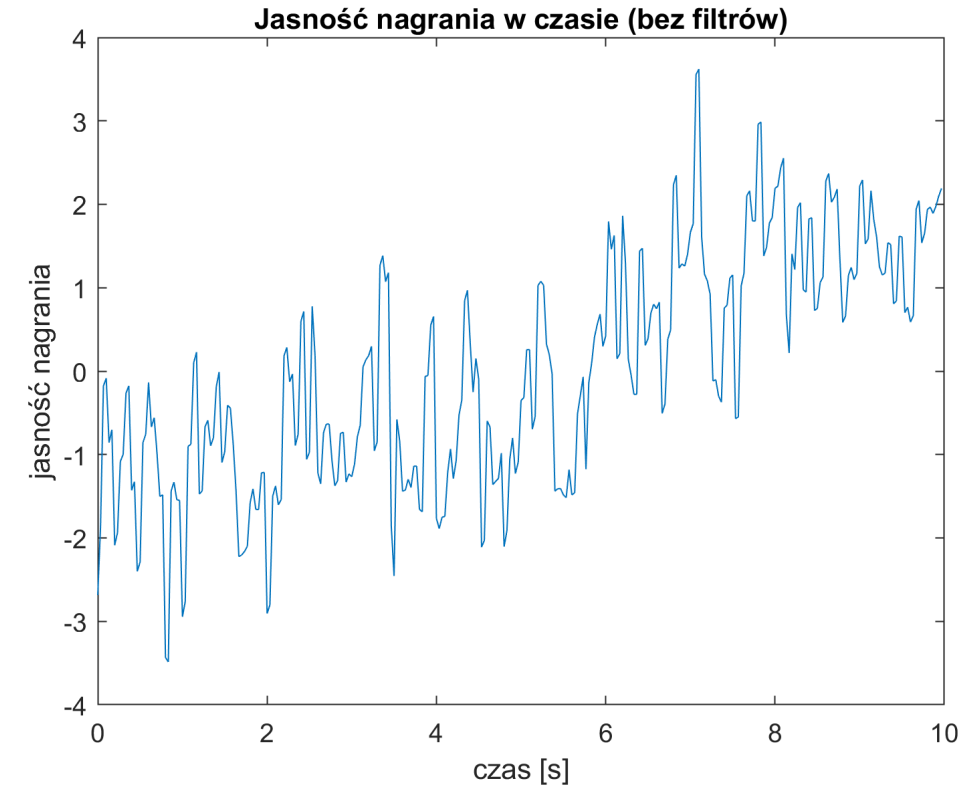
Zadanie zrealizowane w ramach laboratorium nr 1 miało na celu odczyt pulsu człowieka na podstawie dostarczonego nagrania palca. Choć program działał, był podatny na zakłócenia, co wymagało np. wielokrotnego powtarzania nagrania do momentu, aż będzie odpowiedniej “jakości”. Celem trzeciego laboratorium była poprawa kodu z laboratorium pierwszego tak, aby przy zastosowaniu różnych filtrów i operacji splotu uodpornić program na nieidealne dane.

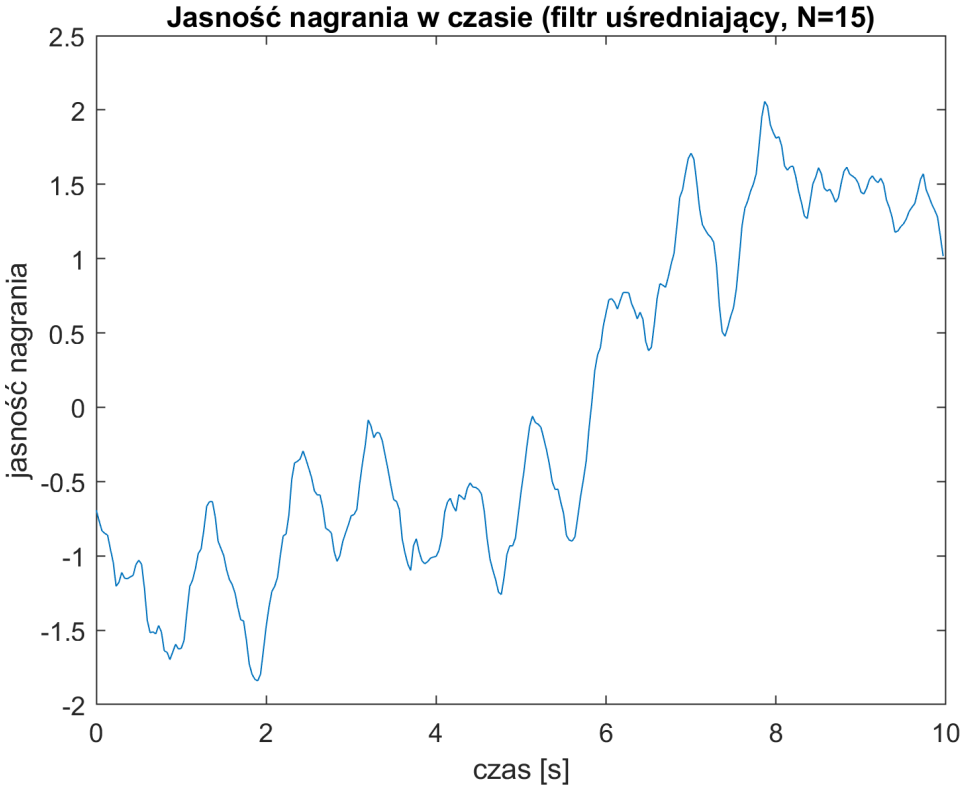
2. Opis rozwiązania:

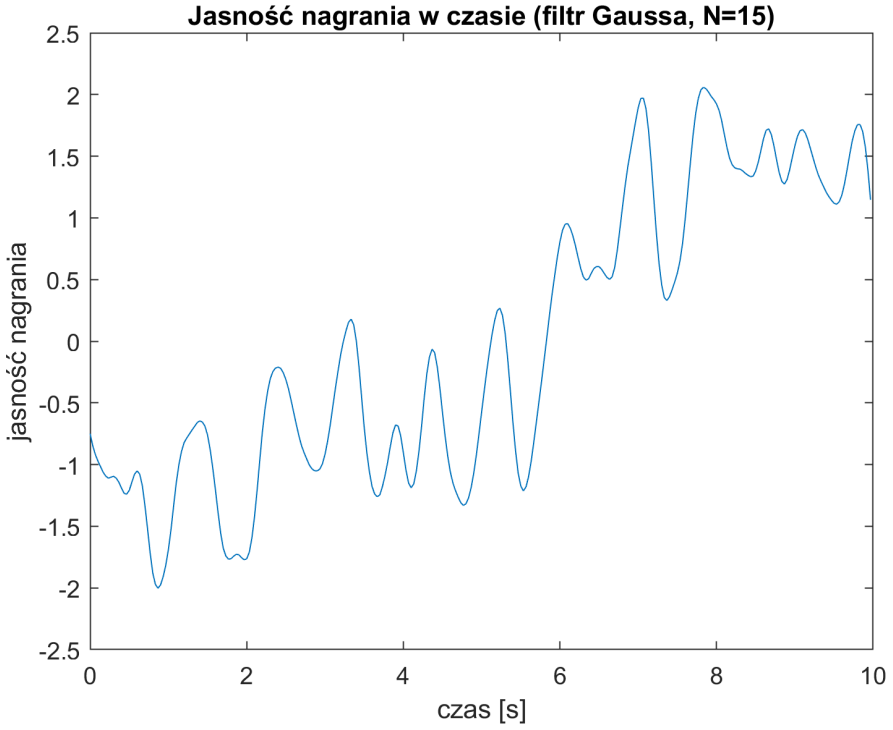
W kodzie zaimplementowano cztery filtry:
- filtr uśredniający,
- filtr Gaussa,
- filtr wykrywający zbocza,
- autokorelację.

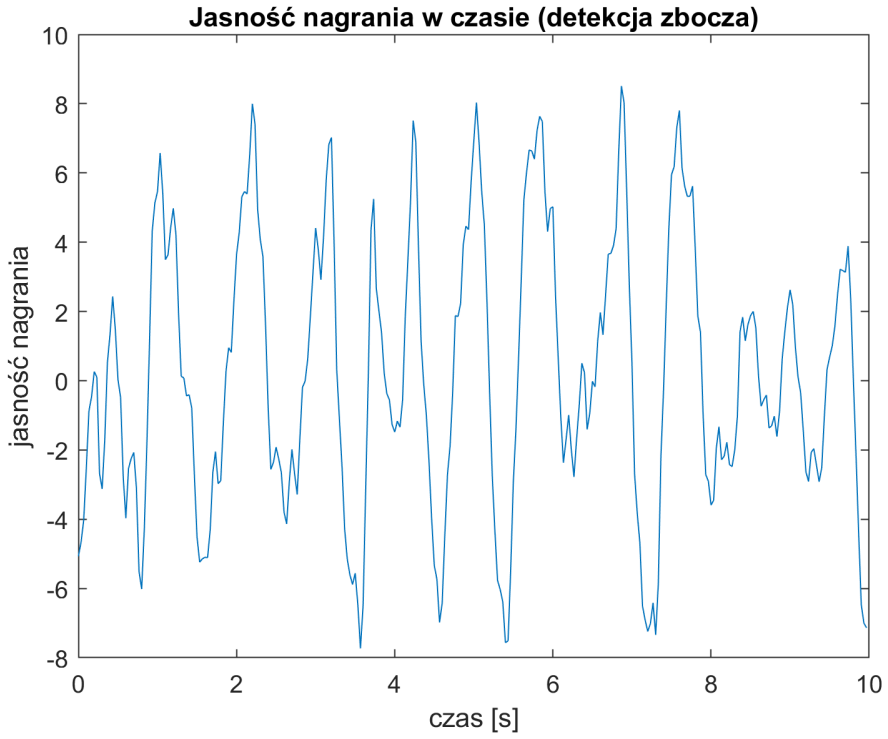
Pierwsze trzy filtry wykorzystują operację splotu sygnału wejściowego z wektorami różnej postaci. Filtr uśredniający jest postaci [1/N, 1/N, 1/N… 1/N] i umożliwia uśrednienie sygnału na przestrzeni N kolejnych próbek. Gaussowski ma wartości zależne od rozkładu Gaussa na N elementów wektora filtra takie, aby suma ich wynosiła 1. Jego działanie jest podobne do filtra uśredniającego, ale z priorytetem na wartości centralne przedziału uśredniania . Filtr wykrywający zbocza to przestrzeń liniowa od 1 do -1 na N elementów, umożliwia wykrywanie szybkich zmian sygnału (jak się okaże, w przypadku tego zadania jeden z najpraktyczniejszych filtrów). Autokorelacja polega na wykorzystaniu sygnału wejściowego jako filtra na samym sobie przesuniętym o kilka próbek, więc jest to skala samopodobieństwa w czasie. Umożliwia wyznaczenie częstotliwości dominującej w sygnale.

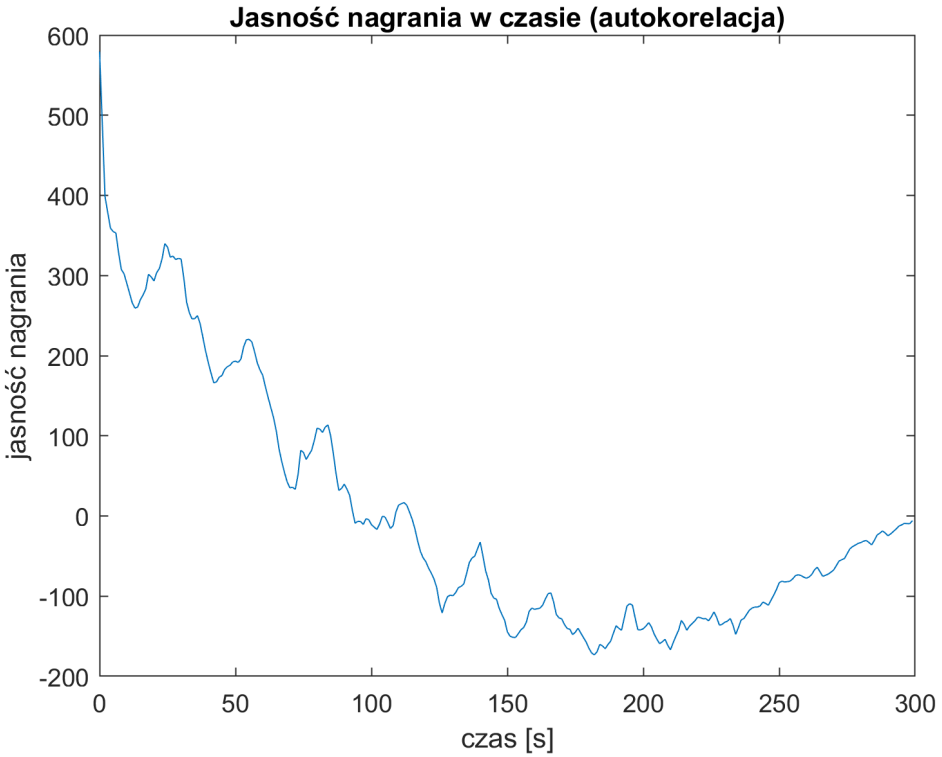
Ten sam sygnał wejściowy, którego uzyskanie opisano w laboratorium pierwszym, poddano działaniu wszystkich filtrów i porównano wyniki. Dla filtrów, gdzie można było dobrać długość wektora filtra zastosowano długość N = 15 dla stosunkowo silnego wygładzania. W poprzednim sprawozdaniu wspomniałem, że eksperyment powiódł się za drugim nagraniem, pierwsze wyłapywało niższą częstotliwość. Poniższe wykresy przedstawiają rezultat zastosowania filtrów na pierwotnym nagraniu.

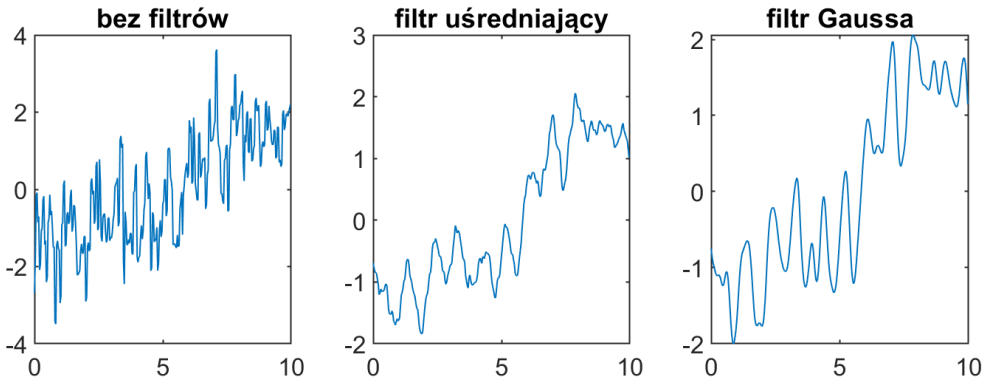

Jasność nagrania w czasie (bez filtrów)
jasność nagrania
czas [s]
4
3
2
1
0
-1
-2
-3
-4
0
2
4
6
8
10

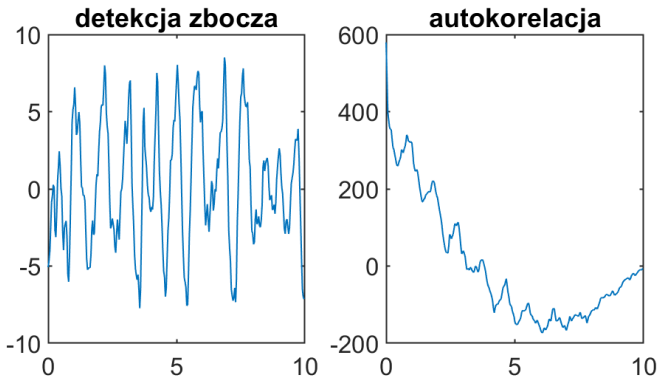

Jasność nagrania w czasie (filtr uśredniający, N=15)
jasność nagrania
czas [s]
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
2
4
6
8
10


Jasność nagrania w czasie (filtr Gaussa, N=15)
jasność nagrania
czas [s]
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
2
4
6
8
10

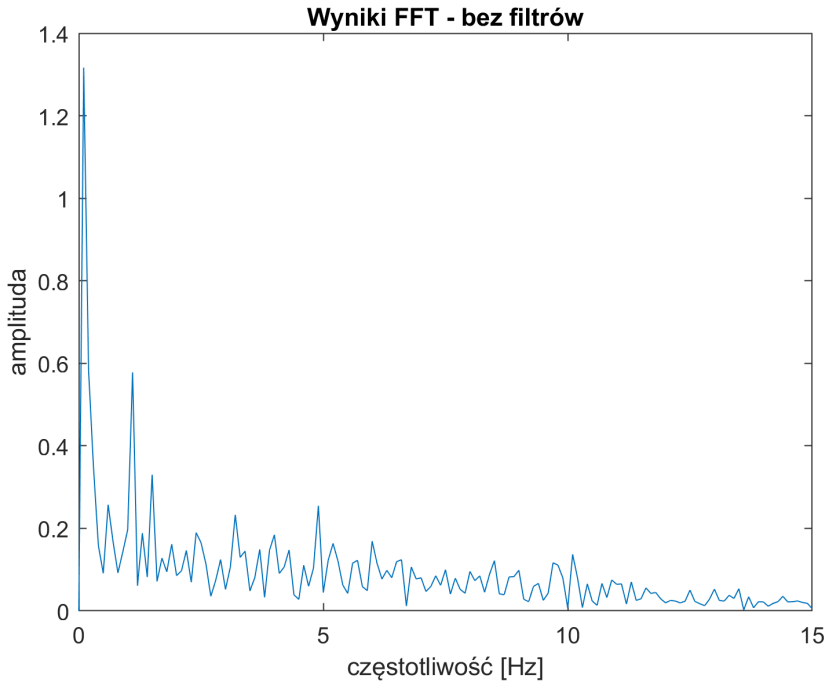

Jasność nagrania w czasie (detekcja zbocza)
jasność nagrania
czas [s]
10
8
6
4
2
0
-2
-4
-6
-8
0
2
4
6
8
10

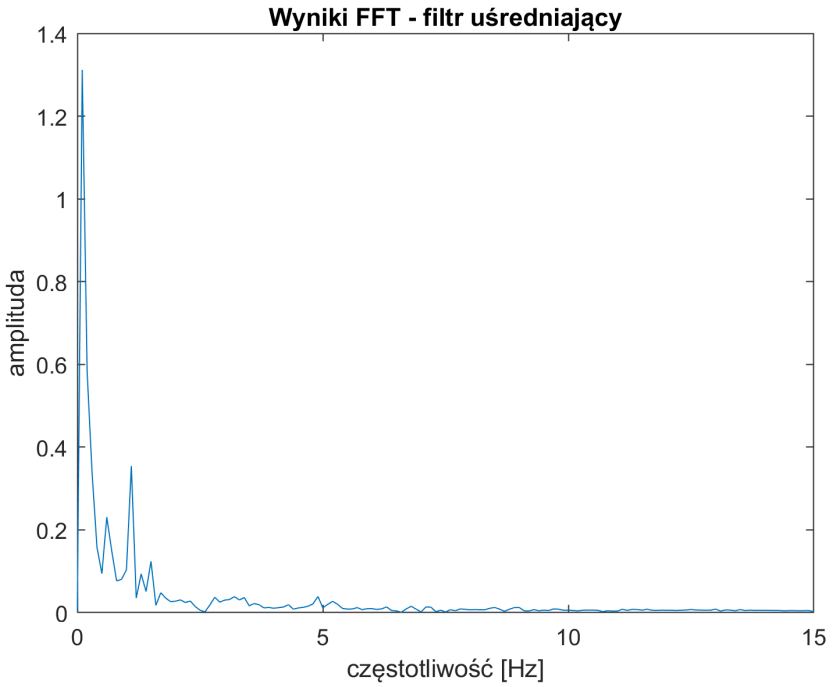

Jasność nagrania w czasie (autokorelacja)
jasność nagrania
czas [s]
600
500
400
300
200
100
0
-100
-200
0
50
100
150
200
250
300

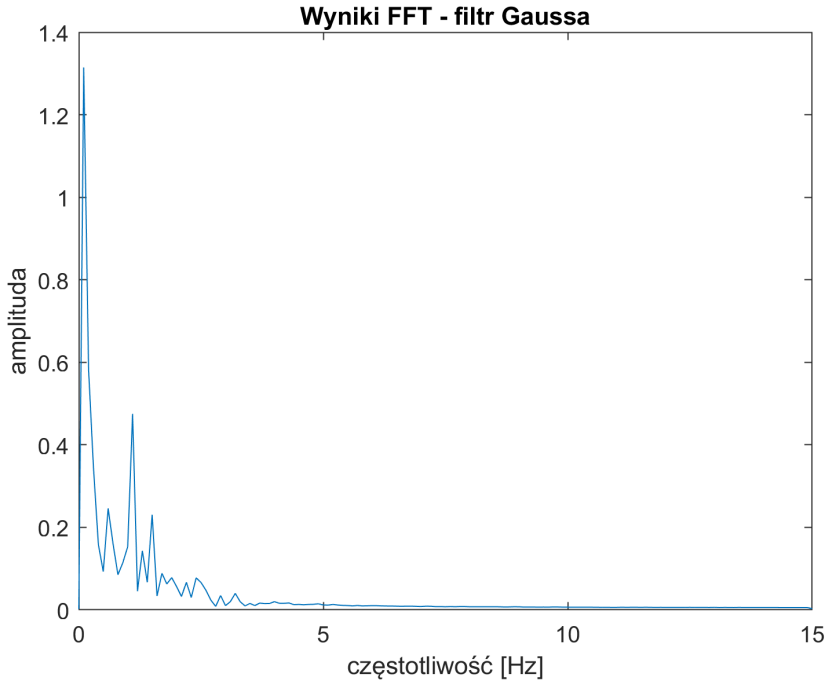

bez filtrów
filtr uśredniający
filtr Gaussa
4
2
0
-2
-4
3
2
1
0
-1
-2
2
1
0
-1
-2
0
5
10

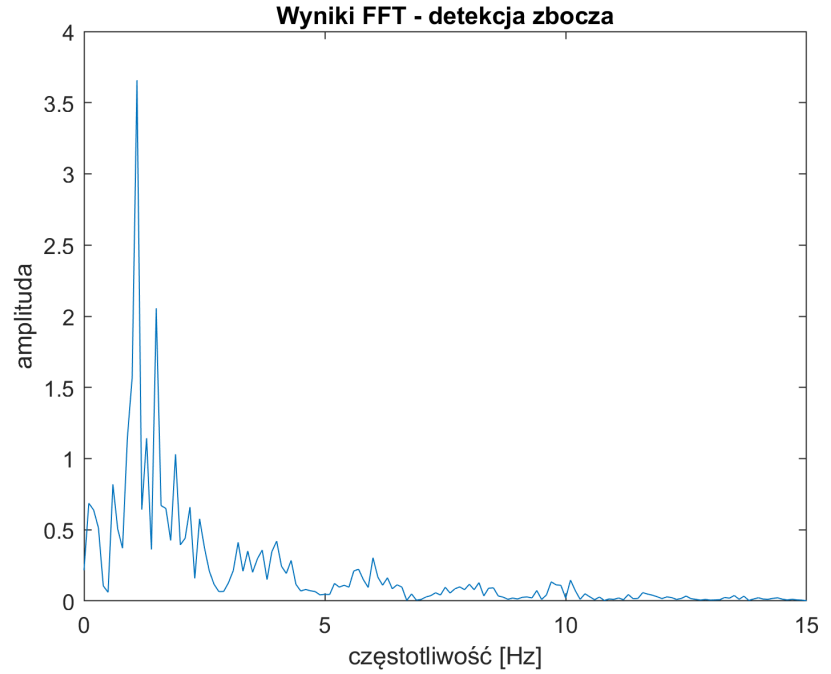

detekcja zbocza
autokorelacja
10
5
0
-5
-10
600
400
200
0
-200
0
5
10

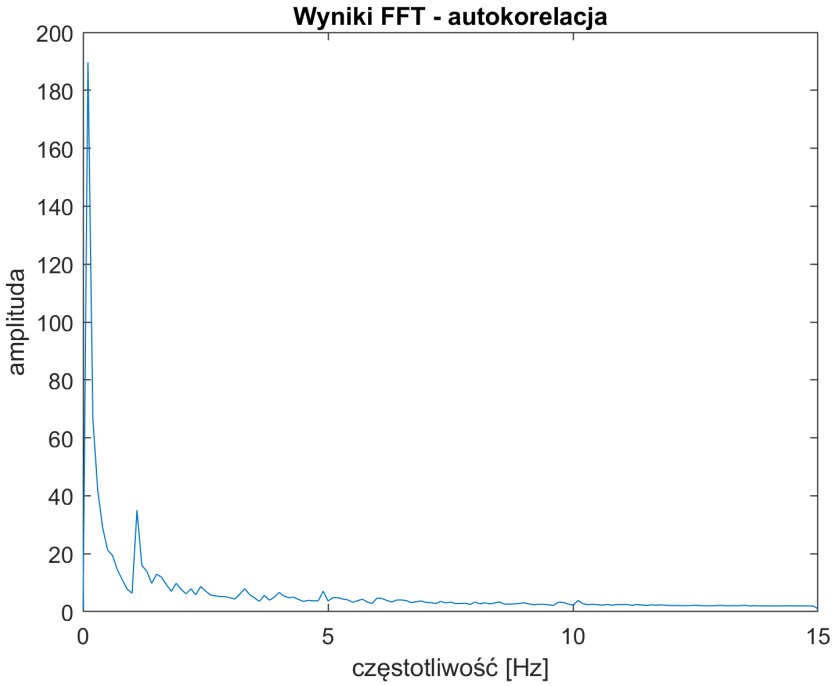
Na każdym z tych sygnałów użyto szybkiej transformaty Fouriera i również porównano wyniki, które prezentują się następująco.

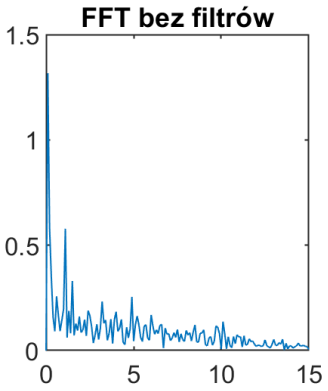


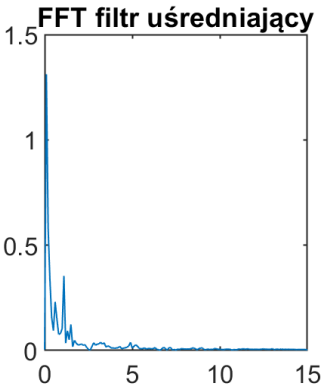


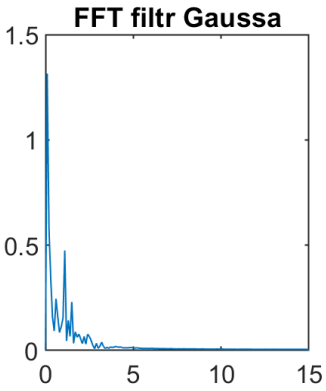

Wyniki FFT - filtr Gaussa
amplituda
częstotliwość [Hz]
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0
5
10
15

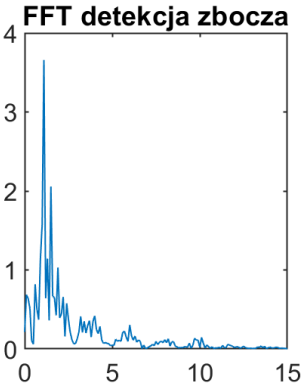

Wyniki FFT - detekcja zbocza
amplituda
częstotliwość [Hz]
0
0.5
1
1.5
2
2.5
3
3.5
4
0
5
10
15

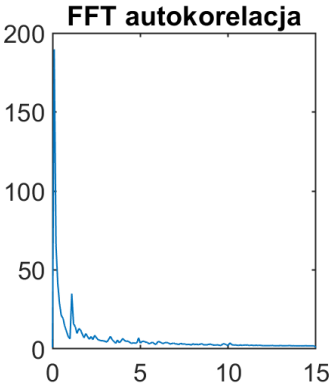

Wyniki FFT - autokorelacja
amplituda
częstotliwość [Hz]
0
20
40
60
80
100
120
140
160
180
200
0
5
10
15


FFT bez filtrów
0
0.5
1
1.5
0
5
10
15


FFT filtr uśredniający
0
0.5
1
1.5
0
5
10
15


FFT filtr Gaussa
0
0.5
1
1.5
0
5
10
15


FFT detekcja zbocza
0
1
2
3
4
0
5
10
15


FFT autokorelacja
0
50
100
150
200
0
5
10
15

3. Wyniki eksperymentu:

Z powyższych doświadczeń można zauważyć, że dla tego konkretnego zadania jedynie filtr wykrywający zbocze zdał egzamin i dla niego transformata FFT jako jedyna zwróciła oczekiwany wynik (BPM = 66, dla pozostałych filtrów wynik ten wynosił 6).

W ramach dalszych eksperymentów postanowiłem sprawdzić inne nagrania i zobaczyć, jak z nimi poradzą sobie różne filtry. Od swojej dziewczyny dostałem dodatkowe 4 nagrania, z których 3 były zbyt zaszumione, by wersja bez filtra mogła sobie z nimi poradzić. Poniższa tabela prezentuje otrzymane wyniki dla różnych nagrań mojego i jej palca.

nagranie\filtr	brak	mean	gauss	cliff	xcorr
puls.mp4	6	6	6	66	6
puls2.mp4	72	72	72	72	72
pulsKamila.mp4	72	6	6	72	72
pulsKamila2.mp4	6	6	6	72	6
pulsKamila3.mp4	12	12	12	84	12
pulsKamila4.mp4	6	6	6	78	6

Jak można zauważyć, najlepiej z zadaniem wykrywania pulsu radzi sobie filtr wykrywający zbocze, który wskazał oczekiwaną wartość pulsu w każdym przypadku. Nagranie “puls2.mp4” było na tyle dobrej jakości, że działalność filtrów nie zakłóciła wyniku, jednak dla “pulsKamila.mp4” zastosowanie filtrów dolnoprzepustowych (mean, gauss) wręcz pogorszyło odczyt częstotliwości.

Należy również zwrócić uwagę na rozdzielczość pomiaru. Ponieważ program pracuje na 10-sekundowej próbce nagrania mającego 30FPS możliwy jest odczyt częstotliwości jedynie z dokładnością do 0,1Hz (30FPS -> przedział częstotliwości 0-30Hz; 10s*30FPS = 300 próbek; 30Hz/300 = 0,1Hz), a 0,1Hz przekłada się na 6 BPM. Większą dokładność odczytu można by osiągnąć, gdyby dostarczać dłuższe nagranie.

4. Kod skryptu “L3_zad.m”:

```
%% WCZYTANIE FILMU

N = 300; % liczba ramek do wczytania (pierwsze 10 sekund nagrania)
br = zeros(1, N); % wektor jasności

FPS = 30;   % częstotliwość próbkowania [Hz]
T = 1/FPS;  % okres próbkowania [s]
```

```
t = (0:N-1)*T; % oś czasu

v = VideoReader(['pulsKamila.mp4']); % załadowanie i odczytanie pliku video

%% UZYSKANIE WYKRESU JASNOŚCI

% wczytanie pierwszych N obrazów i analiza jasności
for i=1:N
   I = rgb2gray(read(v,i)); % wczytanie klatki w skali szarości
   br(i) = mean(I, 'all'); % średnia jasność całego obrazu w danej klatce
end

br = br - mean(br); % przesunięcie o stałą

figure % wykres czasowy bez filtracji
plot(t, br)
xlabel("czas [s]")
ylabel("jasność nagrania")
title("Jasność nagrania w czasie (bez filtrów)")
print("jasność_bez.png","-dpng","-r280")

%% FILTROWANIE WYNIKU

N_filter = 5;

% Filtr uśredniający:
f_mean = ones(1,N_filter)/N_filter;
c_mean = conv(br, f_mean, "same");

figure % wykres czasowy
plot(t, c_mean)
xlabel("czas [s]")
ylabel("jasność nagrania")
title(strcat("Jasność nagrania w czasie (filtr uśredniający, N=", string(N_filter), ")"))
print("jasność_z_usrednieniem.png","-dpng","-r280")

% Filtr Gaussa:
f_gauss = fspecial('gaussian', [1, N_filter], 3);
c_gauss = conv(br, f_gauss, "same");

figure % wykres czasowy
plot(t, c_gauss)
xlabel("czas [s]")
ylabel("jasność nagrania")
title(strcat("Jasność nagrania w czasie (filtr Gaussa, N=", string(N_filter), ")"))
print("jasność_z_gaussem.png","-dpng","-r280")

% Detekcja zbocza:
f_cliff = linspace(1, -1, N_filter);
c_cliff = conv(br, f_cliff, "same");

figure % wykres czasowy
plot(t, c_cliff)
xlabel("czas [s]")
ylabel("jasność nagrania")
title("Jasność nagrania w czasie (detekcja zbocza)")
print("jasność_zbocza.png","-dpng","-r280")

% Autokorelacja
[r, lags] = xcorr(br);
r = r(lags >= 0);
lags = lags(lags>=0);

figure % wykres czasowy
plot(lags, r)
xlabel("czas [s]")
```

```
ylabel("jasność nagrania")
title("Jasność nagrania w czasie (autokorelacja)")
print("jasność_autokorelacja.png","-dpng","-r280")


%% TRANSFORMATA FOURIERA

Y = fft(br);    % szybka transformata Fouriera
A = abs(Y);     % amplituda sygnału
A = A/N;        % normalizacja amplitudy
A = A(1:N/2+1); % wycięcie istotnej części spektrum
A(2:end-1) = 2*A(2:end-1);

Y_mean = fft(c_mean);    % szybka transformata Fouriera
A_mean = abs(Y_mean);     % amplituda sygnału
A_mean = A_mean/N;        % normalizacja amplitudy
A_mean = A_mean(1:N/2+1); % wycięcie istotnej części spektrum
A_mean(2:end-1) = 2*A_mean(2:end-1);

Y_gauss = fft(c_gauss);    % szybka transformata Fouriera
A_gauss = abs(Y_gauss);     % amplituda sygnału
A_gauss = A_gauss/N;        % normalizacja amplitudy
A_gauss = A_gauss(1:N/2+1); % wycięcie istotnej części spektrum
A_gauss(2:end-1) = 2*A_gauss(2:end-1);

Y_cliff = fft(c_cliff);    % szybka transformata Fouriera
A_cliff = abs(Y_cliff);     % amplituda sygnału
A_cliff = A_cliff/N;        % normalizacja amplitudy
A_cliff = A_cliff(1:N/2+1); % wycięcie istotnej części spektrum
A_cliff(2:end-1) = 2*A_cliff(2:end-1);

Y_r = fft(r);    % szybka transformata Fouriera
A_r = abs(Y_r);     % amplituda sygnału
A_r = A_r/N;        % normalizacja amplitudy
A_r = A_r(1:N/2+1); % wycięcie istotnej części spektrum
A_r(2:end-1) = 2*A_r(2:end-1);

f_step = FPS/N;     % zmiana częstotliwości
f = 0:f_step:FPS/2; % oś częstotliwości

figure % wykres amplitudowy
plot(f, A)
xlabel("częstotliwość [Hz]")
ylabel("amplituda")
title("Wyniki FFT - bez filtrów")
print("FFT_bez.png","-dpng","-r280")

figure % wykres amplitudowy
plot(f, A_mean)
xlabel("częstotliwość [Hz]")
ylabel("amplituda")
title("Wyniki FFT - filtr uśredniający")
print("FFT_mean.png","-dpng","-r280")

figure % wykres amplitudowy
plot(f, A_gauss)
xlabel("częstotliwość [Hz]")
ylabel("amplituda")
title("Wyniki FFT - filtr Gaussa")
print("FFT_gauss.png","-dpng","-r280")

figure % wykres amplitudowy
plot(f, A_cliff)
xlabel("częstotliwość [Hz]")
ylabel("amplituda")
title("Wyniki FFT - detekcja zbocza")
```

```
print("FFT_zbocze.png","-dpng","-r280")

figure % wykres amplitudowy
plot(f, A_r)
xlabel("częstotliwość [Hz]")
ylabel("amplituda")
title("Wyniki FFT - autokorelacja")
print("FFT_autokorelacja.png","-dpng","-r280")

%% ODCZYTANIE CZĘSTOTLIWOŚCI

[max_shade_change, heart_rate_sample] = maxk(A, 1); % znalezienie indeksu dla tętna serca
heart_BPM = 60*f(heart_rate_sample) % przeliczenie na BPM

[max_shade_change, heart_rate_sample] = maxk(A_mean, 1); % znalezienie indeksu dla tętna serca
heart_BPM_mean = 60*f(heart_rate_sample) % przeliczenie na BPM

[max_shade_change, heart_rate_sample] = maxk(A_gauss, 1); % znalezienie indeksu dla tętna serca
heart_BPM_gauss = 60*f(heart_rate_sample) % przeliczenie na BPM

[max_shade_change, heart_rate_sample] = maxk(A_cliff, 1); % znalezienie indeksu dla tętna serca
heart_BPM_cliff = 60*f(heart_rate_sample) % przeliczenie na BPM

[max_shade_change, heart_rate_sample] = maxk(A_r, 1); % znalezienie indeksu dla tętna serca
heart_BPM_autor = 60*f(heart_rate_sample) % przeliczenie na BPM

%% WYKRESY ŁĄCZONE

figure
subplot(2,3,1)
plot(t, br)
title("bez filtrów")
subplot(2,3,2)
plot(t, c_mean)
title("filtr uśredniający")
subplot(2,3,3)
plot(t, c_gauss)
title("filtr Gaussa")
subplot(2,3,5)
plot(t, c_cliff)
title("detekcja zbocza")
subplot(2,3,6)
plot(t, r)
title("autokorelacja")
print("czasówki_razem.png","-dpng","-r280")

figure
subplot(2,3,1)
plot(f, A)
title("FFT bez filtrów")
subplot(2,3,2)
plot(f, A_mean)
title("FFT filtr uśredniający")
subplot(2,3,3)
plot(f, A_gauss)
title("FFT filtr Gaussa")
subplot(2,3,5)
plot(f, A_cliff)
title("FFT detekcja zbocza")
subplot(2,3,6)
plot(f, A_r)
title("FFT autokorelacja")
print("FFT_razem.png","-dpng","-r280")
```