Sprawozdanie z laboratorium 3. z przedmiotu TRA prowadzonego w semestrze 24Z

Piotr Sienkiewicz 324 887

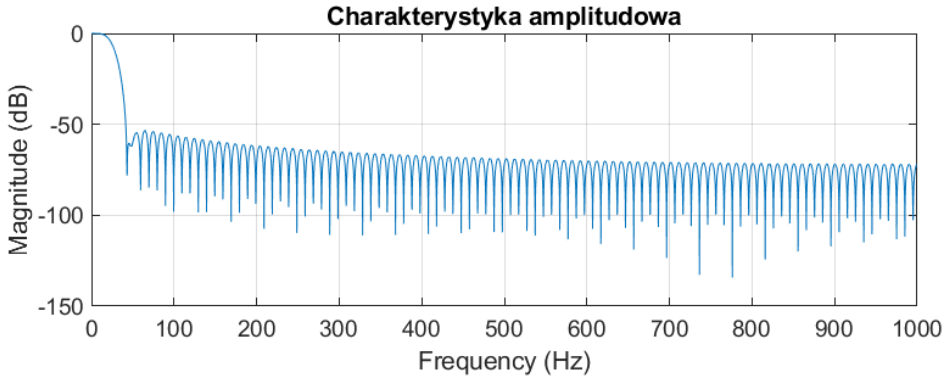
1 Zadanie 1

W celu doboru współczynników filtra o zadanych parametrach metodą prób i błędów napisano poniższy skrypt w MATLAB:

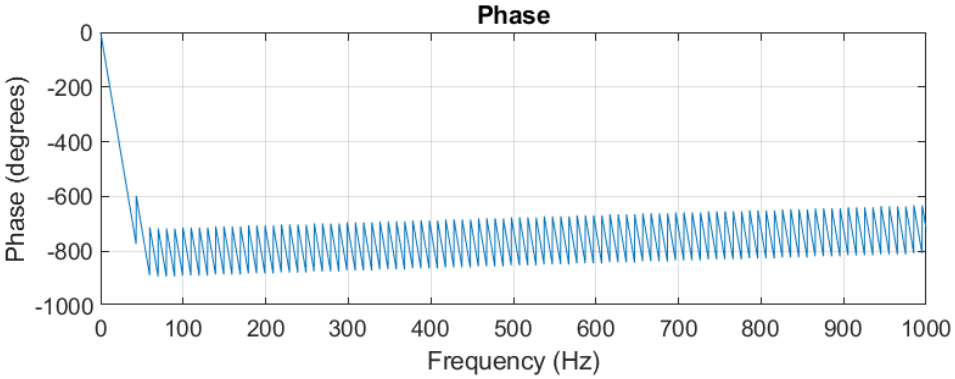
Listing 1: Skrypt MATLAB do zadania 1.

```
clear all

Fs = 2000;
Wp = 0.025;
Ws = 0.050;

order = 200;

b = fir1(order, Wp, 'low');

figure;
freqz(b, 1, 4096, Fs);
title('Charakterystyka amplitudowa i fazowa filtra');

figure;
grpdelay(b, 1, 2048, Fs);
title('Opoznienie grupowe filtra');

figure;
zplane(b, 1);
title('Zera i bieguny filtra na okregu jednostkowym');
```

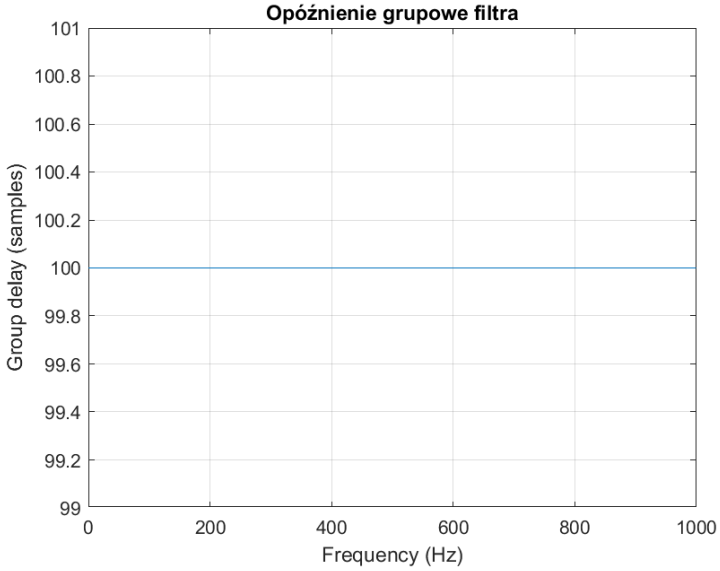
Wybrano taką częstotliwość próbkowania `Fs`, aby uzyskać ładne okrągłe wartości na osi OX charakterystyk filtrów. Następnie eksperymentalnie dobrano rząd filtru na 200. W ten sposób uzyskano filtr o następujących charakterystykach i parametrach:



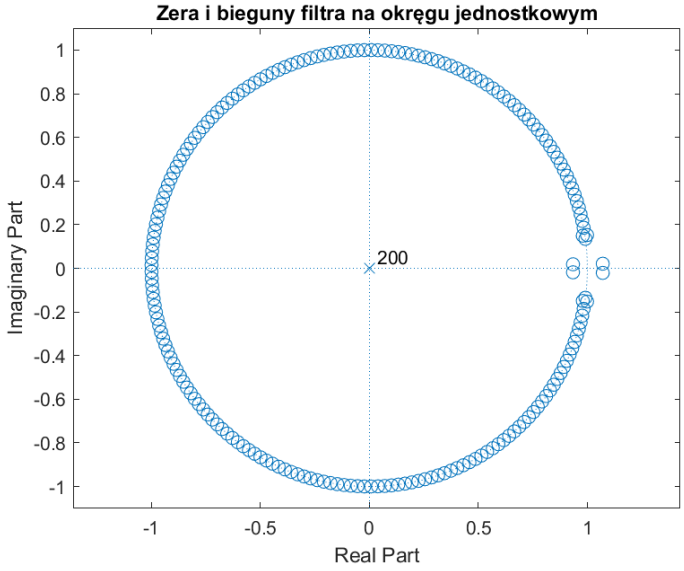
Rysunek 1: Charakterystyka amplitudowa filtra



Rysunek 2: Charakterystyka fazowa filtra



Rysunek 3: Opóźnienie grupowe filtra



Rysunek 4: Rozmieszczenie zer i biegunów filtra na płaszczyźnie zespolonej

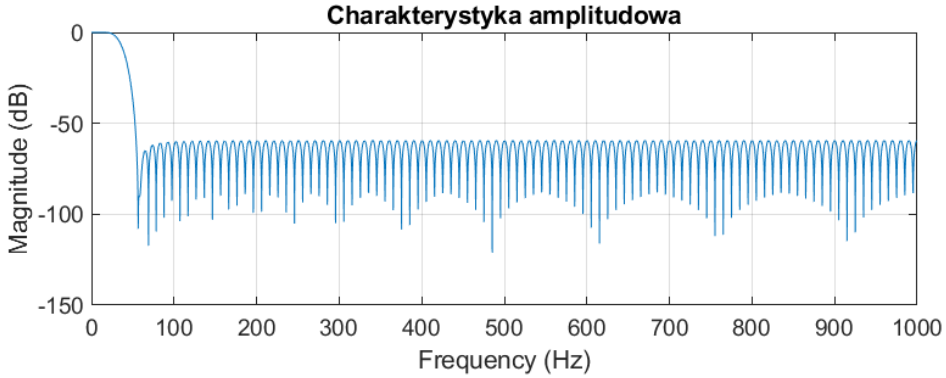
Jak widać charakterystyka filtru wygląda dobrze, jest zapewnione tłumienie -60dB prawie w całym pasmie zaporowym filtru, nie licząc początku pasma zaporowego. Po powiększeniu wykresu w pasmie przepustowym zauważono, że dla częstotliwości 25Hz (co odpowiada `Wp = 0.025`) tłumienie filtru wynosiło około -5dB, a więc zdecydowano się na kolejne zmiany parametrów filtru. W tym celu zmieniono linijkę:

```
b = fir1(order, Wp, 'low');
```

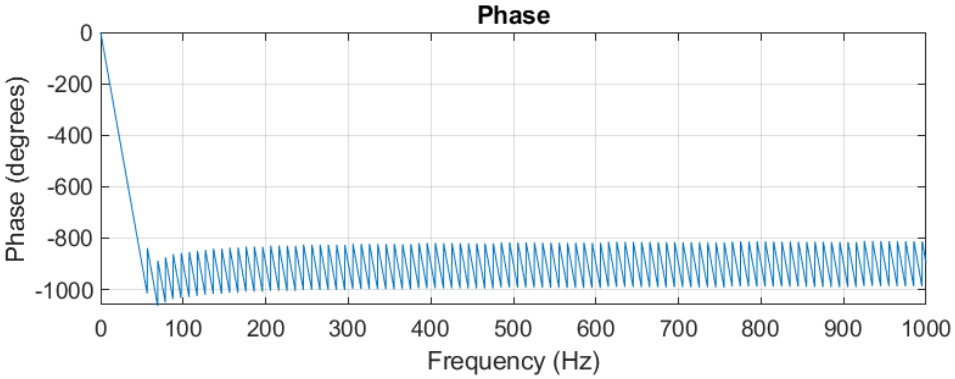
na:

```
b = fir1(order, Wp + 0.01, 'low', chebwin(order + 1, 50));
```

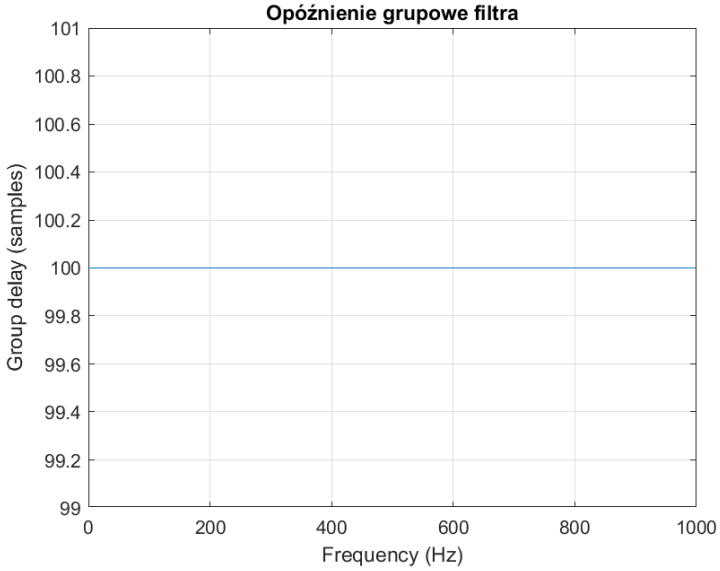
Sprawiło to, że narzucono, aby filtr został zaprojektowany z oknem Czebyszewa typu II, co sprawiło, że tłumienie filtru było stałe w całym paśmie zaporowym oraz otrzymano ostrzejsze załamanie charakterystyki na granicy pasma przepustowego i zaporowego. W ten sposób przy tym samym rzędzie filtru otrzymano lepszy rezultat:



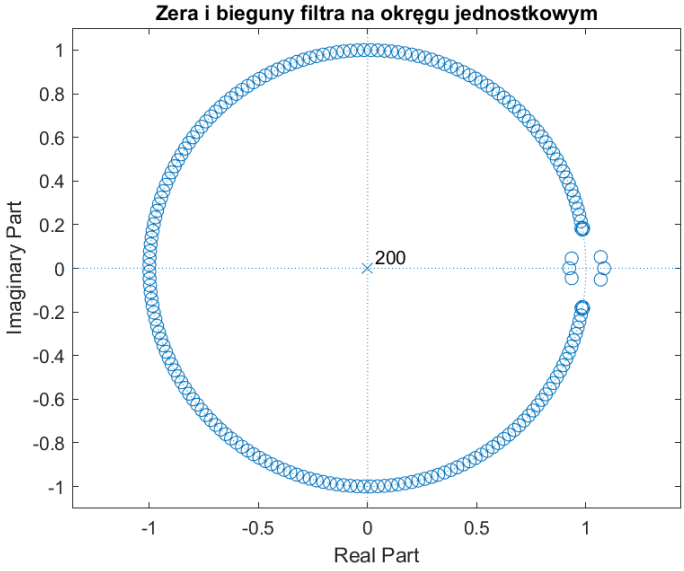
Rysunek 5: Charakterystyka amplitudowa filtra



Rysunek 6: Charakterystyka fazowa filtra



Rysunek 7: Opóźnienie grupowe filtra



Rysunek 8: Rozmieszczenie zer i biegunów filtra na płaszczyźnie zespolonej

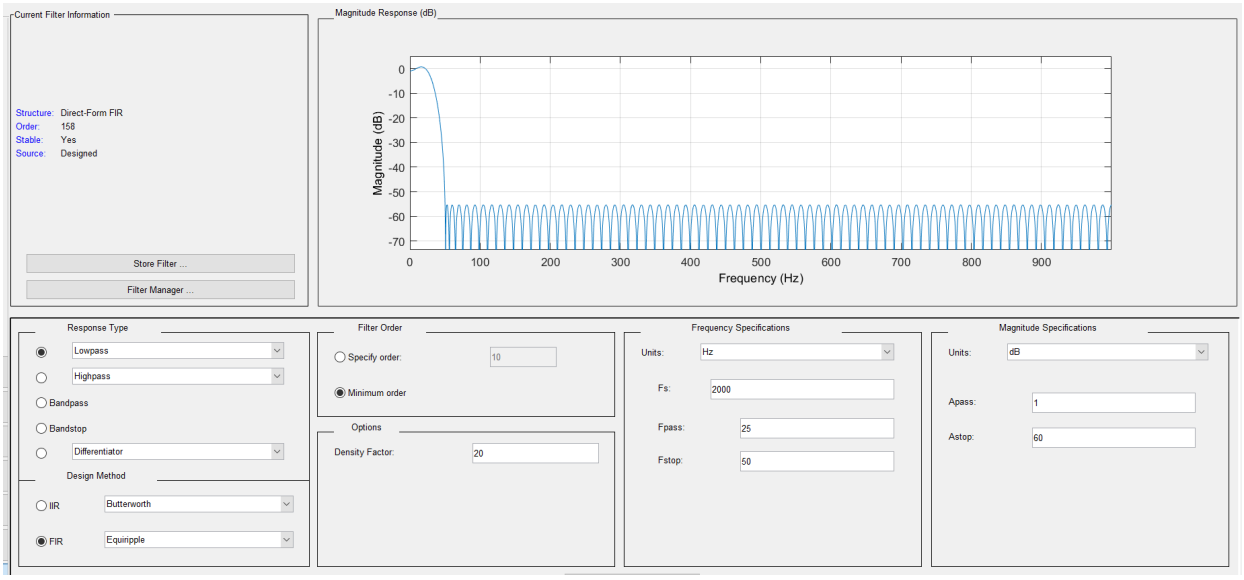
Dzięki takim zmianom otrzymano filtr, którego tłumienie jest stałe w całym paśmie zaporowym. Dodatkowo zwiększono nieznacznie częstotliwość graniczną pasma przepustowego. Dzięki temu tłumienie filtra w prawie całym pasmie przepustowym jest stałe, dodatkowo wynosi poniżej -1dB dla częstotliwości `Wp = 0.025`, po czym prawie od razu bardzo gwałtownie spada aż do poziomu około -60dB.

Oba filtry mają stałe opóźnienie grupowe, co jest bardzo pożądaną cechą filtrów pozwalającą uniknąć zniekształceń fazowych. Wynosi ona równo połowę rzędu filtra, czyli 100, co jest typową zależnością dla filtrów tego typu.

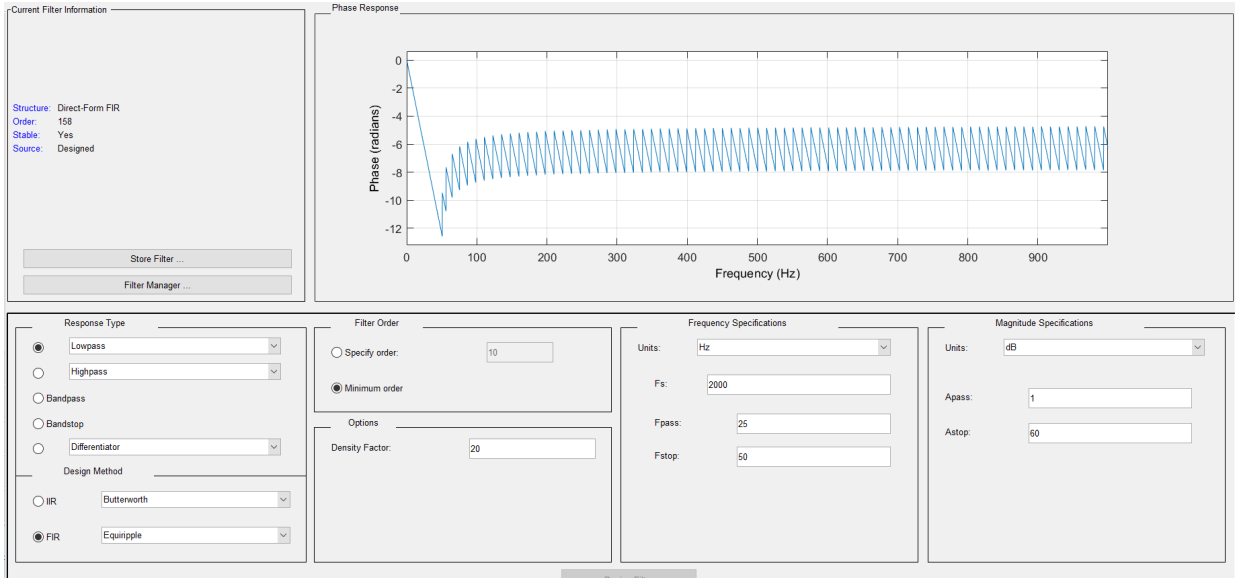
Oba filtry są stabilne, co wynika z rozłożenia zer i biegunów na płaszczyźnie zespolonej. W obu filtrach zera leżą na okręgu jednostkowym (co jest dozwolone, jedynie bieguny muszą leżeć wewnątrz okręgu jednostkowego) z wyjątkiem kilku zer. Dodatkowo w punkcie (0, 0) znajduje się 200 biegunów, co jest równe rzędowi filtrów. Na początku może to wydawać się dziwne, ponieważ filtr FIR w równaniu transmitancji nie ma mianownika, a więc nie powinien mieć biegunów. Jednakże wielomian opisujący transmitancję dyskretną składa się ze składników zawierających czynniki z^{-n}, czyli $\frac{1}{z^n}$, a więc jednak okazuje się, że taki filtr też ma bieguny, ale wszystkie są w punkcie (0, 0), a ich liczba równa jest rzędowi filtru.

1.1 Zadanie 2

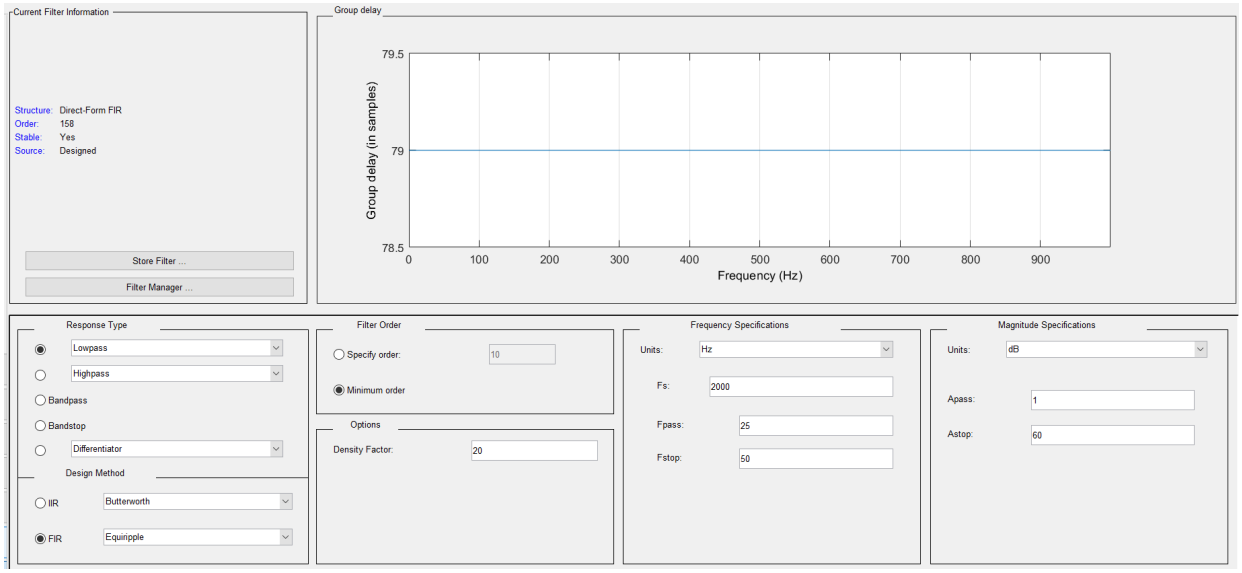
W tym zadaniu zaprojektowano filtr o takich samych wymaganiach, lecz tym razem z użyciem środowiska `filterDesigner`:



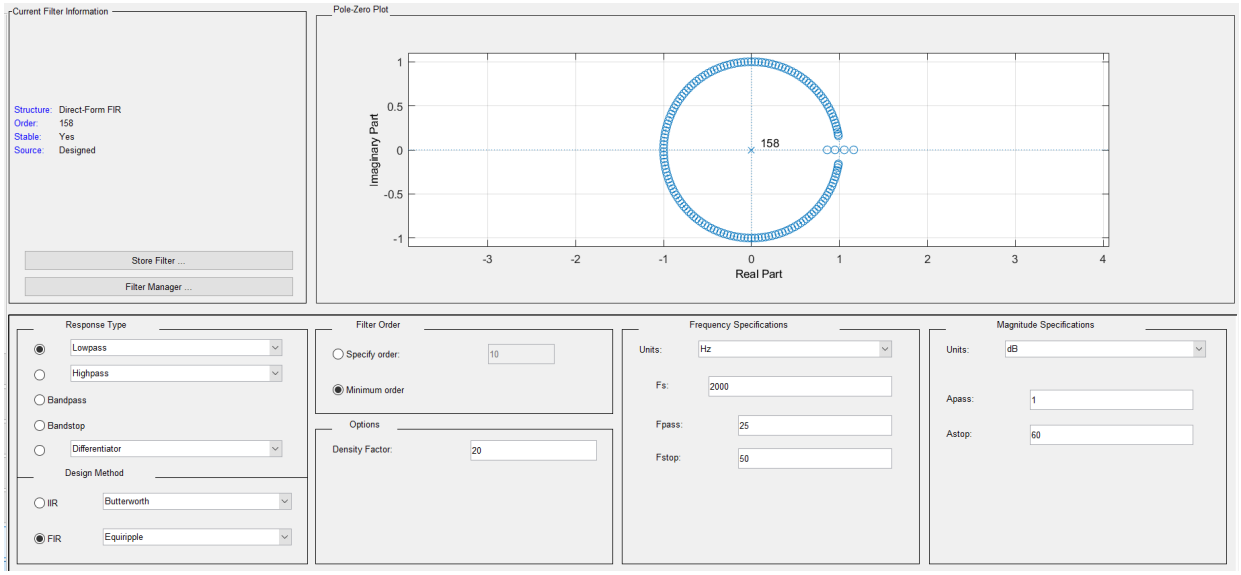
Rysunek 9: Charakterystyka amplitudowa filtra



Rysunek 10: Charakterystyka fazowa filtra



Rysunek 11: Opóźnienie grupowe filtra



Rysunek 12: Rozmieszczenie zer i biegunów filtra na płaszczyźnie zespolonej

Program wyznaczył filtr o najmniejszym rzędzie wynoszącym 158, który spełniał założenia. Wybrana opcja `Equiripple` oznacza, że filtr ma mieć jednakowe zafalowania na charakterystyce częstotliwościowej w całym paśmie zaporowym. Rząd filtru w tym przypadku jest trochę mniejszy, lecz nadal spory. Jednak po powiększeniu charakterystyk można było zauważyć, że filtr zaprojektowany w zadaniu 1. ma trochę lepsze właściwości. W paśmie przepustowym miał mniejsze zafalowania - ten filtr miał nawet delikatne podbicie wzmocnienia na granicy pasma. Dodatkowo w filtrze z zadania 1. zbocze charakterystyki jest delikatnie bardziej strome.

Wszystkie te filtry mają bardzo duże rzędy, co wynika z narzuconych wymagań, gdyż to, aby filtry miały

bardzo strome zbocze, duże tłumienie i najlepiej małe zafalowania charakterystyki, bardzo mocno wpływa na minimalny rząd filtru.

2 Zadanie 3

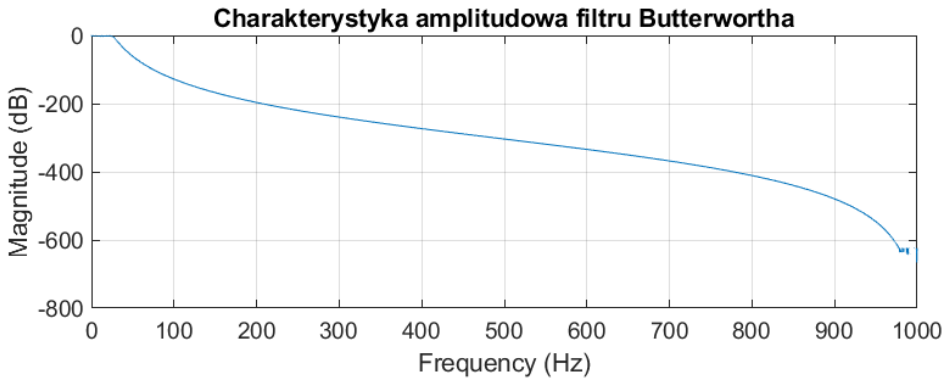
2.1 Filtr Butterwortha

W zadaniu 3. napisano następujący skrypt do wyznaczenia współczynników filtra IIR Butterwortha o najmniejszym możliwym rzędzie:

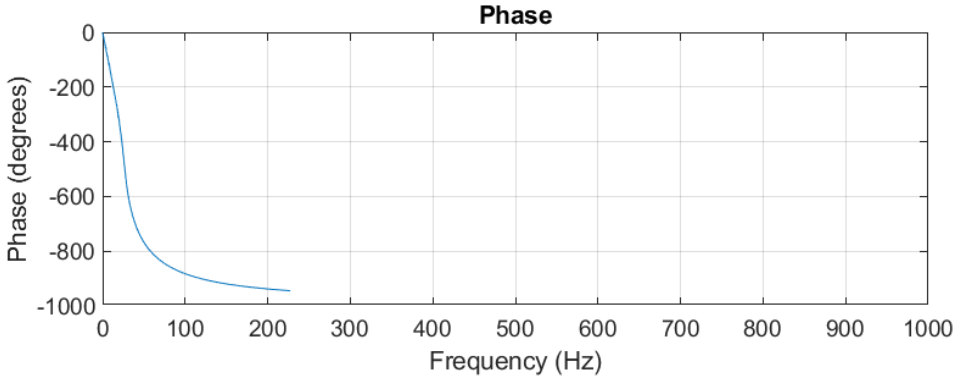
Listing 2: Skrypt MATLAB do zadania 3.

```
clear all

Fs = 2000;
Wp = 0.025;
Ws = 0.050;
ripple_passband = 1;
stopband_attenuation = 60;

[n_min, Wn] = buttord(Wp, Ws, ripple_passband, stopband_attenuation);

[b, a] = butter(n_min, Wn, 'low');

figure;
freqz(b, a, 4096, Fs);
title('Charakterystyka amplitudowa filtru Butterwortha');

figure;
grpdelay(b, a, 2048, Fs);
title('Opoznienie grupowe filtru Butterwortha');

figure;
zplane(b, a);
title('Zera i bieguny filtru Butterwortha');

fprintf("Rzad filtru: %.d \n", n_min);
```

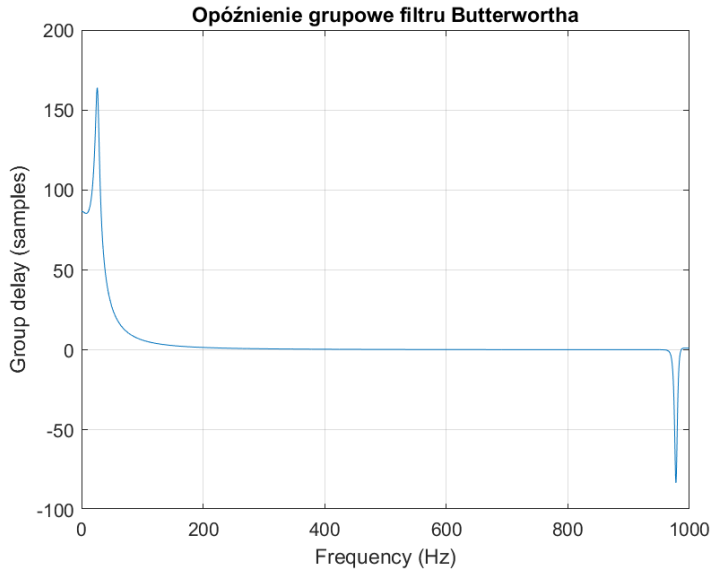
W wyniku działania programu otrzymano filtr o następujących charakterystykach:



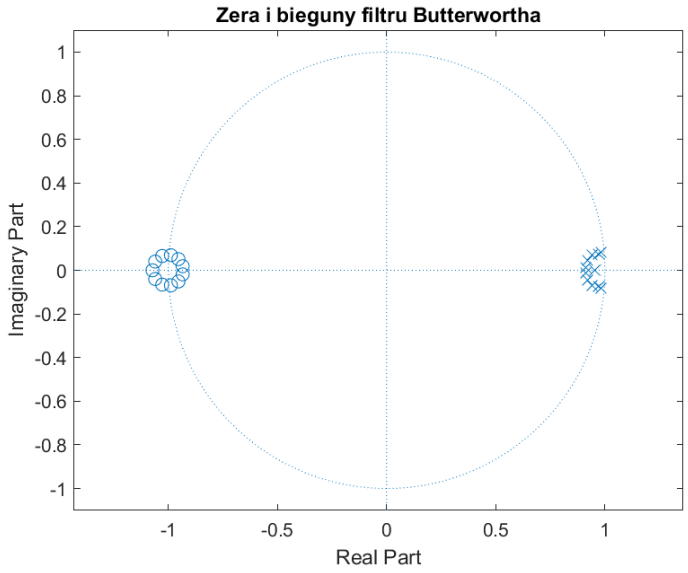
Rysunek 13: Charakterystyka amplitudowa filtra



Rysunek 14: Charakterystyka fazowa filtra



Rysunek 15: Opóźnienie grupowe filtra



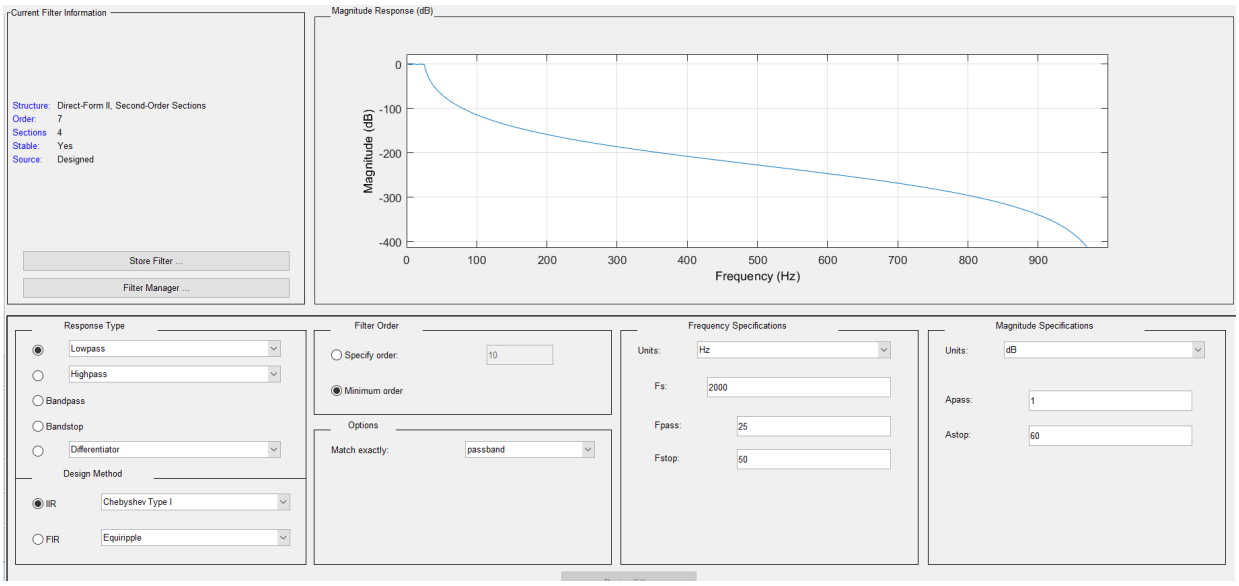
Rysunek 16: Rozmieszczenie zer i biegunów filtra na płaszczyźnie zespolonej

Otrzymano filtr o rzędzie zaledwie 11. Od razu widać, że różni się on od poprzednich filtrów i można powiedzieć, że ma gorsze parametry. Nachylenie charakterystyki amplitudowej na skraju pasm przepustowego i zaporowego jest mniejsze, a charakterystyka nie wypłaszcza się w paśmie zaporowym, lecz spada dalej aż do olbrzymiej wartości -600dB, co można uznać albo za wadę, albo za zaletę, w zależności od zastosowania. Problem zaczyna być widoczny w przypadku opóźnienia grupowego filtra. W paśmie przepustowym rośnie ono bardzo szybko, następnie w paśmie zaporowym wynosi zero, a na koniec występuje jeszcze dodatkowy ujemny pik opóźnienia. Jest to bardzo niepożądane zjawisko, gdyż składowe sygnału o różnych częstotliwościach będą się propagowały przez filtr z różnym czasem, co prowadzi do zniekształceń fazowych lub może powodować inne problemy.

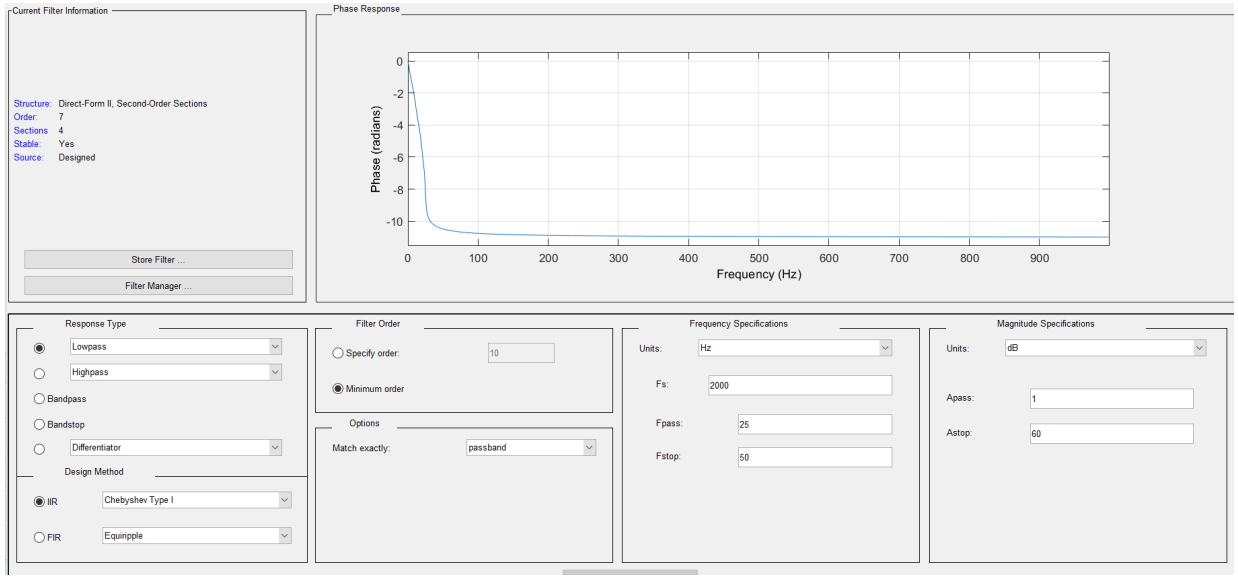
Widać też inne rozłożenie zer i biegunów niż ostatnio. Filtr nadal jest stabilny, ale bieguny blisko okręgu jednostkowego oznaczają, że odpowiedź impulsowa takiego filtru będzie bardzo długo ustalała się w okolicach punktu 0, nigdy go nie osiągając - wszakże jest to filtr IIR.

2.2 Filtr Czebyszewa

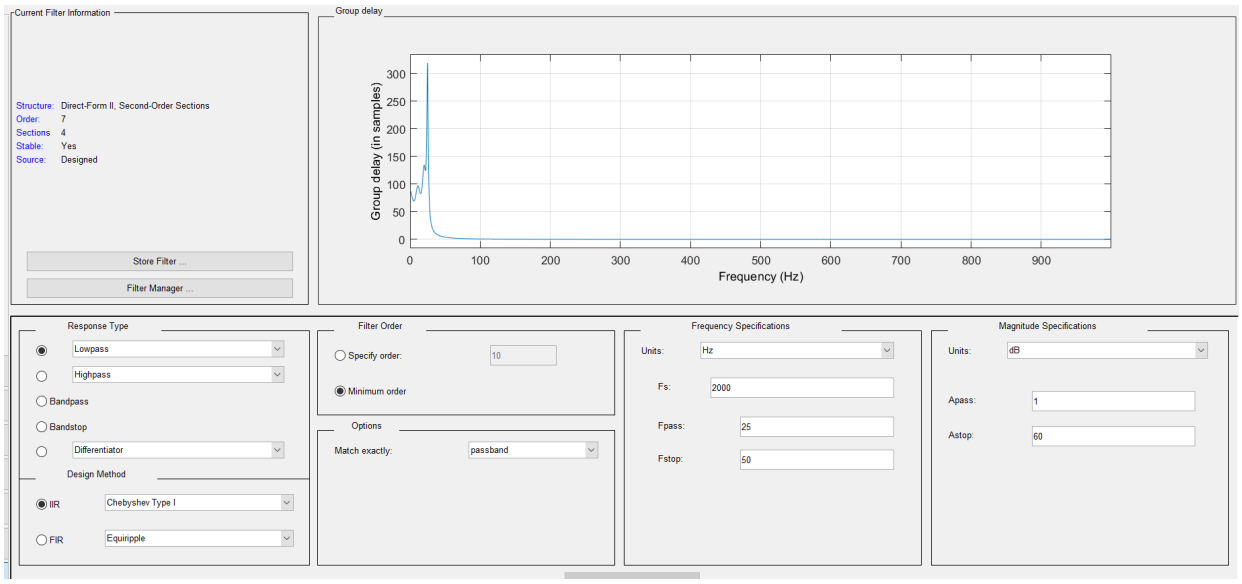
Tym razem posłużono się programem `filterDesigner`:



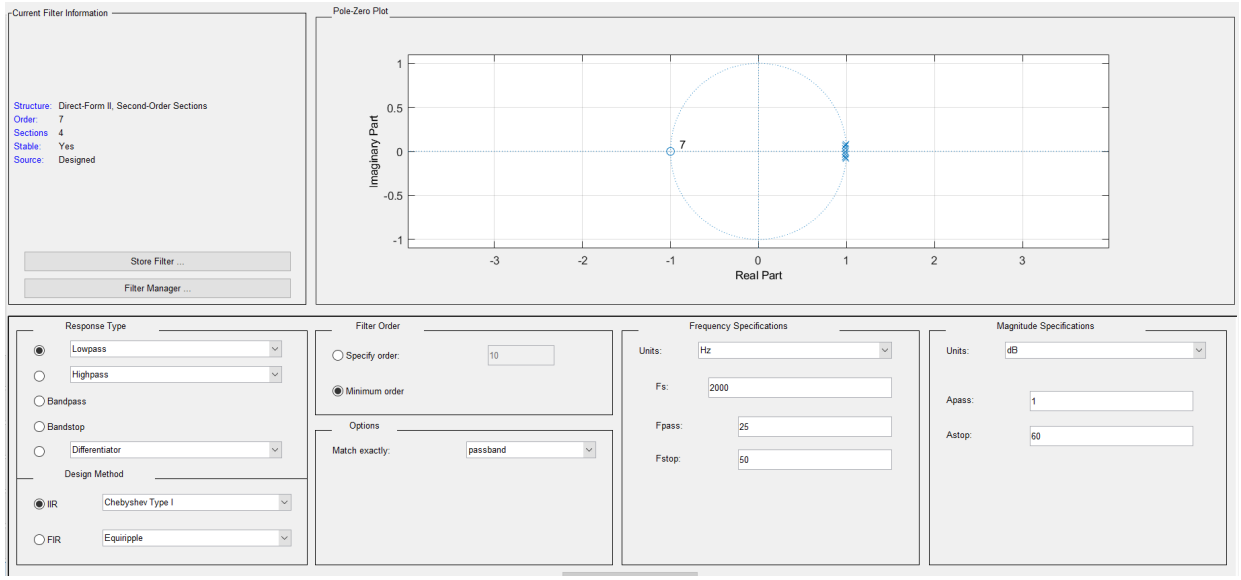
Rysunek 17: Charakterystyka amplitudowa filtra



Rysunek 18: Charakterystyka fazowa filtra



Rysunek 19: Opóźnienie grupowe filtra

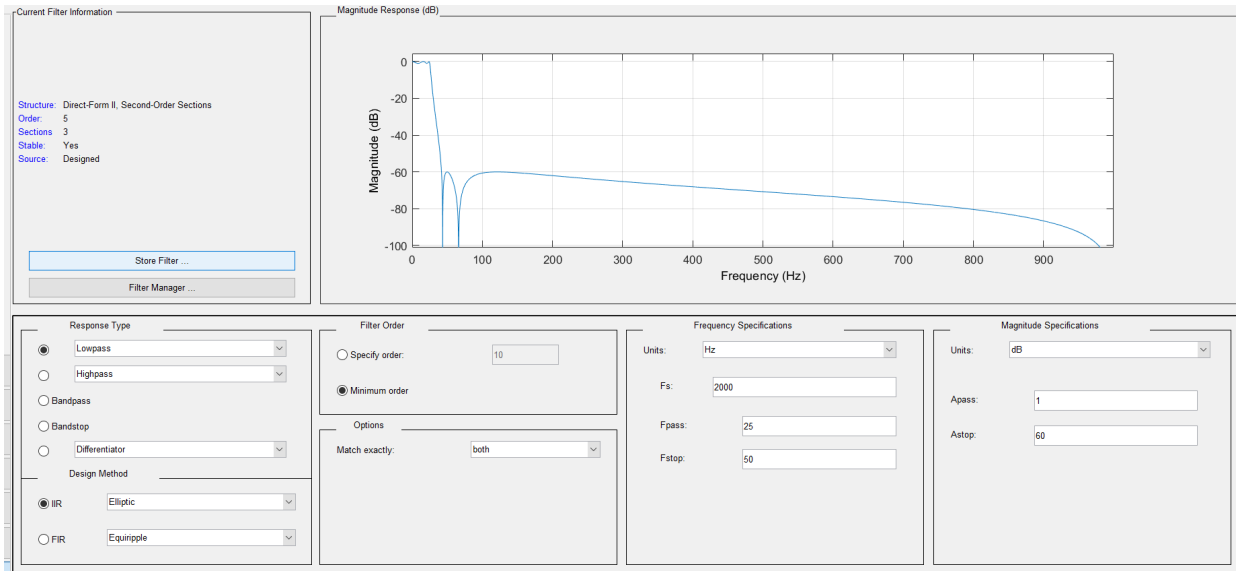


Rysunek 20: Rozmieszczenie zer i biegunów filtra na płaszczyźnie zespolonej

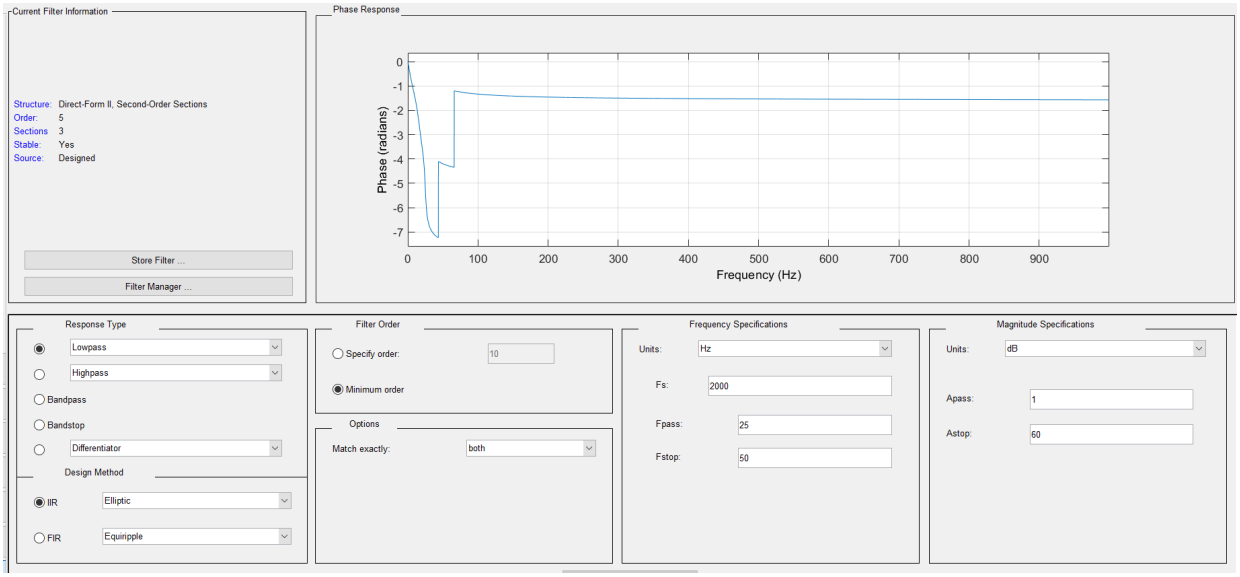
Filtr ten wygląda podobnie jak poprzedni zaprojektowany z użyciem funkcji `butter` z tą różnicą, że widoczne jest większe załamanie charakterystyki dla częstotliwości granicznej. Dalej tłumienie filtru ciągle rośnie jak w poprzednim przypadku. Charakterystyka fazowa jest podobna - na początku jest duży skok fazy, a następnie w całym paśmie zaporowym faza nie zmienia się zbyt mocno. Opóźnienie grupowe i rozmieszczenie zer i biegunów także są podobne.

2.3 Filtr eliptyczny

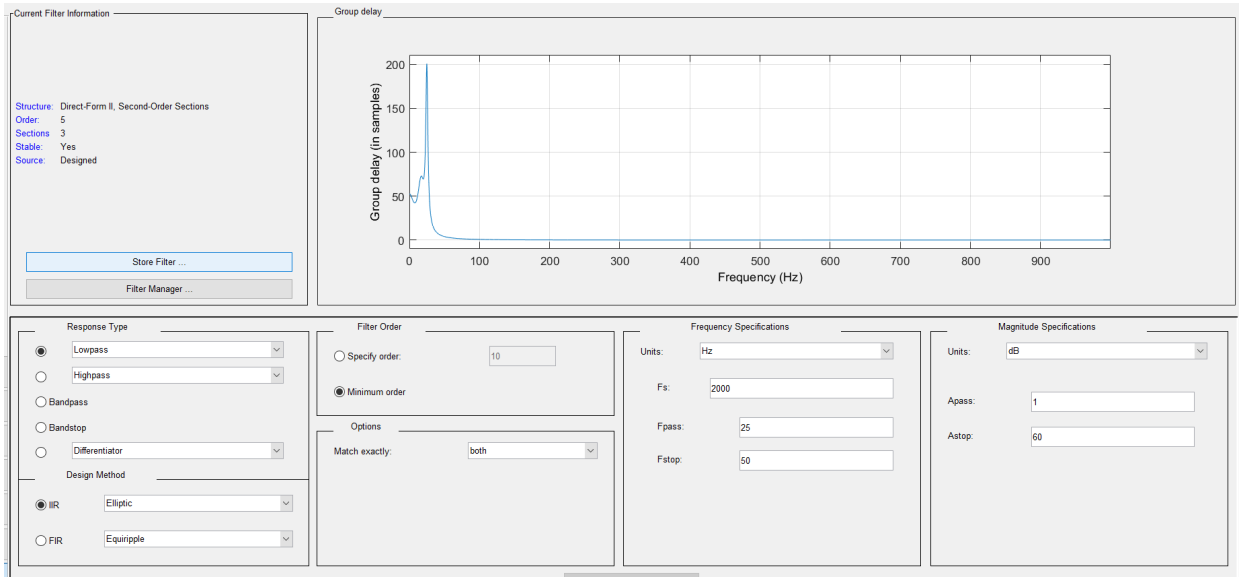
Także i tym razem posłużono się programem `filterDesigner`:



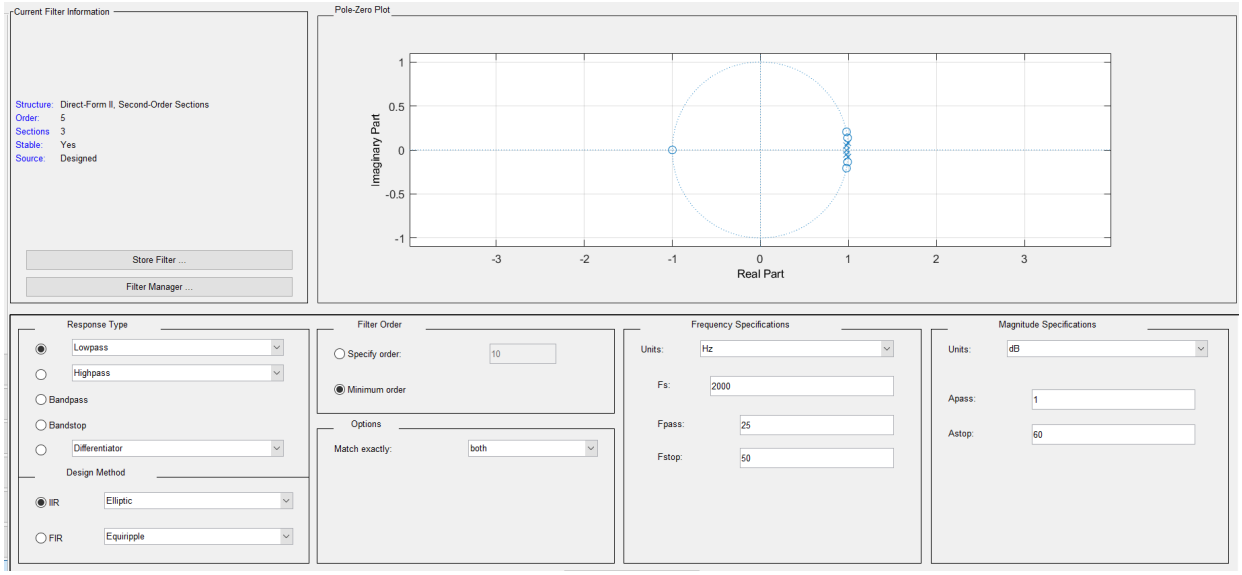
Rysunek 21: Charakterystyka amplitudowa filtra



Rysunek 22: Charakterystyka fazowa filtra



Rysunek 23: Opóźnienie grupowe filtra



Rysunek 24: Rozmieszczenie zer i biegunów filtra na płaszczyźnie zespolonej

Największą różnicą w tutaj jest charakterystyka amplitudowa ze stromym zboczem i dwoma zafalowaniami przy częstotliwości granicznej, która się wypłaszcza w paśmie zaporowym. Charakterystyka fazowa ma duży skok na początku, a następnie filtr ma nieduże i stałe przesunięcie fazowe w całym paśmie zaporowym.

2.4 Porównanie filtrów

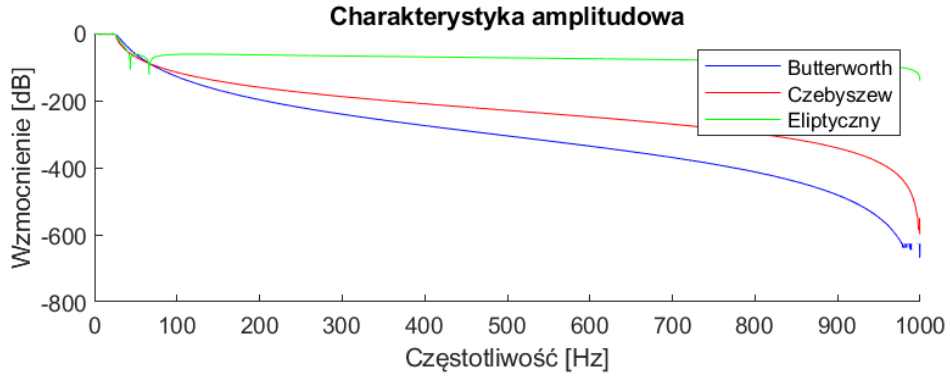
Napisano skrypt pozwalający na wyświetlenie charakterystyk filtrów zaimportowanych z `filterDesignera` do workspace MATLABa na wspólnych wykresach. Lepszym rozwiązaniem mogłoby być zaimportowanie

jednego filtru do `filterDesignera`, lecz tutaj program nie posiada dedykowanej funkcji i trzeba by było przeklejać wektory współczynników ręcznie.

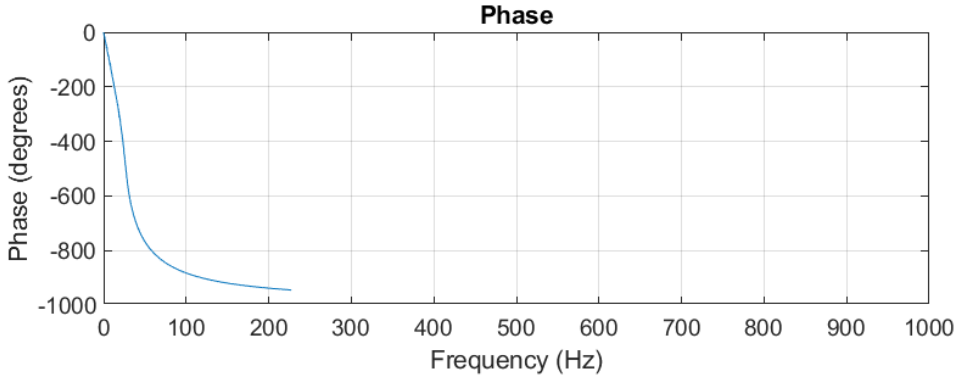
Listing 3: Skrypt MATLAB do zadania 3b.

```
Fs = 2000;
Wp = 0.025;
Ws = 0.050;
ripple_passband = 1;
stopband_attenuation = 60;

% Filtr Butterwortha
[n_min, Wn] = buttord(Wp, Ws, ripple_passband, stopband_attenuation);
[num_butter, den_butter] = butter(n_min, Wn, 'low');

% Ustawienia kolorow dla trzech filtrow
colors = {'b', 'r', 'g'};

% Charakterystyka amplitudowa i fazowa
figure;
subplot(2, 1, 1);  % Wykres amplitudowy
hold on;
[h_butter, f] = freqz(num_butter, den_butter, 4096, Fs);
[h_cheby, ~] = freqz(num_chebyshev, den_chebyshev, 4096, Fs);
[h_ellip, ~] = freqz(num_eliptic, den_eliptic, 4096, Fs);
plot(f, 20*log10(abs(h_butter)), colors{1});
plot(f, 20*log10(abs(h_cheby)), colors{2});
plot(f, 20*log10(abs(h_ellip)), colors{3});
title('Charakterystyka amplitudowa');
xlabel('Czestotliwosc [Hz]');
ylabel('Wzmocnienie [dB]');
legend('Butterworth', 'Czebyszew', 'Eliptyczny');
hold off;

subplot(2, 1, 2);  % Wykres fazowy
hold on;
plot(f, angle(h_butter), colors{1});
plot(f, angle(h_cheby), colors{2});
plot(f, angle(h_ellip), colors{3});
title('Charakterystyka fazowa');
xlabel('Czestotliwosc [Hz]');
ylabel('Faza [rad]');
legend('Butterworth', 'Czebyszew', 'Eliptyczny');
hold off;

% Opoznienie grupowe
figure;
hold on;
[gd_butter, f] = grpdelay(num_butter, den_butter, 2048, Fs);
[gd_cheby, ~] = grpdelay(num_chebyshev, den_chebyshev, 2048, Fs);
[gd_ellip, ~] = grpdelay(num_eliptic, den_eliptic, 2048, Fs);
plot(f, gd_butter, colors{1});
plot(f, gd_cheby, colors{2});
plot(f, gd_ellip, colors{3});
title('Opoznienie grupowe');
xlabel('Czestotliwosc [Hz]');
ylabel('Opoznienie [probki]');
```

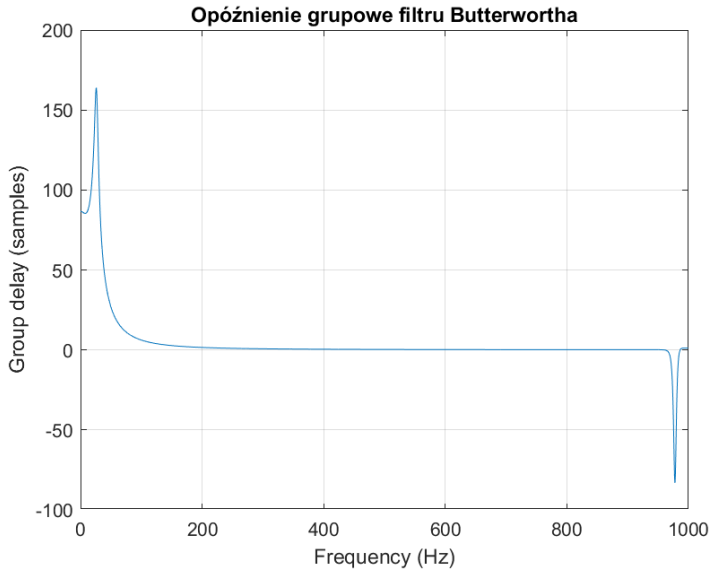
```
legend('Butterworth', 'Czebyszew', 'Eliptyczny');
hold off;

% Zera i bieguny na okregu jednostkowym
figure;
hold on;

% Butterworth
[z_butter, p_butter] = tf2zp(num_butter, den_butter);
plot(real(z_butter), imag(z_butter), 'bo', 'MarkerSize', 6, ...
    'DisplayName', 'Butterworth');
plot(real(p_butter), imag(p_butter), 'bx', 'MarkerSize', 8, ...
    'DisplayName', '');

% Czebyszew
[z_cheby, p_cheby] = tf2zp(num_chebyshev, den_chebyshev);
plot(real(z_cheby), imag(z_cheby), 'ro', 'MarkerSize', 6, ...
    'DisplayName', 'Czebyszew');
plot(real(p_cheby), imag(p_cheby), 'rx', 'MarkerSize', 8, ...
    'DisplayName', '');

% Eliptyczny
[z_ellip, p_ellip] = tf2zp(num_eliptic, den_eliptic);
plot(real(z_ellip), imag(z_ellip), 'go', 'MarkerSize', 6, ...
    'DisplayName', 'Eliptyczny');
plot(real(p_ellip), imag(p_ellip), 'gx', 'MarkerSize', 8, ...
    'DisplayName', '');

% Jednostkowy okrag
theta = linspace(0, 2*pi, 1000);
plot(cos(theta), sin(theta), 'k--', 'DisplayName', 'Okrag jednostkowy');

% Ustawienia wykresu
axis equal;
xlim([-1.5, 1.5]);
ylim([-1.5, 1.5]);
title('Zera i bieguny na okregu jednostkowym');
xlabel('Re');
ylabel('Im');
legend();
grid on;
hold off;


% Wypisanie rzedu filtru Butterwortha
fprintf("Rzad filtru Butterwortha: %d\n", n_min);
```



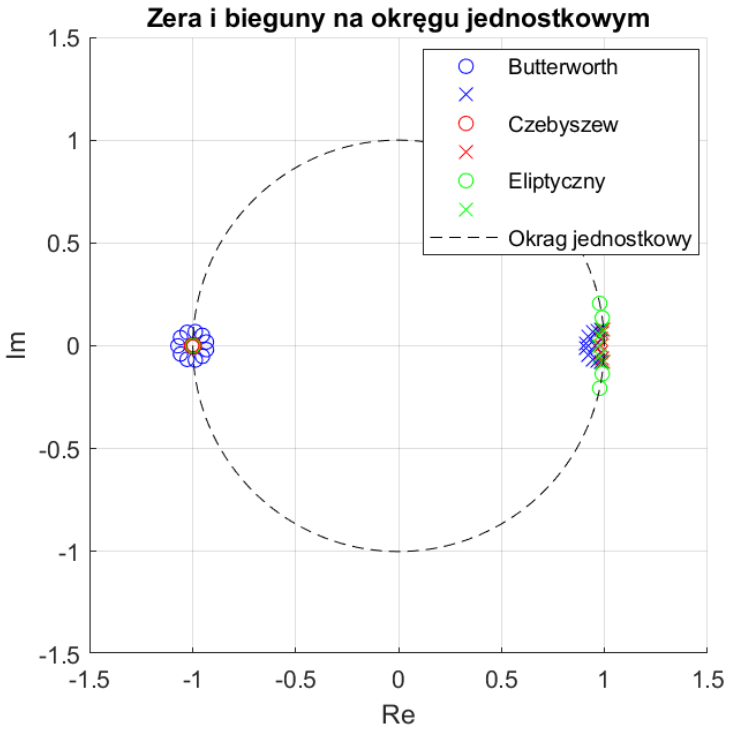
Rysunek 25: Charakterystyka amplitudowa filtrów



Rysunek 26: Charakterystyka fazowa filtrów



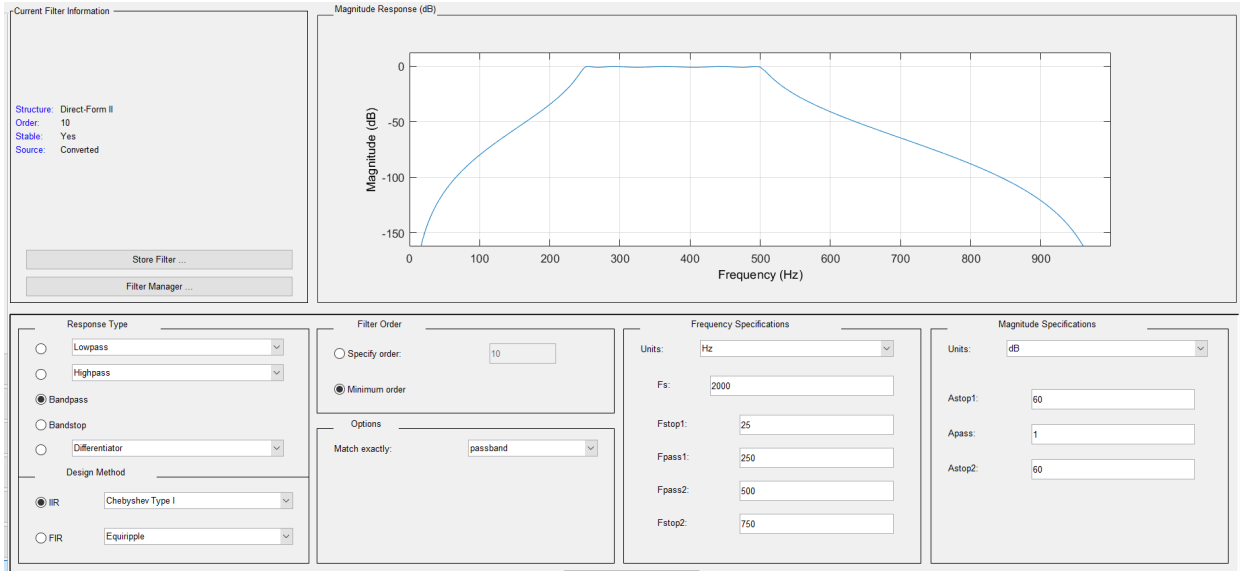
Rysunek 27: Opóźnienie grupowe filtrów



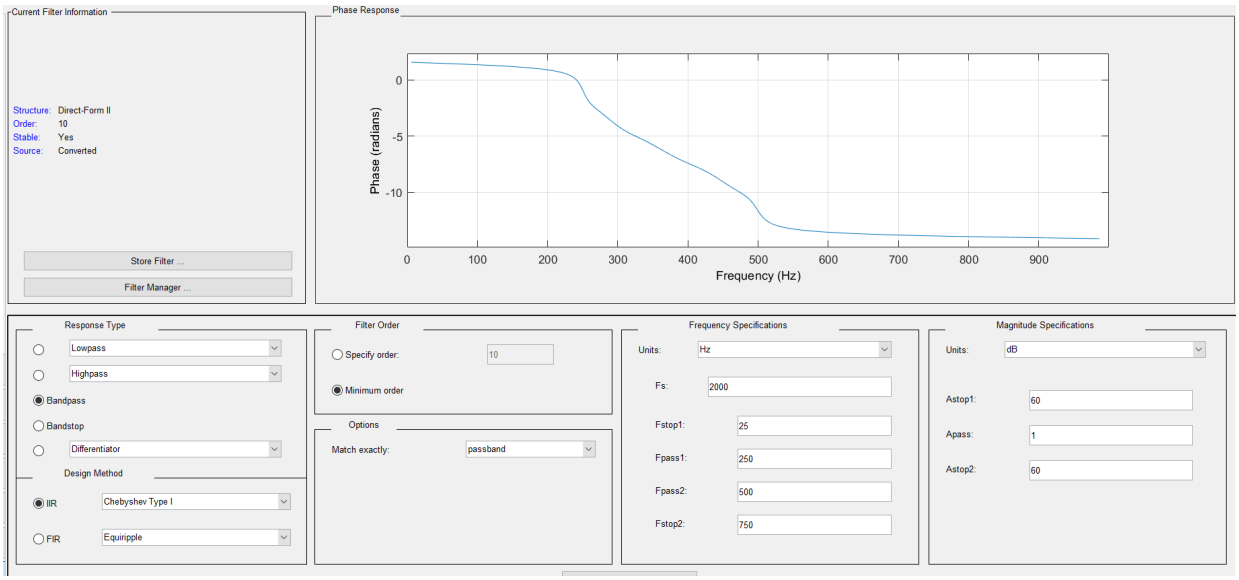
Rysunek 28: Rozmieszczenie zer i biegunów filtrów na płaszczyźnie zespolonej

3 Zadanie 4

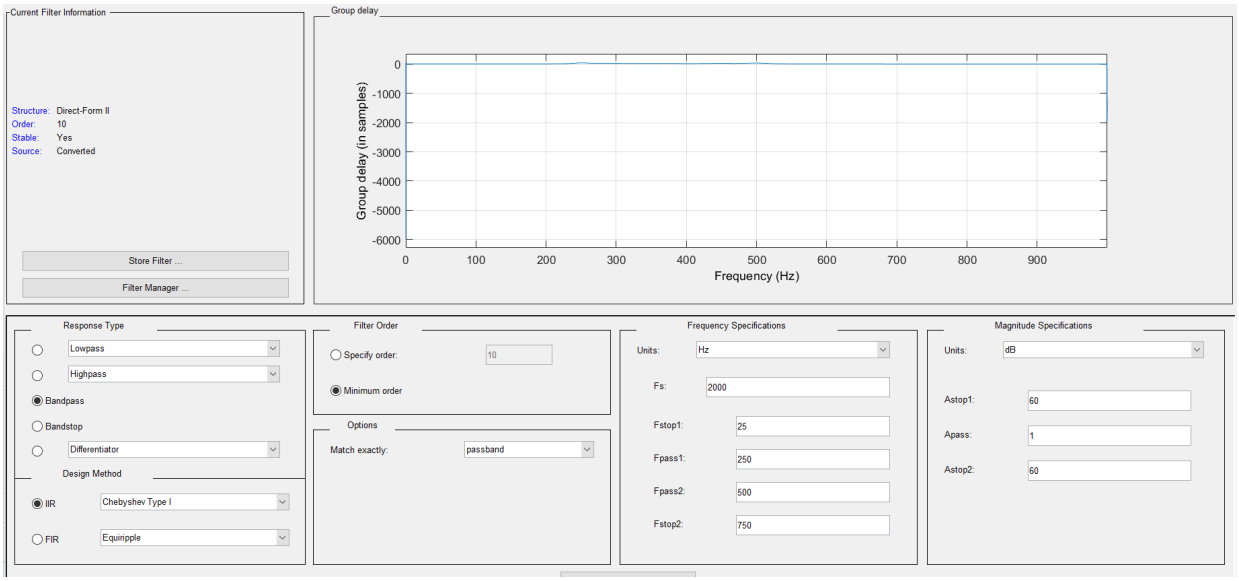
W zadaniu tym zaprojektowano trochę innych filtr - pasmowoprzepustowy IIR Czebyszewa typu I:



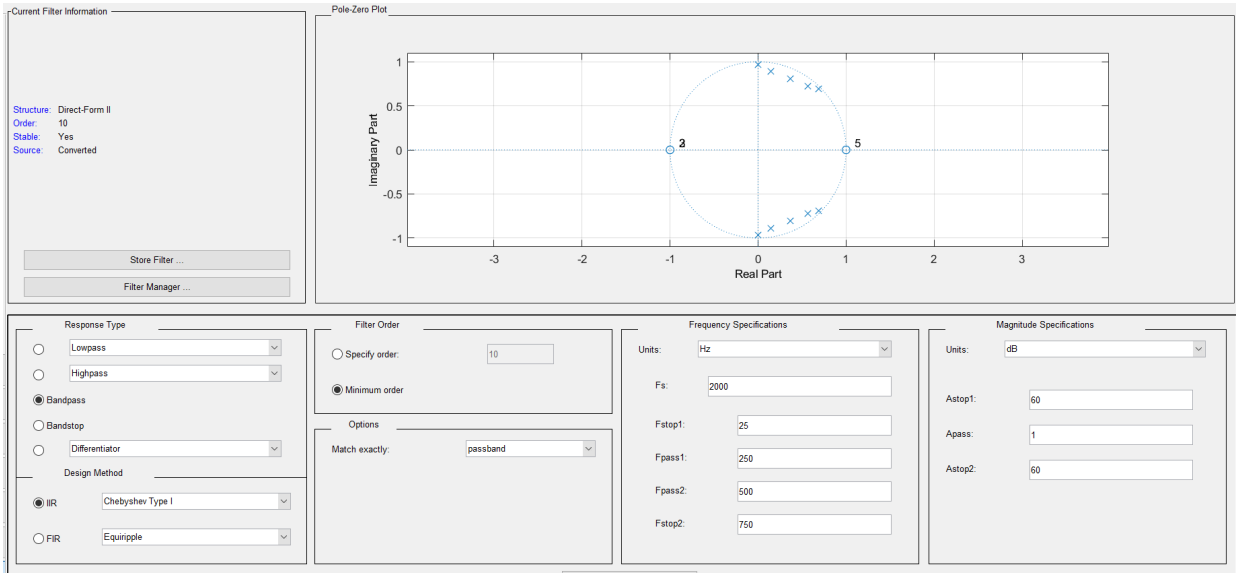
Rysunek 29: Charakterystyka amplitudowa filtra



Rysunek 30: Charakterystyka fazowa filtra



Rysunek 31: Opóźnienie grupowe filtra



Rysunek 32: Rozmieszczenie zer i biegunów filtra na płaszczyźnie zespolonej

Następnie w celu sprawdzenia odpowiedzi filtru na skok jednostkowy, impuls jednostkowy, sinusoidę w paśmie zaporowym i przepustowym napisano kolejny skrypt:

Listing 4: Skrypt MATLAB do zadania 4.

```
Fs = 2000;

% Sygnal impulsowy
impulse = [1; zeros(99,1)];
impulse_response = filter(Num, Den, impulse);

% Sygnal skokowy
step = ones(100, 1);
step_response = filter(Num, Den, step);

% Sinusoida w pasmie zaporowym
sin_stop = gensinsum(1, 0, Fs/200, 400, Fs);
sin_stop_response = filter(Num, Den, sin_stop);

% Sinusoida w pasmie przepustowym
sin_pass = gensinsum(1, 0, Fs/6, 400, Fs);
sin_pass_response = filter(Num, Den, sin_pass);


fprintf("Max numerator coefficient: %.8f \n", max(abs(Num)));
fprintf("Min numerator coefficient: %.8f \n", min(abs(Num)));
fprintf("Max denominator coefficient: %.8f \n", max(abs(Den)));
fprintf("Min denominator coefficient: %.8f \n", min(abs(Den)));


% Wykresy wynikow
figure;
stem(impulse_response);
```

```
title('Odpowiedz impulsowa');

figure;
plot(step_response);
title('Odpowiedz skokowa');

figure;
plot(sin_stop_response);
title('Sinusoida w pasmie zaporowym');

figure;
plot(sin_pass_response);
title('Sinusoida w pasmie przepustowym');


% Obliczanie odpowiedzi dla filtrow SOS
impulse_response_sos = sosfilt(SOS1, impulse);
step_response_sos = sosfilt(SOS1, step);
sin_stop_response_sos = sosfilt(SOS1, sin_stop);
sin_pass_response_sos = sosfilt(SOS1, sin_pass);


% Wykresy wynikow dla SOS
figure;
stem(impulse_response_sos);
title('Odpowiedz impulsowa (SOS)');

figure;
plot(step_response_sos);
title('Odpowiedz skokowa (SOS)');

figure;
plot(sin_pass_response_sos);
title('Sinusoida w pasmie przepustowym (SOS)');

figure;
plot(sin_stop_response_sos);
title('Sinusoida w pazmie zaporowym (SOS)');
```

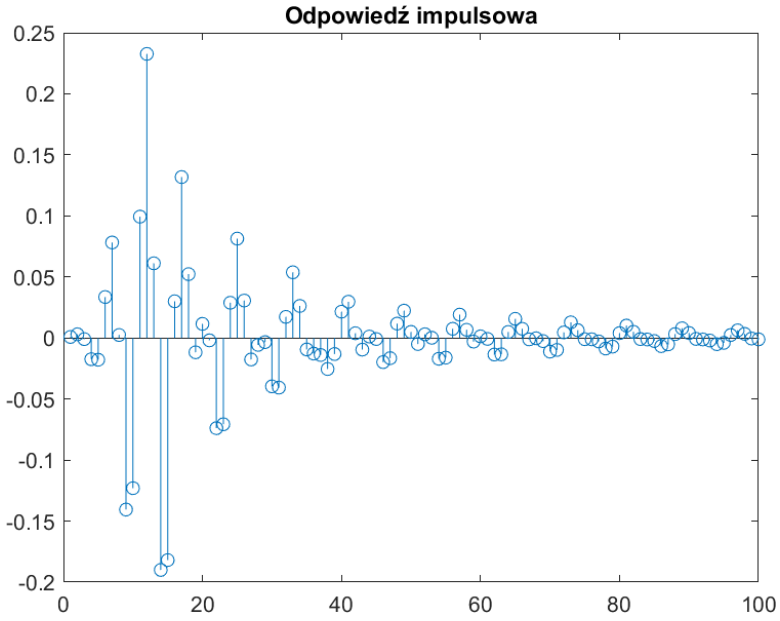
Zwrócił on dodatkowo informację na temat najmniejszych i największych współczynników wielomianów transmitancji filtru:

```
Max numerator coefficient: 0.00848230
Min numerator coefficient: 0.00000000
Max denominator coefficient: 19.78273831
Min denominator coefficient: 0.48031187
```

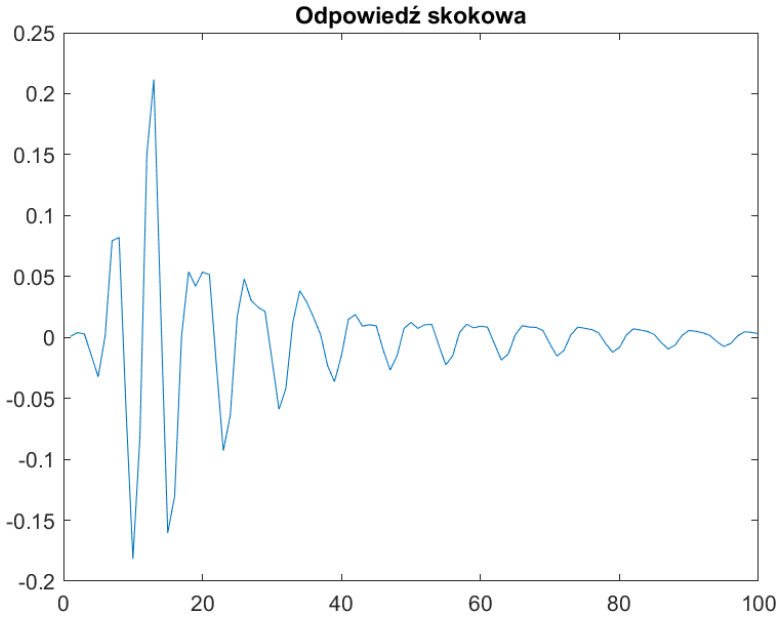
Przy dużych rzędach filtru wielomiany transmitancji są wysokich stopni, przez co niektóre z nich mogą być niestabilne numerycznie. Dodatkowo często występuje duża rozbieżność wartości między współczynnikami, co wprowadza dodatkowy błąd wynikający z zaokrągleń podczas pracy z typami zmiennoprzecinkowymi. Dlatego filtry wysokiego rzędu dzieli się na części bikwadratowe, które zawierają tylko wielomiany 2. stopnia. Są one wtedy tak dobrane, że nawet jeżeli współczynniki są urojone, to aby tworzyły liczby sprzężone, przez co algorytm filtru może pracować już tylko ze współczynnikami rzeczywistymi.

Wykresy wygenerowane dla podejścia zwykłego oraz SOS (Second-Orders Sections) były takie same, co

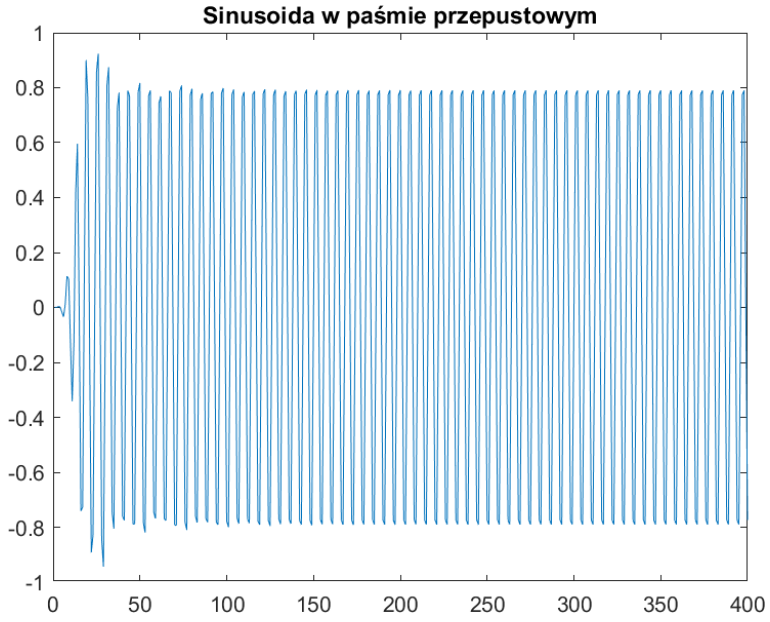
potwierdza, że dekompozycja została przeprowadzona prawidłowo, dlatego pozwolono sobie zamieścić tylko jedną kopię wykresów:



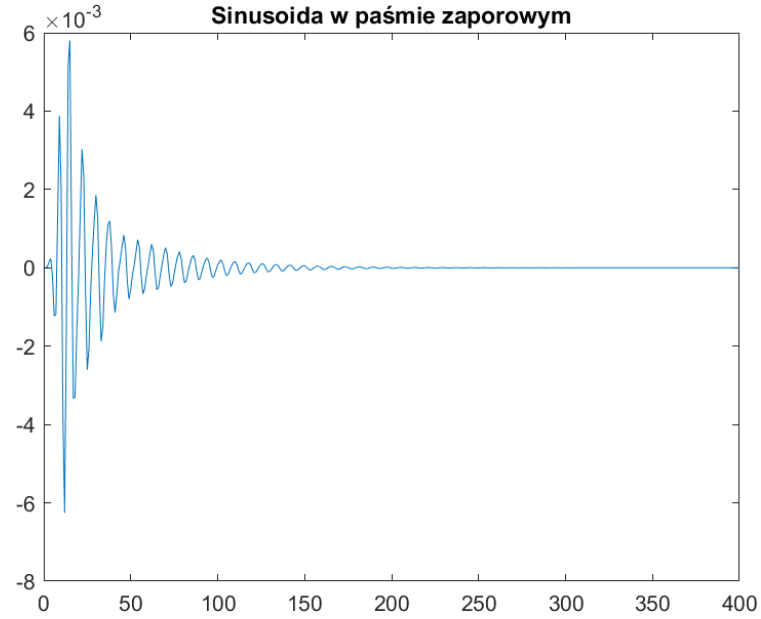
Rysunek 33: Odpowiedź impulsowa filtru



Rysunek 34: Odpowiedź skokowa filtru



Rysunek 35: Odpowiedź filtru na sygnał sinusoidalny w paśmie przepustowym



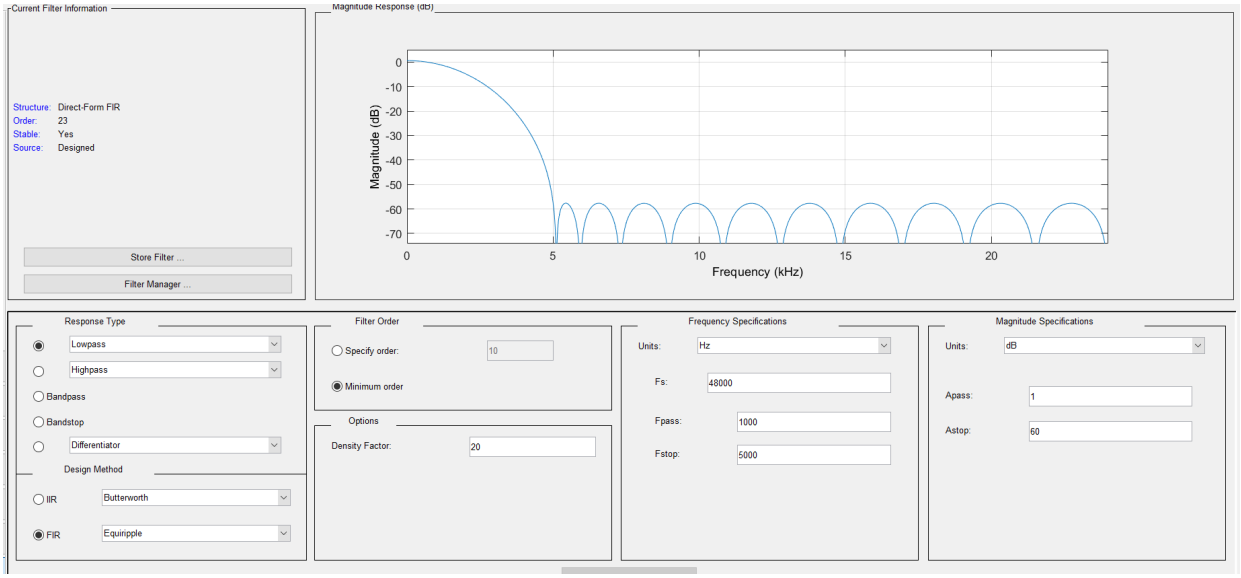
Rysunek 36: Odpowiedź filtru na sygnał sinusoidalny w paśmie zaporowym

Powyższe wykresy pokazują główną cechę i wadę filtrów IIR, czyli ich nieskończoną odpowiedź impulsową. W przypadku sinusoidy o częstotliwości z pasma zaporowego odpowiedź filtru ma bardzo małą amplitudę, która zaraz wygasa. Natomiast w przypadku skoku jednostkowego lub pojedynczego impulsu odpowiedź filtru ma postać oscylacji, które co prawda powoli gasną, ale czas ich trwania może być nieskończenie długi. Jest to często bardzo problematyczne zachowanie filtru.

4 Zadanie 5

Przetwarzając cyfrowo rzeczywiste sygnały, nie wykonuje się operacji na całym zarejestrowanym sygnale. W przypadku przetwarzania na żywo powstałyby zbyt duże i uciążliwe opóźnienia, gdyby chcieć przetworzyć cały sygnał od razu. Z kolei przetwarzanie sygnału bit po bicie też nie ma większego sensu. Dlatego długi sygnał dzieli się na mniejsze części i buforuje, a następnie każdy z buforów przetwarza się oddzielnie odpowiednim algorytmem, na przykład algorytmem filtru cyfrowego. Jednak w tym przypadku może powstać problem - filtr na początku i na końcu swojego działania znajduje się w stanie nieustalonym. Wynika to z faktu wykorzystywania opóźnionych próbek sygnału. Dlatego ważne jest, aby przenosić informację o stanie filtru między blokami, aby uniknąć stanów nieustalonych w miejscach, gdzie bufory są znów łączone. Aby sprawdzić, czym grozi nie zastosowanie się do tego, przeprowadzono eksperyment.

Na początku stworzono kolejny filtr:



Rysunek 37: Charakterystyka amplitudowa filtru

A następnie napisano program:

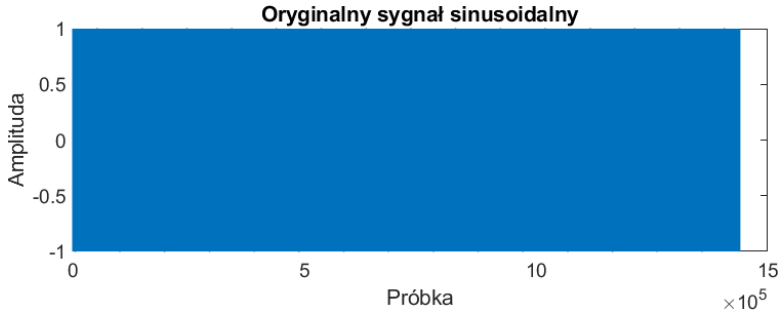
Listing 5: Skrypt MATLAB do zadania 5.

```
elimination_transient_states = 0;

Fs = 48000;
signal_frequency = 2000;
T = 30;
N = T * Fs;
block_size = 4096;
num_blocks = ceil(N / block_size);

signal = gensinsum(1, 0, signal_frequency, N, Fs);

filtered_signal = zeros(size(signal));

if elimination_transient_states == 1
```

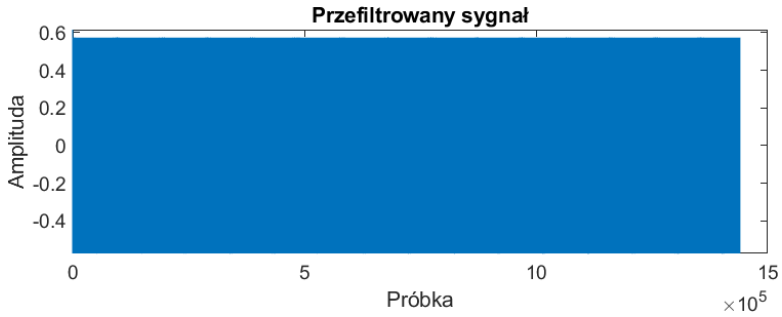
```
    zi = [];  % Poczatkowy stan filtru

    for i = 1:num_blocks
        start_idx = (i - 1) * block_size + 1;
        end_idx = min(i * block_size, N);

        block = signal(start_idx:end_idx);

        [filtered_block, zi] = filter(Num, 1, block, zi);

        filtered_signal(start_idx:end_idx) = filtered_block;
    end

elseif elimination_transient_states == 0
    for i = 1:num_blocks
        start_idx = (i - 1) * block_size + 1;
        end_idx = min(i * block_size, N);

        block = signal(start_idx:end_idx);

        filtered_block = filter(Num, 1, block);

        filtered_signal(start_idx:end_idx) = filtered_block;
    end

end


figure;
subplot(2, 1, 1);
plot(signal);
title('Oryginalny sygnal sinusoidalny');
xlabel('Probka');
ylabel('Amplituda');

subplot(2, 1, 2);
plot(filtered_signal);
title('Przefiltrowany sygnal');
xlabel('Probka');
ylabel('Amplituda');


sound(filtered_signal, Fs);
```

W zadbaniu o prawidłową inicjalizację filtrów pomaga sama funkcja `filter`, która może przyjmować wektor `zi` niosący informacje o stanie filtru z poprzedniego bloku i w ten sposób zabezpieczający przed powstaniem stanów nieustalonych.

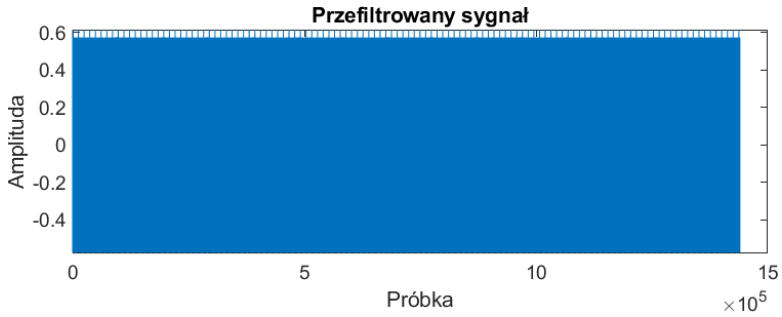
Następnie wygenerowano wykresy sygnału dla dwóch wykonań programu, raz z włączoną opcją `elimination_transient_states` do eliminacji stanów nieustalonych, a raz z wyłączoną tą opcją. Program narysował poniższe wykresy wykonane dla sygnału o częstotliwości 2kHz:



Rysunek 38: Sygnał na wejściu filtru



Rysunek 39: Sygnał na wyjściu filtru z włączoną opcją unikania stanów nieustalonych



Rysunek 40: Sygnał na wyjściu filtru z wyłączoną opcją unikania stanów nieustalonych

Jak widać, jeżeli filtry w każdym buforze nie są właściwie inicjalizowane, w sygnale wyjściowym pojawiają się piki związane ze stanami nieustalonymi. Za pomocą funkcji `sound` wygenerowano dźwięk z obu sygnałów. W pierwszym przypadku był to idealny pisk o częstotliwości 2kHz, natomiast w drugim przypadku oprócz pisku występowało regularne „tykanie” w głośniku.

5 Zadanie 6

`fftfilt()` jest funkcją do filtrowania sygnałów za pomocą algorytmu splotu w dziedzinie częstotliwości. W przeciwieństwie do tradycyjnego splotu w dziedzinie czasu, jak to jest w przypadku zwykłych filtrów, `fftfilt()` przekształca zarówno sygnał wejściowy, jak i współczynniki filtru do dziedziny częstotliwości przy pomocy FFT. Wynikiem jest bardziej efektywne filtrowanie przy długich sygnałach lub wysokich rzędach

filtru. Jednak i tutaj w przypadku dzielenia sygnału na części trzeba uważać, aby właściwie inicjalizować algorytm. Jednak funkcja `fftfilt()` nie posiada odpowiedniego argumentu pomagającego w tym, jak to było w poprzednim zadaniu, a więc tym razem problem ten rozwiązuje się, dodając do sygnału podawanego w bloku jako argument do `fftfilt()` końcówkę sygnału z poprzedniego bloku. W ten sposób uzyskuje się właściwy sygnał po przefiltrowaniu.

W zadaniu tym napisano podobny skrypt jak w zadaniu wyżej, inaczej tylko implementując mechanizm sklejania bloków:

Listing 6: Skrypt MATLAB do zadania 5.

```
elimination_transient_states = 0;

Fs = 48000;
signal_frequency = 2000;
T = 30;
N = T * Fs;
block_size = 4096;
num_blocks = ceil(N / block_size);

signal = gensinsum(1, 0, signal_frequency, N, Fs);
filtered_signal = zeros(size(signal));

filter_order = length(Num) - 1;
overlap = filter_order;

if elimination_transient_states == 1
    prev_tail = zeros(1, overlap);

    for i = 1:num_blocks
        start_idx = (i - 1) * block_size + 1;
        end_idx = min(i * block_size, N);

        block = signal(start_idx:end_idx);
        extended_block = [prev_tail, block];

        filtered_extended_block = fftfilt(Num, extended_block);
        filtered_signal(start_idx:end_idx) = filtered_extended_block( ...
            overlap+1:end);

        prev_tail = block(end-overlap+1:end);
    end

else
    for i = 1:num_blocks
        start_idx = (i - 1) * block_size + 1;
        end_idx = min(i * block_size, N);

        block = signal(start_idx:end_idx);
        filtered_block = fftfilt(Num, block);
        filtered_signal(start_idx:end_idx) = filtered_block;
    end
end

figure;
subplot(2, 1, 1);
plot(signal);
```

```
47 title('Oryginalny sygnal sinusoidalny');
48 xlabel('Probka');
49 ylabel('Amplituda');
50
51 subplot(2, 1, 2);
52 plot(filtered_signal);
53 title('Przefiltrowany sygnal');
54 xlabel('Probka');
55 ylabel('Amplituda');
56
57 sound(filtered_signal, Fs);
```

Wykresy wygenerowane przez program są w zasadzie takie same jak te z poprzedniego zadania, tak samo jak wrażenia słuchowe podczas odtwarzania sygnałów otrzymanych przy właściwym i przy niewłaściwym łączeniu bloków sygnału.