

A G H

**AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE**

Wydział Inżynierii Mechanicznej i Robotyki

Informatyka w inżynierii mechanicznej

Sprawozdanie z przedmiotu Cyfrowe Przetwarzanie Sygnałów

285608, Mateusz Krupnik

Spis treści

Tabele z kodami	2
1. Wstęp	3
1. Laboratorium 1	3
2. Laboratorium 2	4
3. Laboratorium 3	6
4. Laboratorium 4	9
5. Laboratorium 5	19
6. Laboratorium 6	34
7. Laboratorium 7	54
8. Kody programów dodatkowych.	67

Tabele z kodami

Tab. 2.1. Kod programu „Krupnik_Lab_1.m”.	3
Tab. 3.1. Kod programu „Krupnik_Lab_2.m”.	4
Tab. 3.2. Parametry sygnałów sinusoidalnych.	6
Tab. 4.1. Kod programu „Krupnik_Lab_3.m”.	6
Tab. 5.1. Kod programu „Krupnik_Lab_4”.	10
Tab. 5.2. Wzory sygnałów używanych w programie.	16
Tab. 6.1. Kod programu „Krupnik_Lab_5_1”.	20
Tab. 6.2. Kod programu "Krupnik_Lab_5_2.m".	29
Tab. 7.1. Kod programu „Laboratoria_nr_6_1.m”.	35
Tab. 7.2. Kod programu "Krupnik_Lab_6_2.m".	44
Tab. 7.3. Kod programu "Krupnik_Lab_6_3.m".	46
Tab. 7.4. Kod programu "Krupnik_Lab_6_4.m".	48
Tab. 7.5. Kod programu "Krupnik_Lab_6_5.m".	50
Tab. 8.1. Kod programu "Krupnik_Lab_7.m".	54
Tab. 9.1. Kod programów na podstawie rozdziału 1.	67
Tab. 9.2. Kod programów na podstawie rozdziału 2.	71
Tab. 9.3. Kod programów na podstawie rozdziału 3.	73
Tab. 9.4. Kod programów na podstawie rozdziału 4.	75
Tab. 9.5. Kod programów na podstawie rozdziału 5.	76
Tab. 9.6. Kod programów na podstawie rozdziału 6.	77
Tab. 9.7. Kod programów na podstawie rozdziału 9.	88
Tab. 9.8. Kod programów na podstawie rozdziału 10.	91
Tab. 9.9. Kod programów na podstawie rozdziału 11.	94
Tab. 9.10. Kod programów na podstawie rozdziału 12.	106
Tab. 9.11. Kod programów na podstawie rozdziału 14.	114
Tab. 9.12. Kod programów na podstawie rozdziału 17.	115
Tab. 9.13. Kod programów na podstawie rozdziału 22.	119

1. Wstęp

Sprawozdanie przedstawia programy realizowane w ramach laboratoriów. Programy służące do przetwarzania sygnałów oparte są na udostępnionych przykładach. Dodatkowo zrealizowane zostały programy z książki prof. Zielińskiego pt. „Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań”. Wszystkie programy zostały nagrane na płytę CD dołączoną do sprawozdania, kody przykładów z laboratoriów będą opisywane i przedstawiane na bieżąco, a programy oparte na przykładach z wspomnianej książki dostępne są na końcu sprawozdania. Programy wykorzystują budowę sekcji, a więc jeden plik zawiera wszystkie ćwiczenia dla danych laboratoriów, a kolejne etapy można realizować uruchamiając kolejną sekcję. Uruchomienie programu przyciskiem RUN spowoduje zrealizowanie wszystkich sekcji.

1. Laboratorium 1

W ramach tych ćwiczeń zrealizowany został program mający zaznajomić z operacjami wejścia i wyjścia, czyli zapisywania i odczytywania danych z plików. Plik realizujący ćwiczenie nazwany jest „Krupnik_Lab_1.m”. Kolejne ćwiczenia posiadają analogiczną numerację.

Podstawowe polecenia to funkcje:

- fopen – funkcja otwierająca plik o podanej nazwie, z podanym trybem (zapis, odczyt),
- fprintf – funkcja zapisywania danych do otwartego pliku, podać należy: format, zmienną z przestrzeni roboczej,
- fclose – funkcja zamykająca otwarte pliki,
- fscanf – funkcja odczytywania danych z pliku o zadanym formacie.

W programie generowany jest sygnał sinusoidalny, wykreślone są jego wykresy i dokonywany jest zapis do pliku. Następnie następuje otwarcie ponownie pliku i odczyt danych i ponowne wykreślenie wykresu.

Tab. 1.1. Kod programu „Krupnik_Lab_1.m”.

```
%% Lab 1 - operacje wejścia wyjścia - Mateusz Krupnik
% Generowanie przebiegu sinosoidalnego
clc; close all; clear all;
t=0:0.001:1;
A=0.7;
f=100;
omega=2*pi*f;
y=A*sin(omega*t);
% wykres
figure(1)
plot(t,y)
% Otwarcie pliku w trybie zapisywania
uchwyt=fopen('uchwyt.txt','w');
% Zapis kolumny czasu i przebiegu sinosidalnego
fprintf(uchwyt,'%12.4f %12.4f\n',[t;y]);
% Zamknięcie wsztatkich plików
fclose('all');
% Otworzenie pliku w trybie odczytu i wczytanie wartości do macierzy DANE
uchwyt=fopen('uchwyt.txt','r');
DANE=fscanf(uchwyt,'%g %g \n',[2 inf]);
fclose('all');
```

```
% Wykres
figure(2)
plot(DANE(1,:),DANE(2,:))
```

2. Laboratorium 2

W tym ćwiczeniu ponownie generowane są sygnały sinusoidalne, a funkcja sound powoduje ich reprezentację w postaci dźwięku. Przebiegi są zapisywane do pliku z rozszerzeniem .txt oraz .bin z różnymi formatami danych. Następnie dokonywany jest odczyt z plików i generowane są wykresy. Na samym końcu wyznaczane są podstawowe parametry jak minimum, maksimum, średnia czy energia sygnału.

Tab. 2.1. Kod programu „Krupnik_Lab_2.m”.

```
%% Lab 2 - operacje wejścia wyjścia i parametry sygnałów - Mateusz Krupnik
% Wyczyszczenie ekranu i generowanie przebiegów sinusoidalnych
clc
clear all

A=0.5;
B=-0.3;
f1=700;
f2=1200;
fs=10000;
t=0:(1/fs):1;
y1=A*sin(2*pi*f1*t);
y2=B*sin(2*pi*f2*t);
y3=y1+y2;
y4=y1-y2;
sound(y1,fs); pause(t(end));
sound(y2,fs); pause(t(end));
sound(y3,fs); pause(t(end));
sound(y4,fs); pause(t(end));
% Zapis przebiegów do pliku, %12.4f - zapis wartości o dł 12 znaków, 4
% znaki precyzji, \n - nowy wiersz
uchwyty=fopen('dane1.txt','w');
fprintf(uchwyty,'%12.4f %12.4f %12.4f %12.4f %12.4f\n',[t;y1;y2;y3;y4]);
% Zamknięcie pliku, i ponowne otwarcie, odczyt pliku do macierzy D
% %g - odczyt zapisu w postaci dziesiętnej lub wykładniczej, usunięcie zer
% z końca zapisu, %e - notacja wykładnicza, %f - dziesiętna
fclose('all');
uchwyty=fopen('dane1.txt','r');
D=fscanf(uchwyty,'%g %g %g %g %g \n',[5 inf]);
fclose('all');
% Zapis danych w postaci binarnej, a następnie ich odczytanie, inf -
odczyt
% do ostatniej kolumny
uchwyty1=fopen('dane1.bin','w');
fwrite(uchwyty1,[t;y1;y2;y3;y4], 'float');
fclose('all');
uchwyty1=fopen('dane1.bin','r');
y5=fread(uchwyty1,[5 inf], 'float');
fclose('all');

% Wykresy wygenerowanych przebiegów
figure(3)
subplot(2,2,1)
plot(t,y1); title('y1');
```

```

subplot(2,2,2)
plot(t,y2,'r'); title('y2');
subplot(2,2,3)
plot(t,y3,'k'); title('y3');
subplot(2,2,4)
plot(t,y4,'g'); title('y4');
sgtitle('Dane wygenerowane')

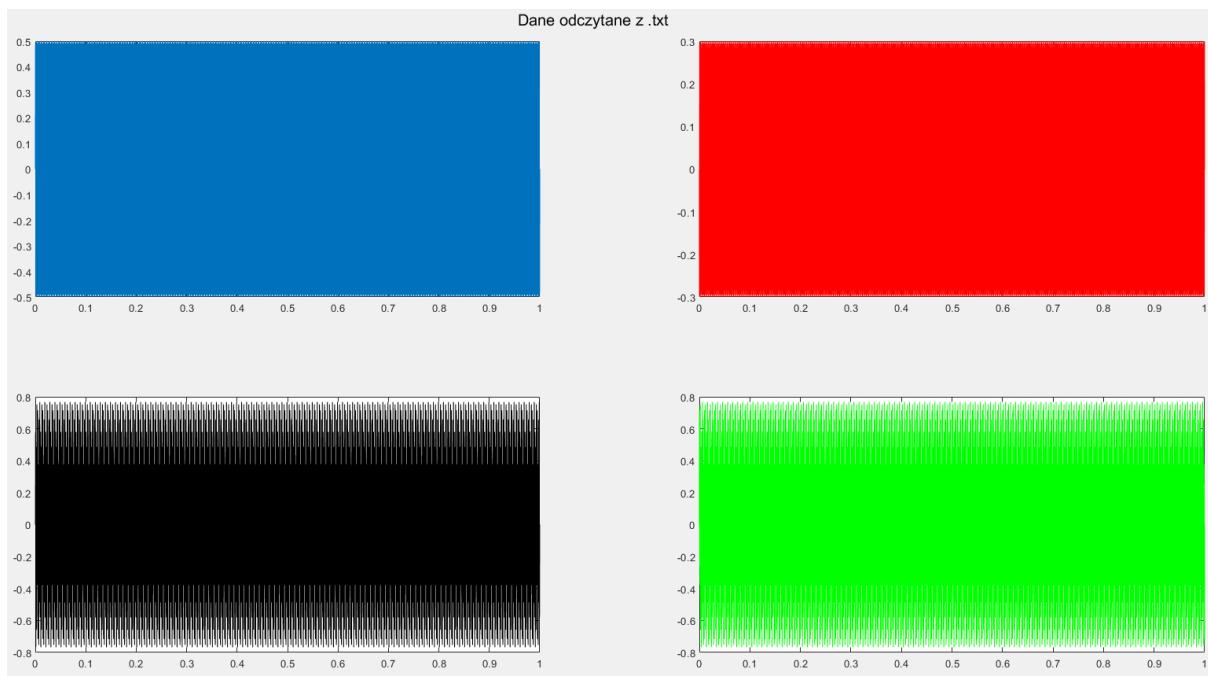
% Wykresy danych odczytanych z pliku .txt
figure(4)
subplot(2,2,1)
plot(D(1,:),D(2,:))
subplot(2,2,2)
plot(D(1,:),D(3,:),'r')
subplot(2,2,3)
plot(D(1,:),D(4,:),'k')
subplot(2,2,4)
plot(D(1,:),D(5,:),'g')
sgtitle('Dane odczytane z .txt')

% Wykresy danych odczytanych z pliku .bin - wykres y5
figure(5)
subplot(2,2,1)
plot(t,y1); title('y1');
subplot(2,2,2)
plot(t,y2,'r'); title('y2');
subplot(2,2,3)
plot(t,y3,'k'); title('y3');
subplot(2,2,4)
plot(t,y4,'g'); title('Odczytana kolumna y4 z pliku .bin');
sgtitle('Dane odczytane z .bin')

%% Wynazczanie parametrów sygnałów za pomocą stworzonych funkcji
% Wyznaczenie wartości minimalnej i maksymalnej
% Wywoływanie funkcji signal_min i signal_max odpowiadają - min() i max()
a = [min(y1) min(y2) min(y3) min(y4)]
b = [max(y1) max(y2) max(y3) max(y4)]
% Wyznaczenie wartości średniej za pomocą funkcji signal_mean - mean()
y_mean = [mean(y1) mean(y2) mean(y3) mean(y4)]
% Energia sygnalu - za pomocą funkcji signal_energy()
e = [signal_energy(y1) signal_energy(y2) ...
      signal_energy(y3) signal_energy(y4)]


function energy = signal_energy(signal, dt)
% Signal energy
% signal - signal, dt - time step
energy = 0;
if nargin > 1
    dt = dt;
else
    dt = 1;
end
for i=1:length(signal)
    energy = energy + (signal(i))^2*dt;
end
end

```



Rys. 2.1. Przykładowe przebiegi sinusoidalne po odczytaniu z pliku.

Wyznaczone parametry sygnałów przedstawione są w tabeli 3.2.

Tab. 2.2. Parametry sygnałów sinusoidalnych.

Parametr \ Sygnał	Niebieski (1)	Czerwony (2)	Czarny (3)	Zielony (4)
Minimum	-0.5000	-0.2994	-0.7675	-0.7675
Maksimum	0.5000	0.2994	0.7675	0.7675
Średnia ($\times 10^{-14}$)	-0.1417	-0.1480	-0.1480	-0.1354
Energia	1250.0	450.00	1700.0	1700.0

3. Laboratorium 3

W ramach tych ćwiczeń realizowane zostały dodatkowo operacje na plikach dźwiękowych. Wykorzystywane funkcje to m. in. audiowrite służąca do zapisu sygnału w postaci pliku dźwiękowego np. o rozszerzeniu .wav. Ponownie funkcja sound pozwala na odtworzenie dźwięku. Generowane zostały sygnały sinusoidalne, zmodulowane i tłumione. Funkcja audiowrite pozwala połączyć kanały (sygnały) w jeden plik. Poniżej przedstawione zostaną wykresy i kod programu.

Tab. 3.1. Kod programu „Krupnik_Lab_3.m”.

```
%% Lab 3 - praca z plikiem dźwiękowym - Mateusz Krupnik
clc; clear all; close all;
% Generowanie przebiegu, zapis i odczyt w postaci pliku wav
A=0.5;
B=-0.3;
f1=700;
fs=10000;
t=0:(1/fs):1;
y1=A*sin(2*pi*f1*t);
```

```

audiowrite('plik.wav', y1, fs);
clear all; % usuniecie danych
%% Odczyt danych
% odczyt danych z pliku wav i jego odtworzenie
[y, Fs] = audioread('plik.wav'); sound(y); pause(1);

% Odtworzenie osi czasu i generacja sygnału
t = 0:(1/Fs):1;
f1 = 700; f2 = 70;
A = 1; y1 = A*sin(2*pi*f1*t); sound(y1); pause(t(end));
alfa1 = 2; alfa2 = 6;

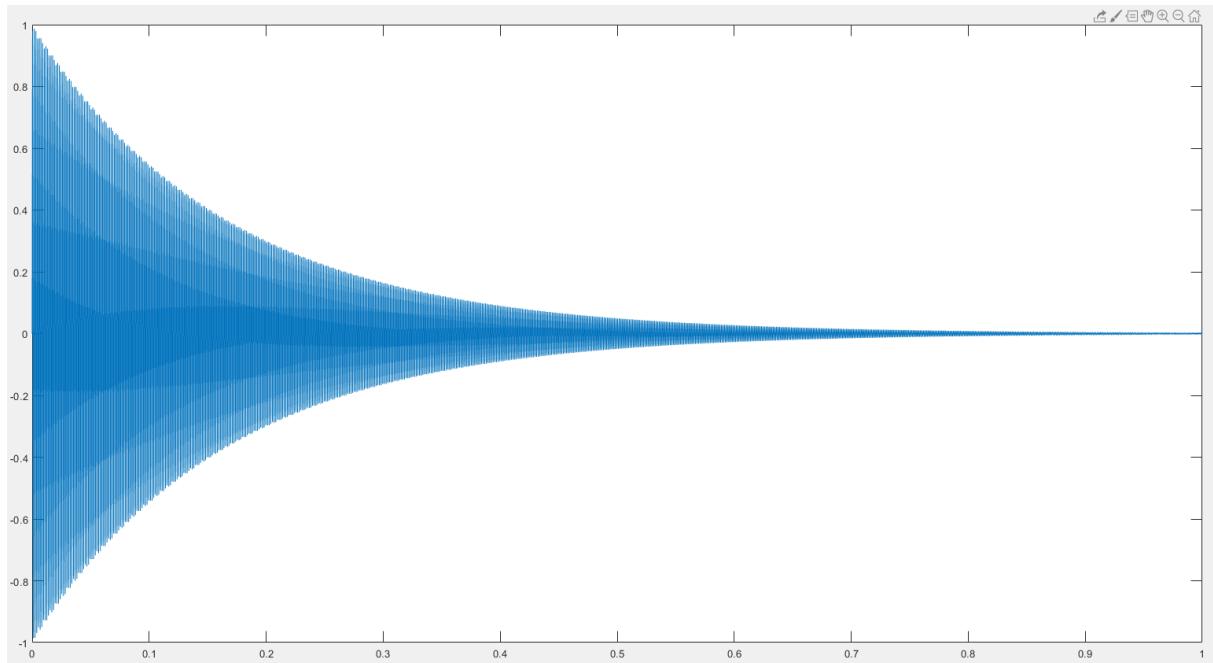
%% Tłumienie wykładnicze sygnału
% Generacja sygnału, odtworzenie dźwięku i wykres
yt=y1.*exp(-alfa2*t);
sound(yt,Fs)
figure(1)
plot(t,yt);

%% Modulacja sygnałów
% Generowanie sygnałów zmodulowanych
ym=2*A*y1.*sin(2*pi*f2*t);
ym1=sin(2*pi*10*t).*sin(2*pi*1000*t);
ym2=sin(2*pi*10*t).*sin(2*pi*1000*t).*exp(-alfa1*t);
ym3=sin(2*pi*10*t).*sin(2*pi*1000*t).*exp(-alfa2*t);
% Generowanie wykresów
figure(2)
subplot(2,2,1); plot(t, ym); title('Modulacja amplitudy');
subplot(2,2,2); plot(t, ym1); title('Modulacja amplitudy');
subplot(2,2,3); plot(t, ym2);
title('Modulacja amplitudy wraz z tłumieniem');
subplot(2,2,4); plot(t, ym3);
title('Modulacja amplitudy wraz z tłumieniem');
sgtitle('Sygnały modulowane');
% Odtworzenie sygnałów
sound(ym); pause(t(end)); sound(ym1); pause(t(end));
sound(ym2); pause(t(end)); sound(ym3);

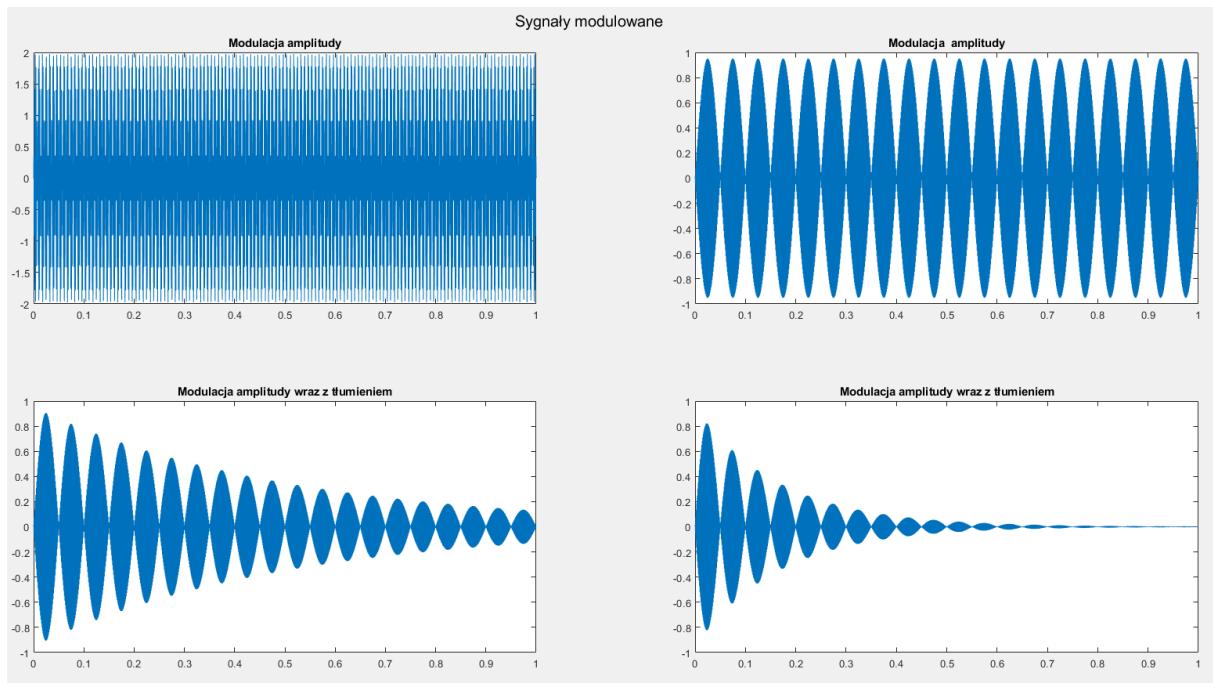
%% Połączenie sygnałów w 2 kanały i zapis pliku wav
% Połączenie przebiegów i ich zapis
Y = [ym2; ym3];
audiowrite('plik2.wav', Y, Fs);
% Odczyt z pliku, wykresy kanałów oraz odtworzenie dźwięku
[Y1, Fs] = audioread('plik2.wav');
figure(3)
plot(Y1(:, 1)); hold on; plot(Y1(:, 2), 'r');
hold off; title('Połączone sygnały');
sound(Y1, Fs); pause(t(end));

%% Połączone sygnały: sygnał tłumiony i narastający
% Generowanie przebiegu sygnału narastającego oraz jego zapis
ym4 = y1.* (1-exp(-alfa1*t));
Y2 = [ym2; ym4];
audiowrite('plik3.wav', Y2, Fs);
% Odczyt sygnału, wykres i odtworzenie dźwięku
[Y3, Fs] = audioread('plik3.wav');
figure(4)
plot(Y3(:, 1)); hold on; plot(Y3(:, 2), 'r.-');
hold off; title('Połączone sygnały');
sound(Y3, Fs); pause(t(end));

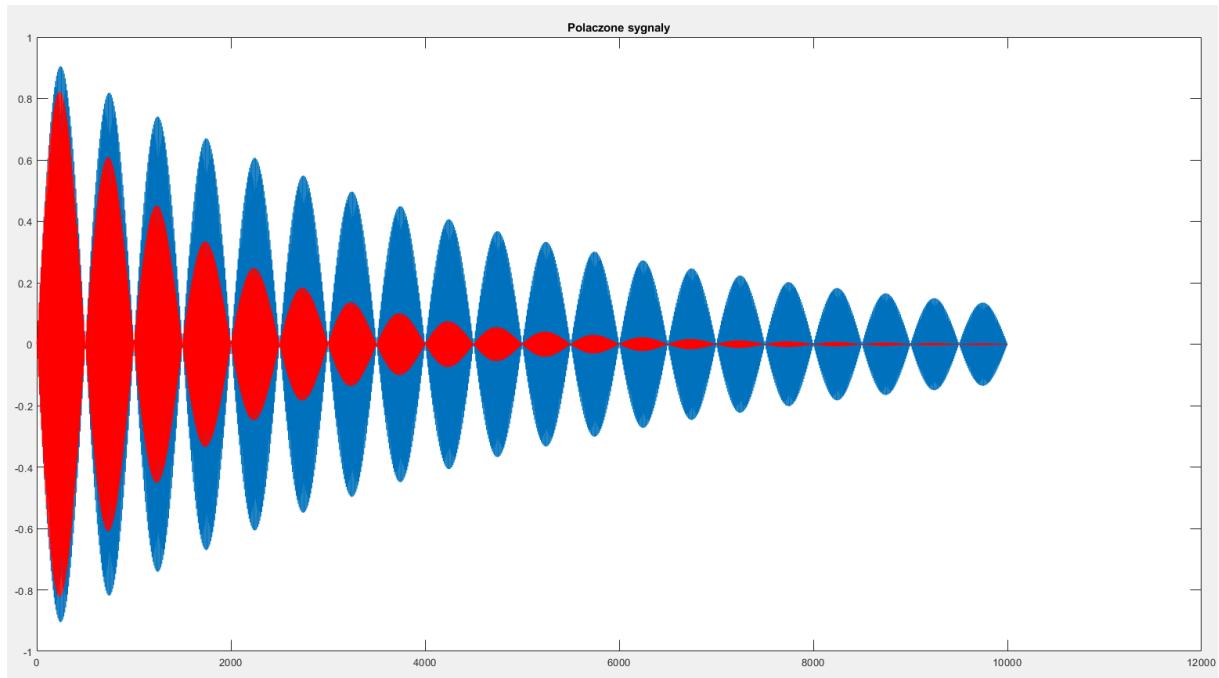
```



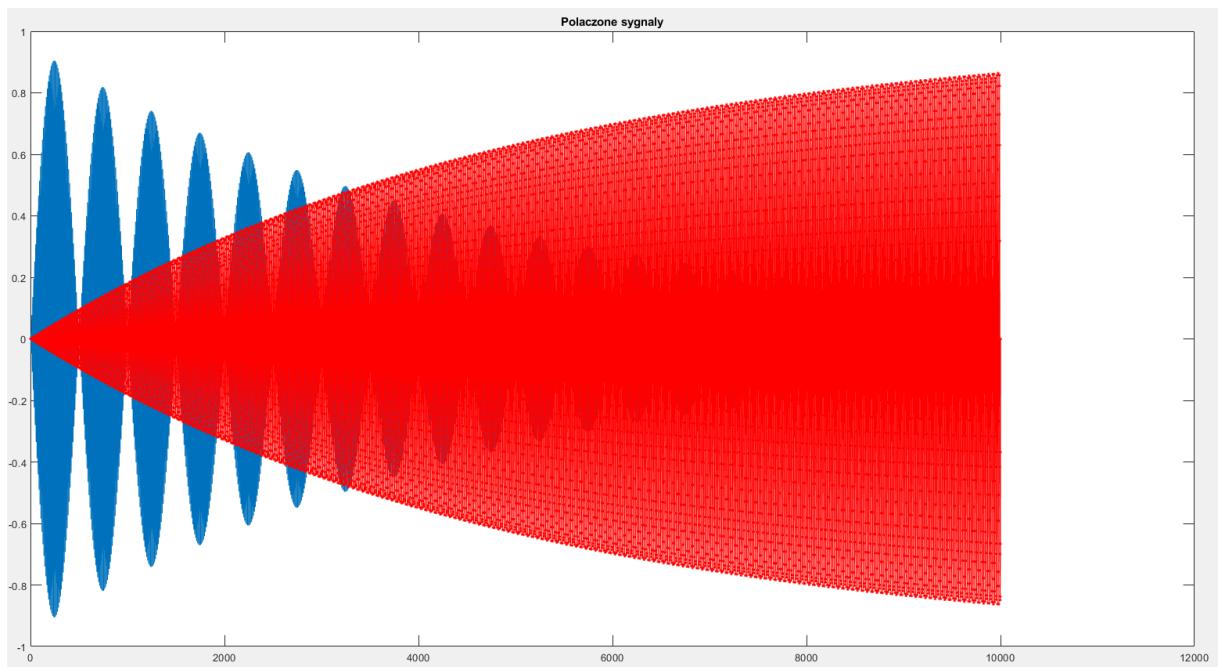
Rys. 3.1. Przykładowy przebieg sygnału sinusoidalnego z tłumieniem wykładniczym.



Rys. 3.2. Przykładowe sygnały. Sinusoidalny: zwykły, z modulacją amplitudy, z modulacją i tłumieniem, z innym tłumieniem.



Rys. 3.3. Połącznie dwóch sygnałów o różnym tłumieniu.



Rys. 3.4. Połączenie sygnału modulowanego z tłumieniem z sygnałem sinusoidalnym narastającym.

4. Laboratorium 4

W ramach tego ćwiczenia stworzone zostały generatory sygnałów podstawowych o określonych parametrach. Każdy sygnał posiada stworzoną funkcję. Lista sygnałów, funkcji i parametrów:

- Fala prostokątna dwuimienna (bipolarna) – $x=sbp(w, A, t, n)$,
- Fala prostokątna jednoimienna (unipolarna) o wypełnieniu $\frac{1}{2}$ – $x=sup_1_2(w, A, t, n)$,
- Fala jak powyżej ale o dowolnym wypełnieniu – $x=sup_wyp(f, A, t, n, tau)$,

- Fala trójkątna dwuimienna (bipolarna) – $x=tbp(w, A, t, n)$,
- Fala piłokształtna bipolarna – $x=tbpp(w, A, t, n)$,
- Fala trójkątna unipolarna – $x=tup(w, A, t, n)$,
- Fala piłokształtna unipolarna – $x=tupp(w, A, t, n)$,
- Fala sinusoidalna, wyprostowana dwupołówkowa – $x=swd(w, A, t, n)$,
- Fala sinusoidalna, wyprostowana jednopołówkowa – $x=swj(w, A, t, n)$,

Gdzie:

w – częstotliwość [rad./s], A – amplituda [-], t - wektor czasu [s], n – rząd ciągu [-], f – częstotliwość [Hz], tau – wypełnienie [-] (od 0 do 1).

Funkcje generujące zamieszczone są na końcu kodu. Każdą z funkcji można zapisać jako osobny plik, aby móc je wykorzystać w innych programach, można także usunąć znaki komentarza, aby funkcje działały wewnątrz skryptu.

Tab. 4.1. Kod programu „Krupnik_Lab_4”.

```
% Lab 4. Tworzenie generatorow sygnalow podstawowych
% Wywolanie nastepuje blokowo.
% Do dzialania wymagane sa deklaracje funkcji:
% sbp, sup_1_2, swd, swj, tbp, tbpp, tup, tupp
% Deklaracje sa zakomentowane na koncu pliku gdyby
% m pliki sie zgubily.
%
% Mateusz Krupnik
clc; clear all; close all;
% Parametry
A=1; % amplituda
f=1; % czestotliwosc
fs=1000; % czest. probkowania
t=0:(1/fs):10; % wektor czasu
n=[7,30,99]; % wektor liczby probek
w=2*pi*f; % wektor czestosci

%% Sygnal prostkątny bipolarny
% Generowanie wykresu
y = sbp(w, A, t, n);

% Wykresy
figure(1)
sgtitle('Fala prostokątna bipolarna');
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:));
    title(['Fala dla: w=' num2str(w) ...
        ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

%% Sygnal prostaktny unipolarny wypełnienie 1/2
% Generowanie wykresu
y = sup_1_2(w, A, t, n);

% Wykresy
figure(2)
sgtitle('Fala prostokątna unipolarna wypelnienie 1/2');
for i=1:length(n)
```

```

    subplot(length(n),1,i)
    plot(t, y(i,:));
    title(['Fala dla: w=' num2str(w) ...
        ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

%% Sygnal prostaktny unipolarny o dowolonym wypełnieniu
tau = 0.2; % okres, wypełnienie
% Generowanie wykresu
y = sup_wyp(f, A, t, n, tau);

% Wykresy
figure(3)
sgtitle(['Fala prostokątna unipolarna wypełnienie ' num2str(tau)]);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:));
    title(['Fala dla: w=' num2str(w) ...
        ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

%% Sygnal trojkatny bipolarny
% Generowanie wykresu
y = tbp(w, A, t, n);

% Wykresy
figure(4)
sgtitle(['Fala trojkątna bipolarna']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:));
    title(['Fala dla: w=' num2str(w) ...
        ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

%% Sygnal trojkatny bipolarny pilokształtny
% Generowanie wykresu
y = tbpp(w, A, t, n);

% Wykresy
figure(5)
sgtitle(['Fala trojkątna bipolarna piłopkształtna']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:));
    title(['Fala dla: w=' num2str(w) ...
        ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

%% Sygnal trojkatny unipolarny
% Generowanie wykresu
y = tup(w, A, t, n);

% Wykresy
figure(6)
sgtitle(['Fala trojkątna unipolarna']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:));

```

```

    title(['Fala dla: w=' num2str(w) ...
           ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

%% Sygnal trojkatny unipolarna pilokształtna
% Generowanie wykresu
y = tupp(w, A, t, n);

% Wykresy
figure(7)
sgtitle(['Fala trojkatna unipolarna pilokształtna']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:));
    title(['Fala dla: w=' num2str(w) ...
           ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

%% Sygnal sinusoidalny wyprostowany dwupołówkowy
% Generowanie wykresu
y = swd(w, A, t, n);

% Wykresy
figure(8)
sgtitle(['Fala sinusoidalna wyprostowana dwupołówkowa']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:));
    title(['Fala dla: w=' num2str(w) ...
           ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

%% Sygnal sinusoidalny wyprostowany jednopołówkowy
% Generowanie wykresu
y = swj(w, A, t, n);

% Wykresy
figure(9)
sgtitle(['Fala sinusoidalna wyprostowana jednopołówkowa']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:));
    title(['Fala dla: w=' num2str(w) ...
           ' n=' num2str(n(i)) ' A=' num2str(A)]);
end

% %%%%%% DEFINICJE FUNKCJI %%%%%%
% Definicja funkcji 1
function x = sbp(w, A, t, n)
    % Funkcja generująca fale prostokątna bipolarna
    % w - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
    x=zeros(length(n), length(t));
    for i=1:length(n)
        for j=1:2:2*n(i)
            x(i,:) = x(i,:) + ((1/j)*sin(j*w*t));
        end
    end

```

```

    end
    x = x*4*A/pi;
end

% Definicja funkcji 2
function x = sup_1_2(w, A, t, n)
    % Funkcja generująca fale prostokątna unipolarna o wypełnieniu 1/2
    % w - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
x=zeros(length(n), length(t));
for i=1:length(n)
    for j=1:4:2*n(i)
        x(i,:)=x(i,:)+((1/j)*cos(j*w*t));
    end
    for j=3:4:2*n(i)
        x(i,:)=x(i,:)-((1/j)*cos(j*w*t));
    end
end
x = x*2*A/pi + A/2;
end

% Definicja funkcji 3
function x = sup_wyp(f, A, t, n, tau)
    % Funkcja generująca fale prostk. unipolarna o dowolnym wypełnieniu
    % f - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
    % tau - czas trwania impulsu
x=zeros(length(n), length(t)); T = 1/f;
for i=1:length(n)
    for j=1:n(i)
        x(i,:)=x(i,:)+...
            sin(pi*j*tau/T)*cos(2*j*pi*f*t)/(pi*j*tau/T);
    end
end
x = A*tau/T + 2*A*tau*x/T;
end

% Definicja funkcji 4
function x = tbp(w, A, t, n)
    % Funkcja generująca fale trojkątna bipolarna
    % w - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
x=zeros(length(n), length(t));
for i=1:length(n)
    for j=1:4:n(i)
        x(i,:)=x(i,:)+((1/j^2)*sin(j*w*t));
    end
    for j=3:4:n(i)
        x(i,:)=x(i,:)-((1/j^2)*sin(j*w*t));
    end
end
x = x*8*A/(pi^2);
end

% Definicja funkcji 5
function x = tbpp(w, A, t, n)
    % Funkcja generująca fale trojkątna bipolarna pilokształtna
    % w - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
x=zeros(length(n), length(t));
for i=1:length(n)

```

```

for j=1:2:n(i)
    x(i,:) = x(i,:) + ((1/j)*sin(j*w*t));
end
for j=2:2:n(i)
    x(i,:) = x(i,:) - ((1/j)*sin(j*w*t));
end
end
x = x*2*A/pi;
end

% Definicja funkcji 6
function x = tup(w, A, t, n)
    % Funkcja generująca fale trojkątna unipolarna
    % w - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
    x=zeros(length(n), length(t));
    for i=1:length(n)
        for j=0:n(i)
            x(i,:) = x(i,:) + ((1/((2*j+1)^2))*cos((2*j+1)*w*t));
        end
    end
    x = x*(-4*A) / (pi^2) + A/2;
end

% Definicja funkcji 7
function x = tupp(w, A, t, n)
    % Funkcja generująca fale trojkątna unipolarna pilokształtna
    % w - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
    x=zeros(length(n), length(t));
    for i=1:length(n)
        for j=1:n(i)
            x(i,:) = x(i,:) - ((1/j)*sin(j*w*t));
        end
    end
    x = x*A/pi + A/2;
end

% Definicja funkcji 8
function x = swd(w, A, t, n)
    % Funkcja generująca fale sinusoidalne wyprostowane dwupołowkowe
    % w - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
    x=zeros(length(n), length(t));
    for i=1:length(n)
        for j=1:n(i)
            x(i,:) = x(i,:) + (1/(4*j^2-1))*cos(2*j*w*t);
        end
    end
    x = x*(-4)*A/pi + 2*A/pi;
end

% Definicja funkcji 9
function x = swj(w, A, t, n)
    % Funkcja generująca fale sinusoidalne wyprostowane jednopółowkowe
    % w - częstotliwość, A - amplituda
    % t - wektor czasu, n - rzad ciągu
    x=zeros(length(n), length(t));
    for i=1:length(n)
        for j=1:n(i)

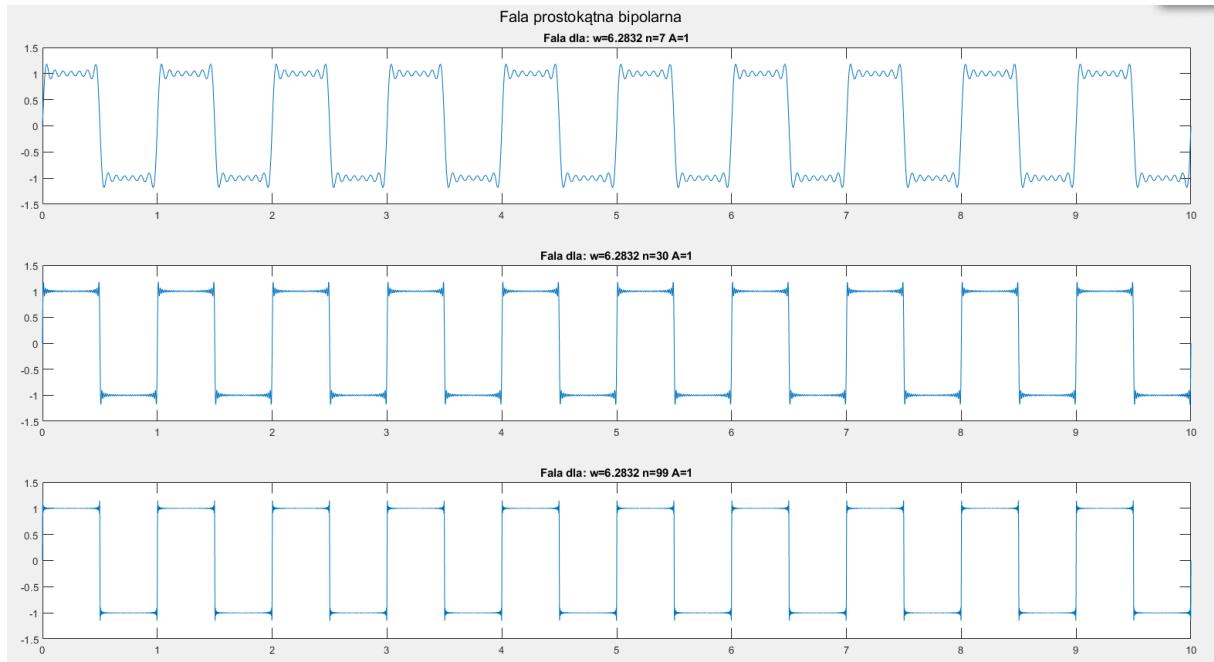
```

```

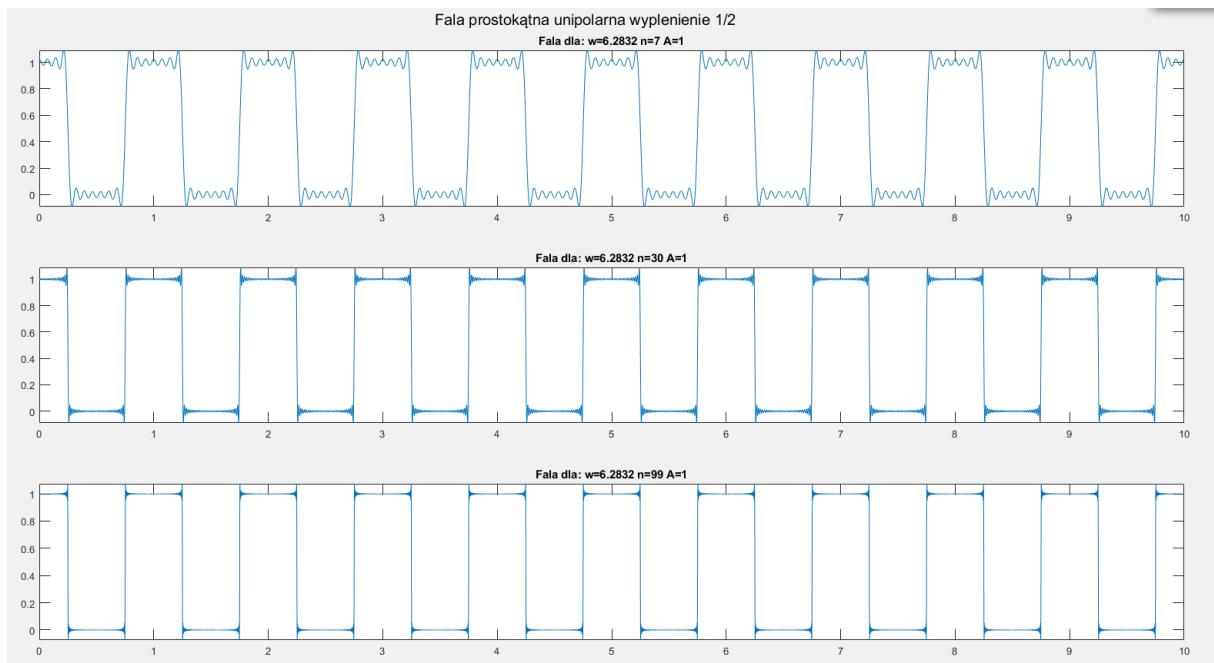
x(i,:) = x(i,:)+ (1/(4*j^2-1))*cos(2*j*w*t);
end
x(i,:) = x(i,:)*A*pi + A/pi + sin(w*t)*A/2;
end

```

W tabeli 5.2 przedstawione są wzory szeregow aproksymujących opisane funkcje. Dodatkowo przedstawione zostaną wykresy funkcji dla różnych rzędów ciągów aproksymujących.



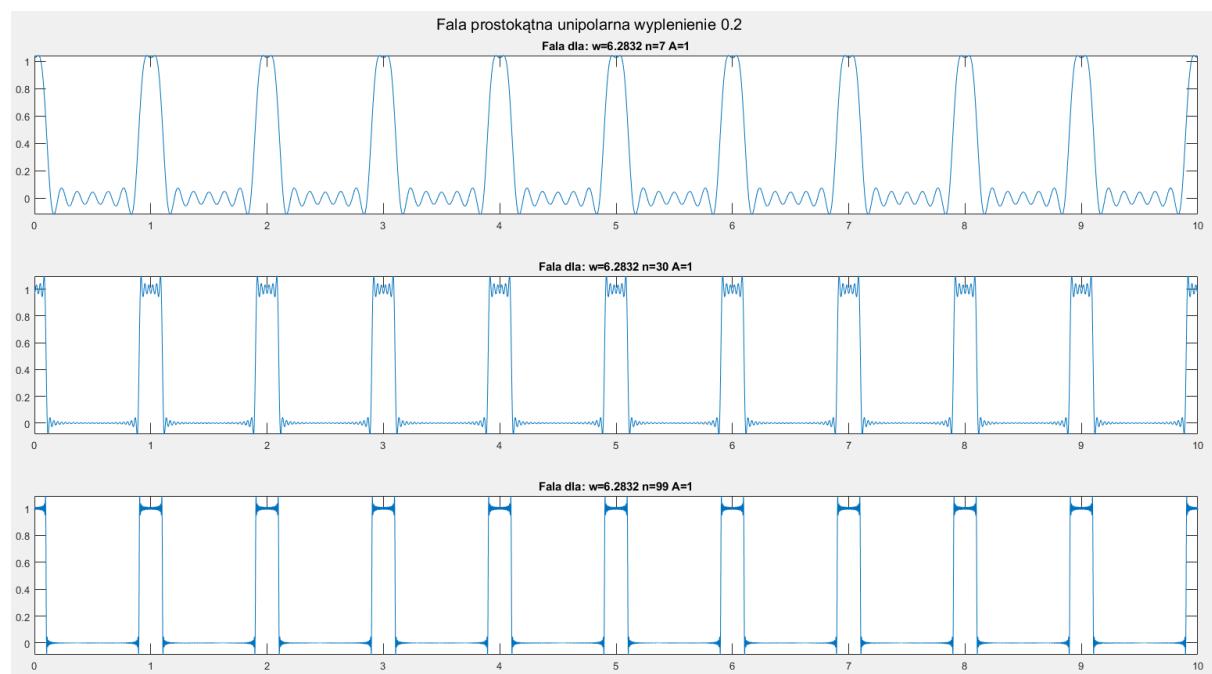
Rysunek 4.1. Fala prostokątna bipolarna.



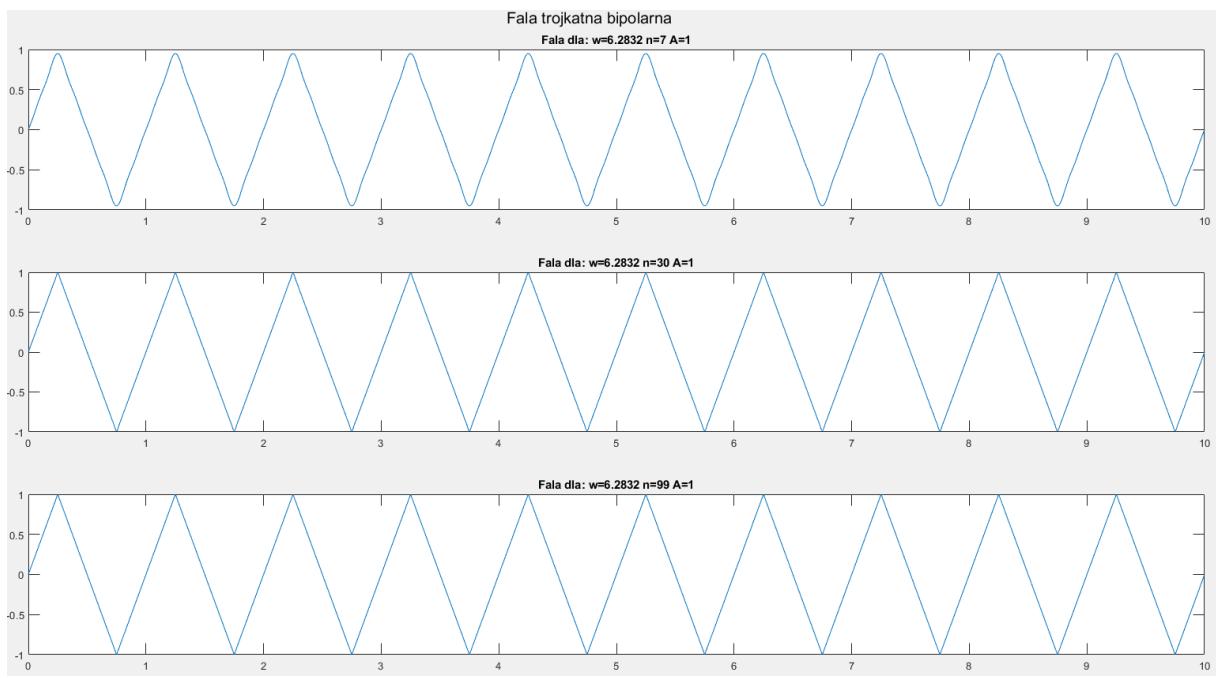
Rysunek 4.2. Fala unipolarna o wypełnieniu 1/2.

Tab. 4.2. Wzory sygnałów używanych w programie.

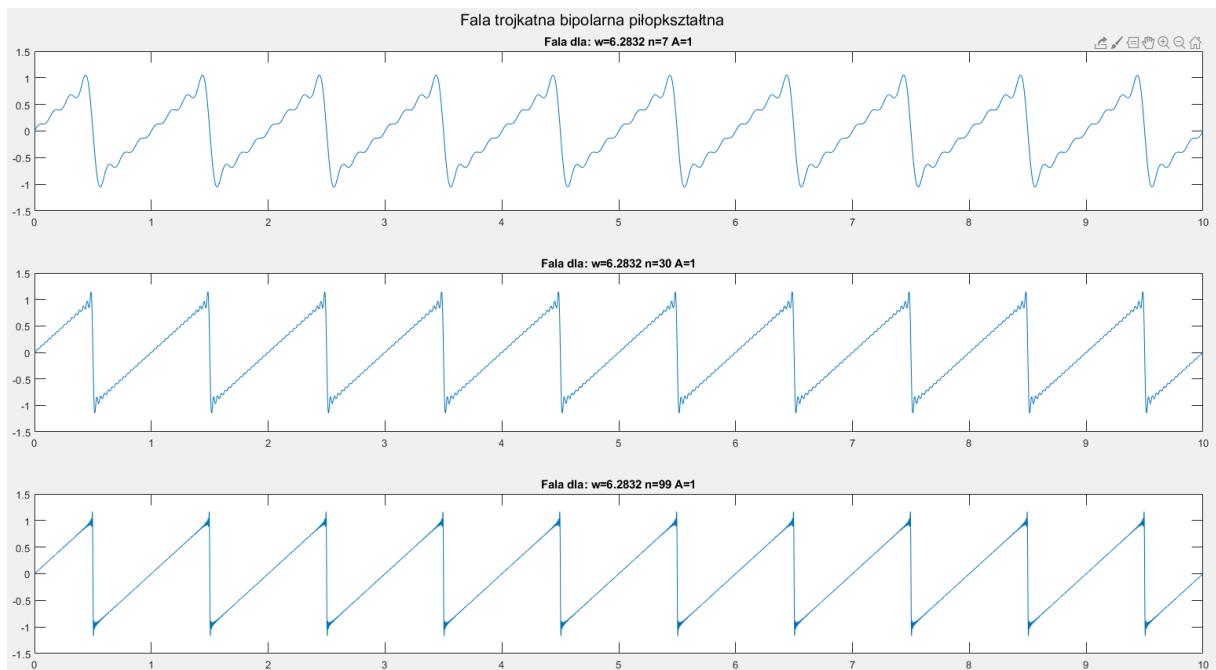
Sygnal	Współczynniki szeregu Fouriera
Prostokątny, bipolarny rys. 3.1a	$x(t) = \frac{4A}{\pi} \left(\sin \omega_0 t + \frac{1}{3} \sin 3\omega_0 t + \frac{1}{5} \sin 5\omega_0 t + \dots \right)$
Prostokątny, unipolarny wypełnienie 1/2 rys. 3.1b	$x(t) = \frac{A}{2} + \frac{2A}{\pi} \left(\cos \omega_0 t - \frac{1}{3} \cos 3\omega_0 t + \frac{1}{5} \cos 5\omega_0 t - \dots \right)$
Prostokątny, unipolarny, wypełnienie dowolne rys. 3.1c	$x(t) = \frac{A\tau}{T} + \frac{2A\tau}{T} \sum_{k=1}^{\infty} \frac{\sin(\pi k\tau/T)}{\pi k\tau/T} \cos k\omega_0 t$
Trójkątny, bipolarny 1 rys. 3.1d	$x(t) = \frac{8A}{\pi^2} \left(\sin \omega_0 t - \frac{1}{3^2} \sin 3\omega_0 t + \frac{1}{5^2} \sin 5\omega_0 t - \dots \right)$
Trójkątny, bipolarny 2 rys. 3.1e	$x(t) = \frac{2A}{\pi} \left(\sin \omega_0 t - \frac{1}{2} \sin 2\omega_0 t + \frac{1}{3} \sin 3\omega_0 t - \dots \right)$
Trójkątny, unipolarny 1 rys. 3.1f	$x(t) = \frac{A}{2} - \frac{4A}{\pi^2} \sum_{k=0}^{\infty} \frac{1}{(2k+1)^2} \cos(2k+1)\omega_0 t$
Trójkątny, unipolarny 2 rys. 3.1g	$x(t) = \frac{A}{2} - \frac{A}{\pi} \sum_{k=1}^{\infty} \frac{\sin k\omega_0 t}{k}$
Sinusoidalny wyprostowany dwupołówkowo rys. 3.1h	$x(t) = \frac{2A}{\pi} - \frac{4A}{\pi} \sum_{k=1}^{\infty} \frac{1}{4k^2 - 1} \cos 2k\omega_0 t$
Sinusoidalny wyprostowany jednopołówkowo rys. 3.1i	$x(t) = \frac{A}{\pi} + \frac{A}{2} \sin \omega_0 t - \frac{2A}{\pi} \sum_{k=1}^{\infty} \frac{1}{4k^2 - 1} \cos 2k\omega_0 t$



Rysunek 4.3. Fala unipolarna o wypełnieniu dowolnym.

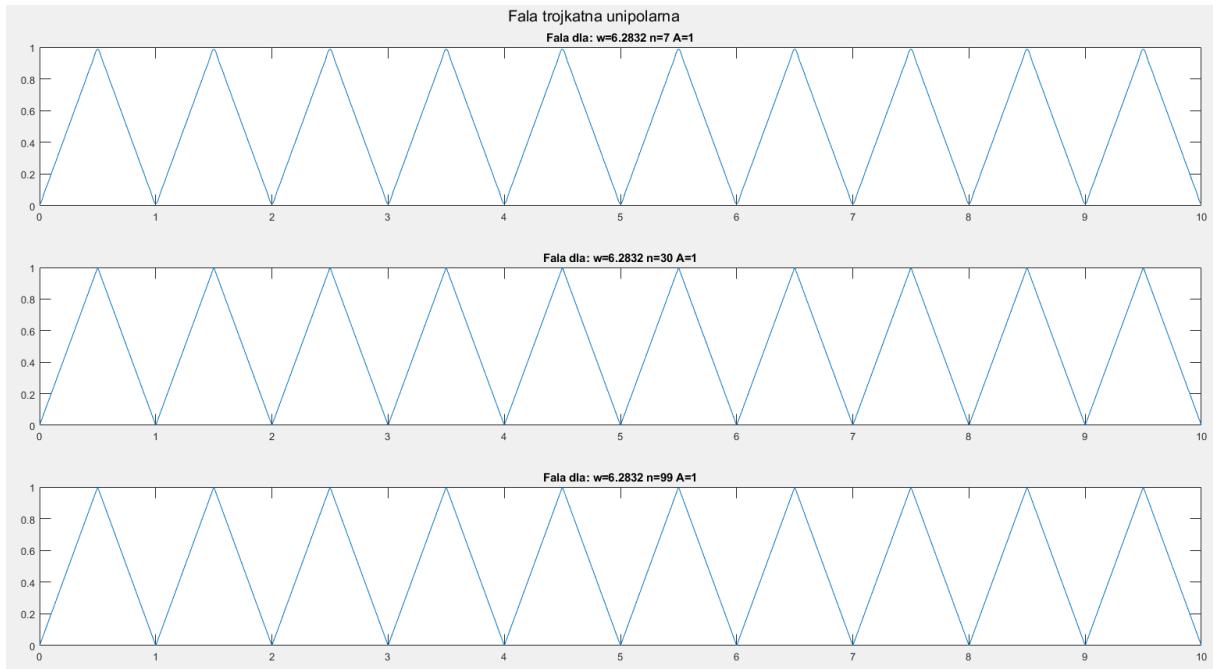


Rysunek 4.4. Fala trójkątna bipolarna.

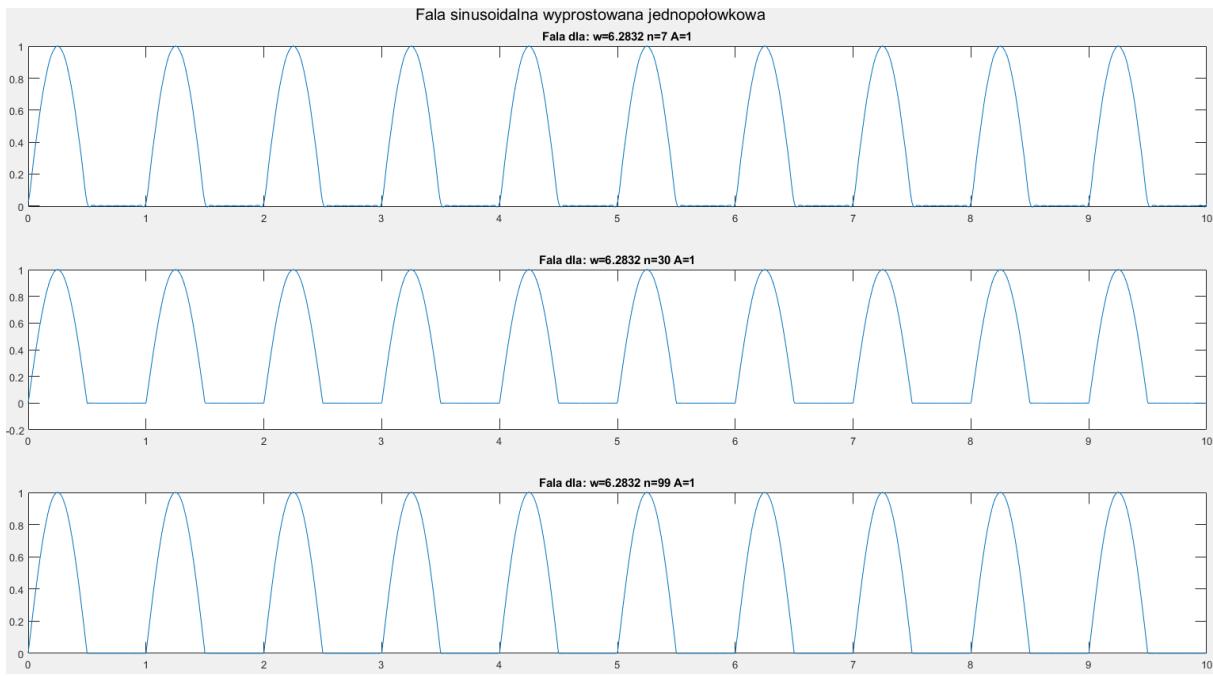


Rysunek 4.5. Fala piłokształtna bipolarna.

Jak można zauważyć szereg aproksymujący sygnał prostokątny potrzebuje o wiele większy rząd, aby dokładniej odwzorowywać sygnał idealny niż w przypadku sygnału trójkątnego. Fala piłokształtna potrzebuje mniejszy rząd niż fale prostokątne, ale uwidacznia się w niej efekt z dużych oscylacji przy nagłych zmianach wartości funkcji.

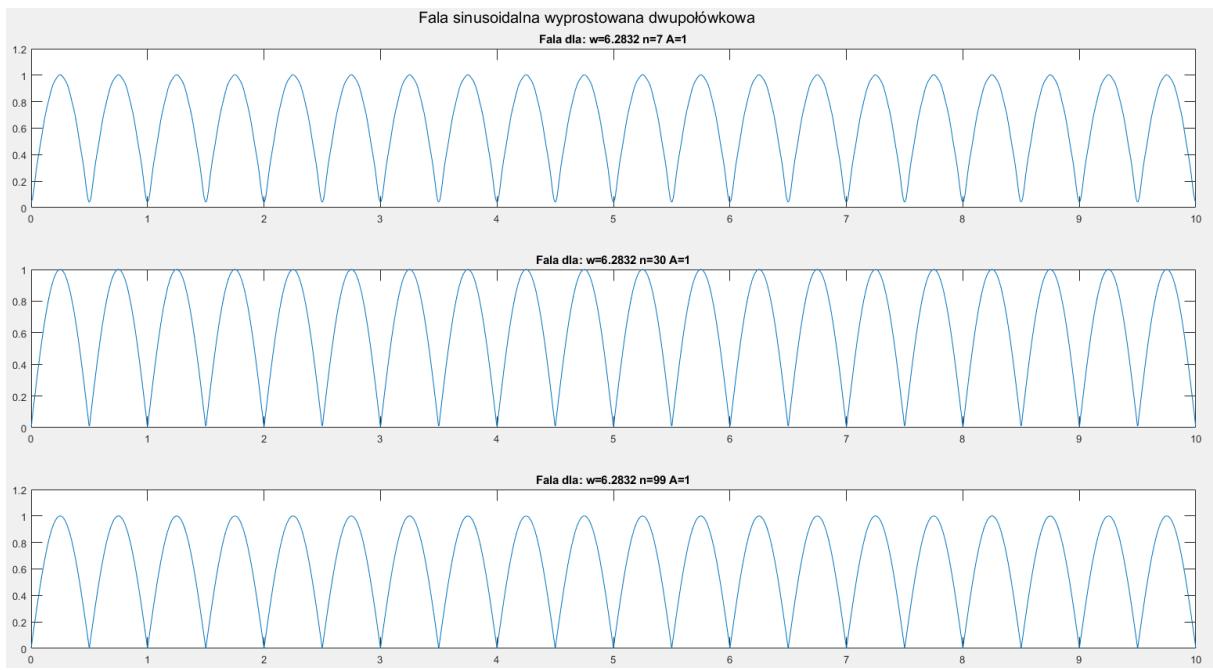


Rysunek 4.6. Fala trójkątna unipolarna.

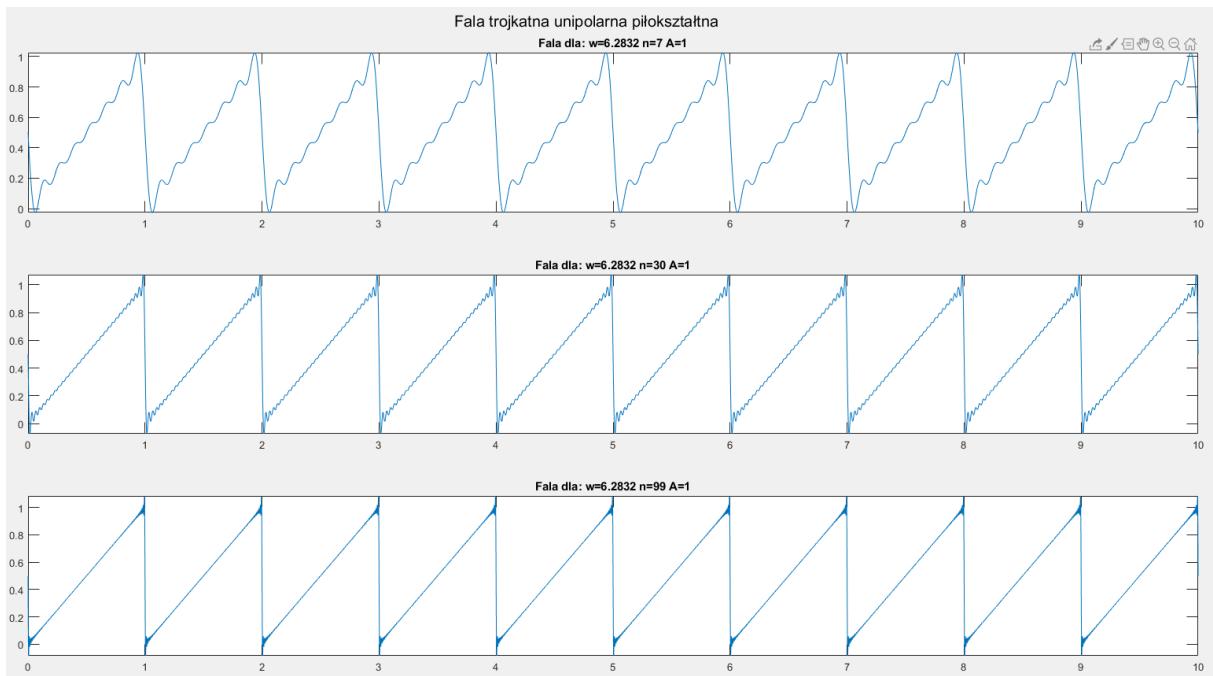


Rysunek 4.7. Fala sinusoidalna jednopołówkowa, wyprostowana.

Sygnały sinusoidalne i ich pochodne wykazują się bardzo dobrą aproksymacją już dla niewielkiego rzędu ciągu.



Rysunek 4.8. Fala sinusoidalna dwupołówkowa, wyprostowana.



Rysunek 4.9. Fala piłokształtna unipolarna.

5. Laboratorium 5

W ramach tego ćwiczenia zrealizowana została analiza częstotliwościowa fal obliczony i pokazanych w poprzednim rozdziale. Dodatkowo dokonana zostanie analiza układów RLC i RC. Stworzona została funkcja $[f, M, W]=\text{fft_from_signal}(x, fs)$. Funkcja ta znajduje się na końcu pliku oraz w osobnym pliku tak aby możliwe było jej wykorzystanie w pozostałych programach. Parametry wejściowe do funkcji to sygnał lub sygnały będące kolejnymi wierszami macierzy oraz częstotliwość próbkowania. W

wyniku otrzymuje się wektor częstotliwości, widmową moc sygnałów (kolejne wiersze) oraz widmo sygnału (kolejne wiersze).

Tab. 5.1. Kod programu „Krupnik_Lab_5_1”.

```
% Lab 5. Procedura szybkiej transformacji Fouriera
% Definicja funkcji obliczającej FFT z sygnałów
% znajduje się na koncu pliku.
%                                         Mateusz Krupnik

% Generowanie sygnału sinusoidalnego oraz jego widmo
clc; clear all; close all;
f1=100;
fs=1000;
t=0:(1/fs):1.3;
Ts=1/fs;
A=1;
y=A*sin(2*pi*f1*t);
N=1024;
fft_moc=fft(y(1:N));
moc_wid=fft_moc.*conj(fft_moc)/N;
f=fs*(0:N/2-1)/N;
figure(1)
plot(f,moc_wid(1:N/2))

%% Analiza sygnałów z Lab_4
% Dane podstawowe
clc; clear all; close all;
A=1; f=5; fs=1000; w=2*pi*f;
t=0:(1/fs):1;
n=[7,30,99];
j=1; % numer wykresu
%% Sygnał prostokątny bipolarny
% Generowanie wykresu
y = sbp(w, A, t, n);

% Wykresy
figure(1);
sgtitle('Fala prostokątna bipolararna');
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i)) ...
    ' A=' num2str(A)]);
    xlabel('Czas [s]');
end

% Obliczenie FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(2);
sgtitle('Analiza widmowa fali prostokątnej bipolarnej');
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n=' ...
    num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Częstotliwość [Hz]'); ylabel('Moc widmowa');
    subplot(2,length(n),3+i)
    plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n=' ...
    num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Częstotliwość [Hz]'); ylabel('Widmo');
```

```

end

%% Sygnał prostokątny unipolarny o wypełnieniu 1/2
% Generowanie wykresu
y = sup_1_2(w, A, t, n);

% Wykresy
figure(3);
sgtitle('Fala prostokątna unipolarna o wyp. 1/2');
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i)) ...
    ' A=' num2str(A)]);
    xlabel('Czas [s]');
end

% Obliczenie FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(4);
sgtitle('Analiza widmowa fali prostokątnej unipolarnej wyp. 1/2');
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n=' ...
    num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Moc widmowa');
    subplot(2,length(n),3+i)
    plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n=' ...
    num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Widmo');
end

%% Sygnał prostokątny unipolarny o wypełnieniu dowolnym
% Generowanie wykresu
tau = 0.75; % okres, wypełnienie w procentach
if tau <=1 && tau >=0
    tau = tau*1/f; % w sekundach
else
    tau = 0.3
end
y = sup_wyp(f, A, t, n, tau);

% Wykresy
figure(5);
sgtitle(['Fala prostokątna unipolarna wypłelenie ' num2str(tau)]);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i)) ...
    ' A=' num2str(A)]);
    xlabel('Czas [s]');
end

% Obliczenie FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(6);
sgtitle(['Analiza widmowa fali prostokątnej unipolarnej o wypłeleniu ' ...
num2str(tau)]);
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n=' ...
    num2str(n(i)) ' A=' num2str(A)]);
end

```

```

xlabel('Czestotliwosc [Hz]'); ylabel('Moc widmowa');
subplot(2,length(n),3+i)
plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n='
num2str(n(i)) ' A=' num2str(A)]);
xlabel('Czestotliwosc [Hz]'); ylabel('Widmo');
end

%% Sygнал троекатный bipolarny
% Генерирование графика
y = tbp(w, A, t, n);

% Графики
figure(7)
sgtitle(['Fala trojkatna bipolararna']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i))
' A=' num2str(A)]);
    xlabel('Czas [s]');
end

% Вычисление FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(8);
sgtitle(['Analiza widmowa fali trojkatnej bipolarnej']);
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n='
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Moc widmowa');
    subplot(2,length(n),3+i)
    plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n='
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Widmo');
end

%% Sygнал троекатный bipolarny pilokształtny
% Генерирование графика
y = tbpp(w, A, t, n);

% Графики
figure(9)
sgtitle(['Fala trojkatna bipolararna pilopksztaltna']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i))
' A=' num2str(A)]);
end

% Вычисление FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(10);
sgtitle(['Analiza widmowa fali trojkatnej bipolarnej pilopksztaltnej']);
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n='
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Moc widmowa');
    subplot(2,length(n),3+i)
    plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n='

```

```

num2str(n(i)) ' A=' num2str(A));
    xlabel('Czestotliwosc [Hz]'); ylabel('Widmo');
end

%% Sygnal trojkatny unipolarny
% Generowanie wykresu
y = tup(w, A, t, n);

% Wykresy
figure(11)
sgtitle(['Fala trojkatna unipolarna']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i))
' A=' num2str(A)]);
end

% Obliczenie FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(12);
sgtitle(['Analiza widmowa fali trojkatnej unipolarnej']);
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n='
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Moc widmowa');
    subplot(2,length(n),3+i)
    plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n='
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Widmo');
end

%% Sygnal trojkatny unipolarna pilokształtna
% Generowanie wykresu
y = tupp(w, A, t, n);

% Wykresy
figure(13)
sgtitle(['Fala trojkatna unipolarna pilokształtna']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i))
' A=' num2str(A)]);
end

% Obliczenie FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(14);
sgtitle(['Analiza widmowa fali trojkatnej pilokształtnej']);
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n='
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Moc widmowa');
    subplot(2,length(n),3+i)
    plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n='
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Widmo');
end

```

```

%% Sygnal sinusoidalny wyprostowany dwupołówkowy
% Generowanie wykresu
y = swd(w, A, t, n);

% Wykresy
figure(15)
sgtitle(['Fala sinusoidalna wyprostowana dwupołówkowa']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i)) ...
' A=' num2str(A)]);
end

% Obliczenie FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(16);
sgtitle(['Analiza widmowa fali sinusoidalnej wyprostowanej ...
dwupołówkowej']);
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n=' ...
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Moc widmowa');
    subplot(2,length(n),3+i)
    plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n=' ...
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Widmo');
end

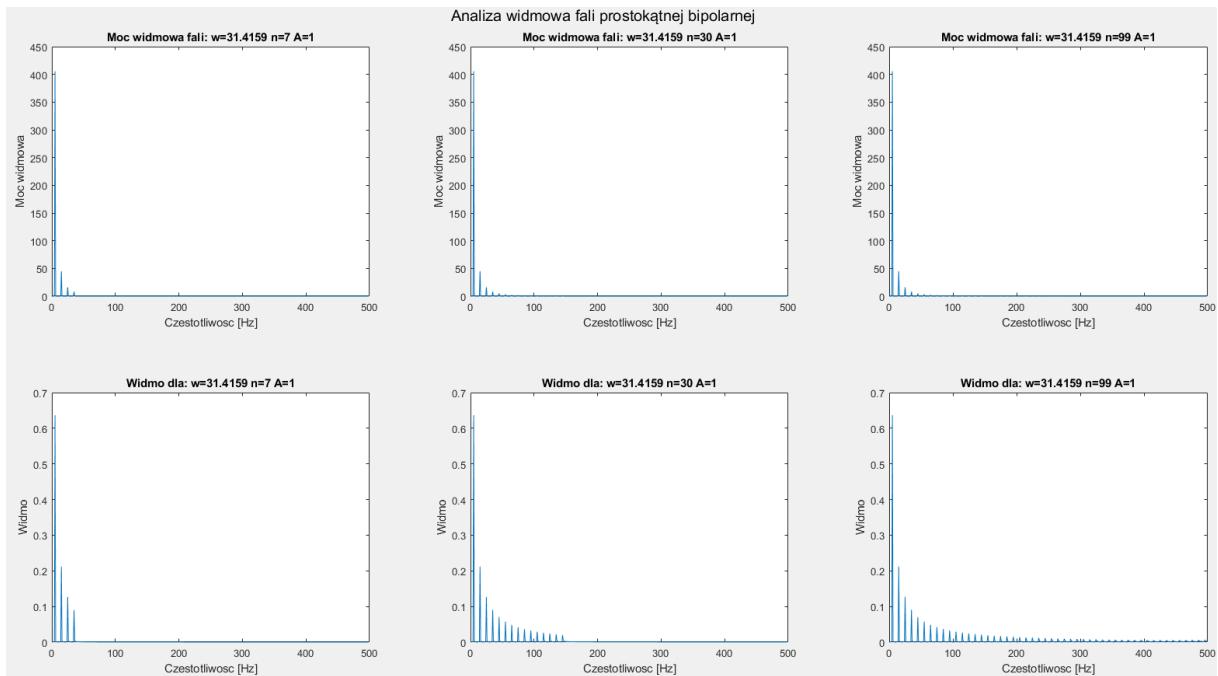
%% Sygnal sinusoidalny wyprostowany jednopołówkowy
% Generowanie wykresu
y = swj(w, A, t, n);

% Wykresy
figure(17)
sgtitle(['Fala sinusoidalna wyprostowana jednopołówkowa']);
for i=1:length(n)
    subplot(length(n),1,i)
    plot(t, y(i,:)); title(['Fala dla: w=' num2str(w) ' n=' num2str(n(i)) ...
' A=' num2str(A)]);
end

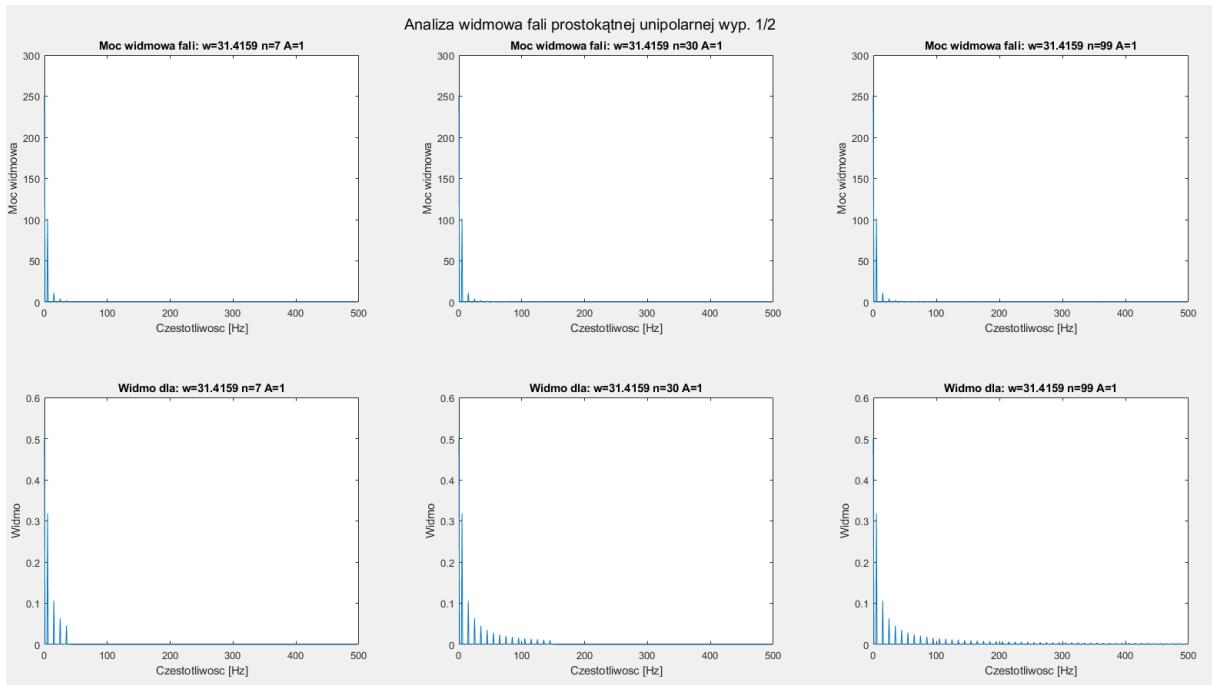
% Obliczenie FFT
[f_w, M, W] = fft_from_signal(y, fs);
figure(18);
sgtitle(['Analiza widmowa fali sinusoidalnej wyprostowanej ...
jednopolowkowej']);
for i=1:length(n)
    subplot(2,length(n),i)
    plot(f_w, M(i, :)); title(['Moc widmowa fali: w=' num2str(w) ' n=' ...
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Moc widmowa');
    subplot(2,length(n),3+i)
    plot(f_w, W(i, :)); title(['Widmo dla: w=' num2str(w) ' n=' ...
num2str(n(i)) ' A=' num2str(A)]);
    xlabel('Czestotliwosc [Hz]'); ylabel('Widmo');
end

```

```
% DEFINICJA FUNKCI %%%%%%
function [f, M, W] = fft_from_signal(y, fs)
% fft_from_signal
% Summary of this function goes here
% Detailed explanation goes here
% y - signal matrix, with signals as rows
% f - frequency, M - power spectrum, W - spectrum
N = length(y);
for i=1:size(y, 1)
    fft_moc=fft(y(i, 1:N));
    moc_wid=fft_moc.*conj(fft_moc)/N;
    widmo=sqrt(fft_moc.*conj(fft_moc))/N;
    f=fs*(0:N/2-1)/N;
    M(i,:)=moc_wid;
    W(i,:)=widmo;
end
M = M(:, floor(1:N/2));
W = W(:, floor(1:N/2));
end
```

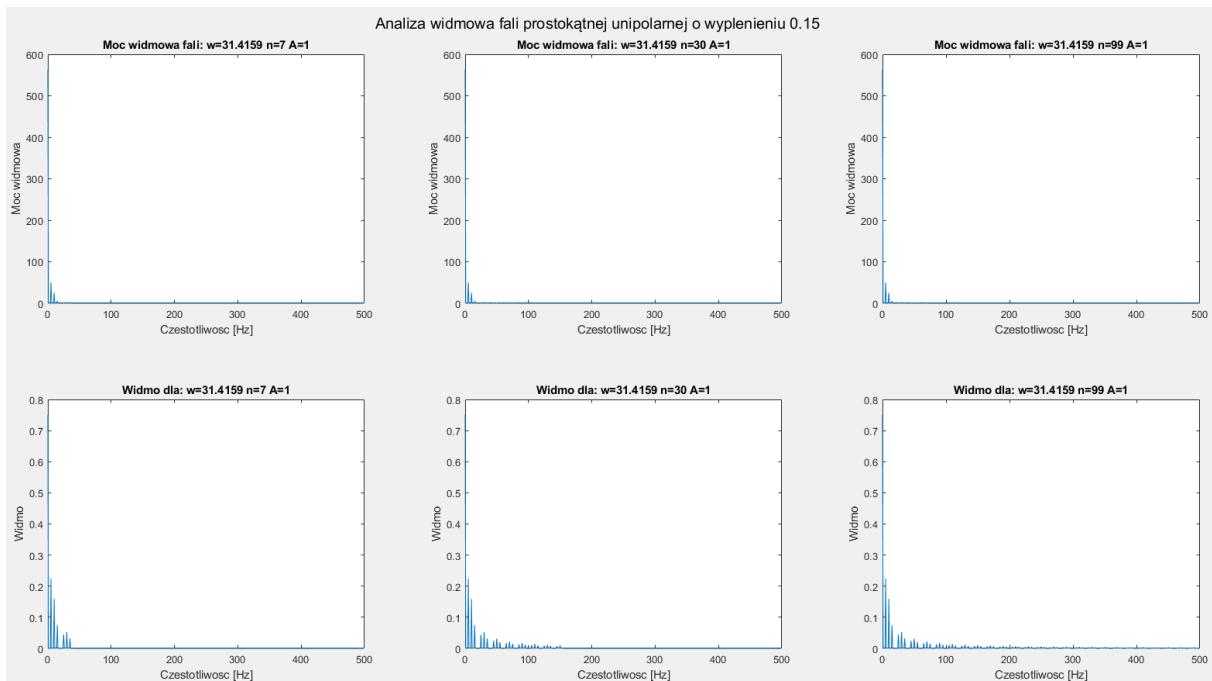


Rys. 5.1. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej bipolarnej. Każda kolumna odpowiada większemu rzędowi ciągu.

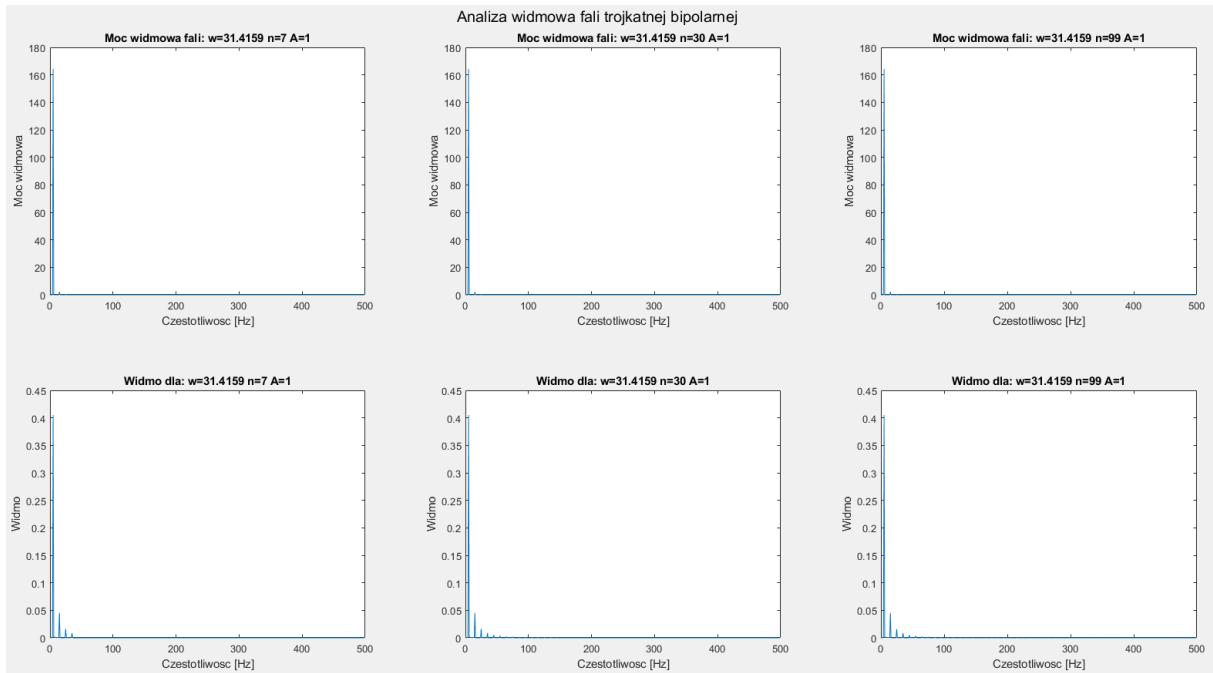


Rys. 5.2. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej unipolarnej. Każda kolumna odpowiada większemu rzędowi ciągu.

Widoczne jest ze wraz ze wzrostem rzędu ciągu pojawiają się kolejne składowe częstotliwościowe, ale o coraz mniejszych amplitudach. Dlatego pierwsze elementy ciągu posiadają największą moc widmową.

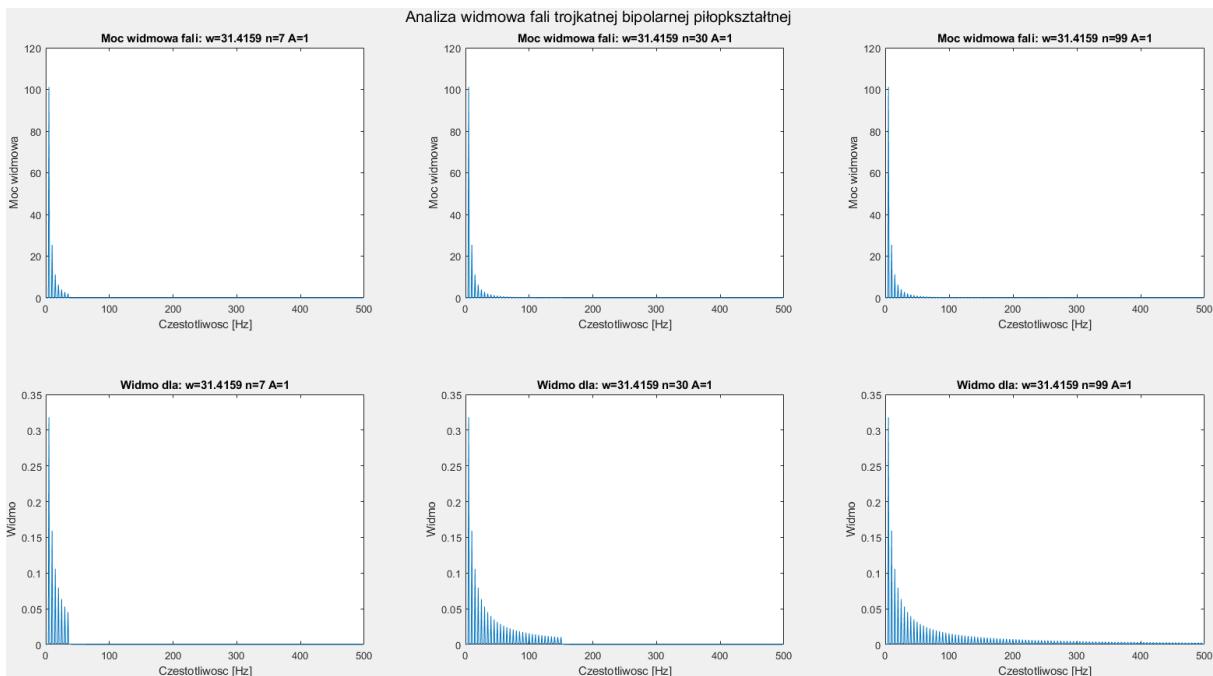


Rys. 5.3. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej unipolarnej o wypełnieniu 0.15. Każda kolumna odpowiada większemu rzędowi ciągu.

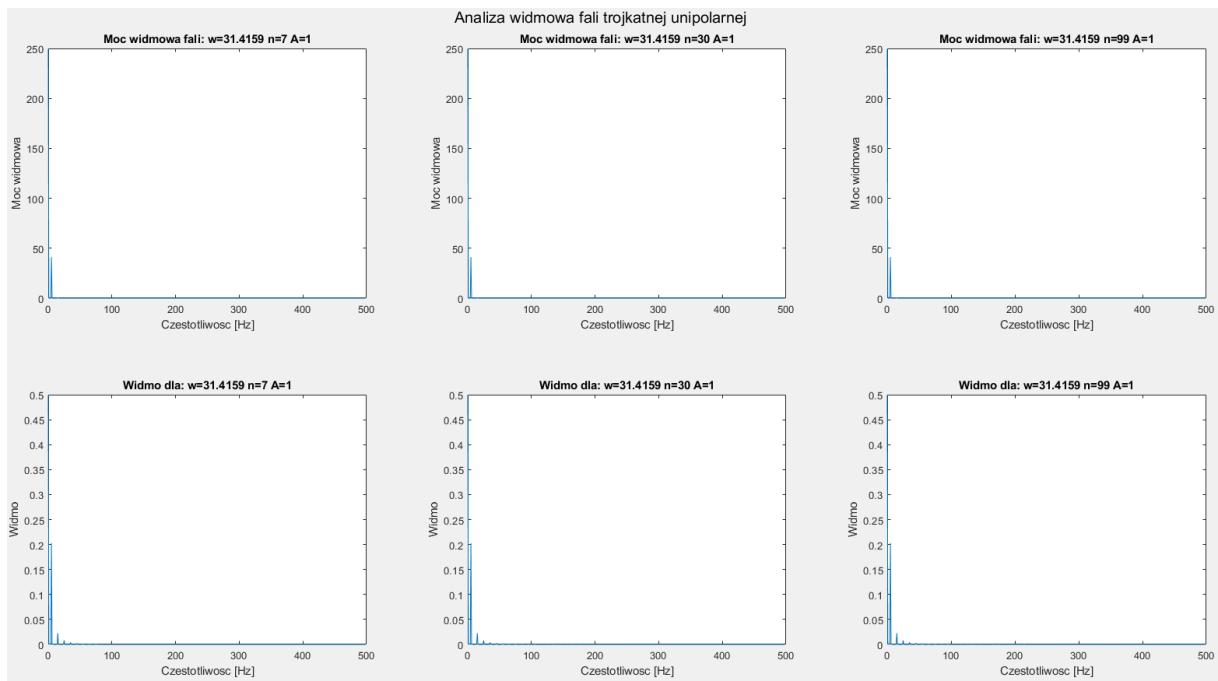


Rys. 5.4. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej bipolarnej. Każda kolumna odpowiada większemu rzędowi ciągu.

Na wykresie 6.4. można zauważyc ze dla fali trójkątnej już pierwsze 2 składowe odpowiadają za aproksymacje kształtu. Kolejne składowe stanowią kosmetykę załamań fali. W przypadku rysunku 6.3. widoczne jest, że kolejne prążki widma nie maleją wykładniczo jak w przypadku rysunku 6.2. Pojawiają się pewne oscylacje.

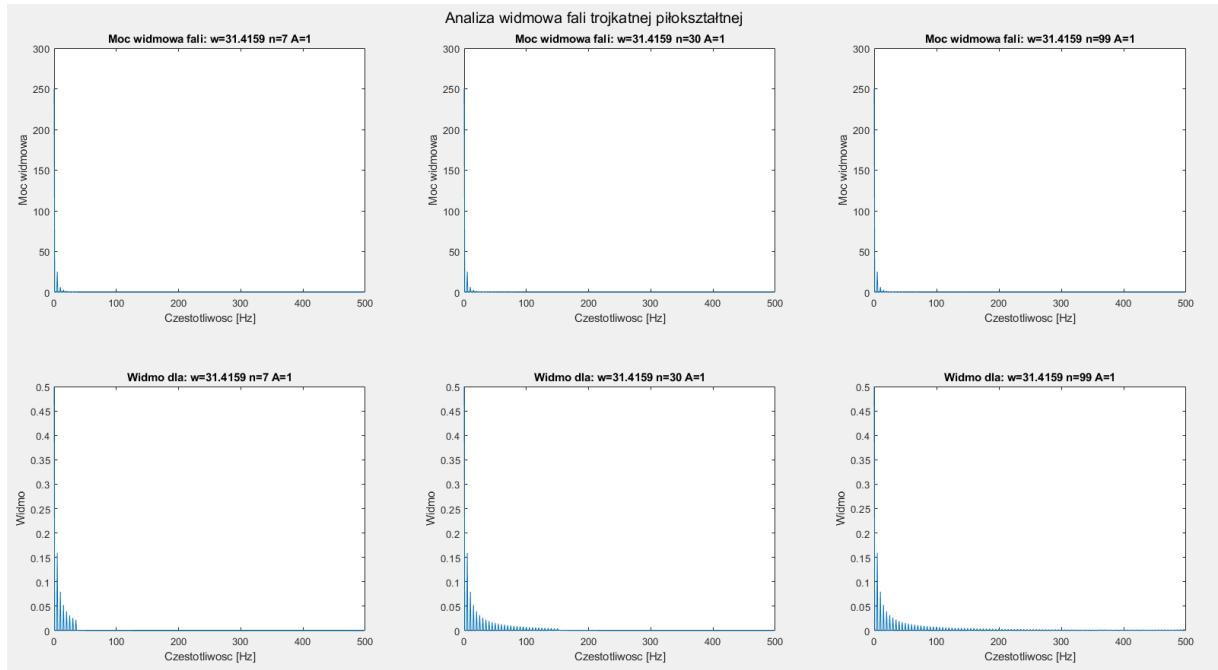


Rys. 5.5. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej bipolarnej pitokształtnej. Każda kolumna odpowiada większemu rzędowi ciągu.

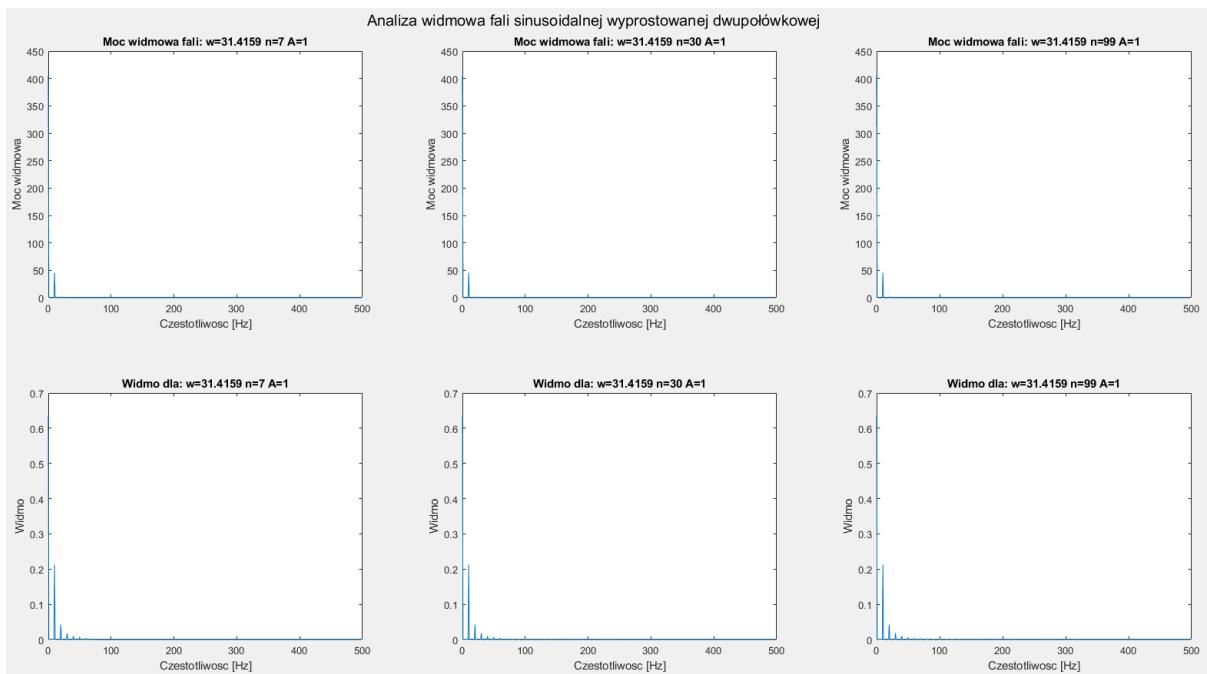


Rys. 5.6. Moc widmowa (1 rzad wykresów) i widmo (2 rzad wykresów) dla fali trójkątnej unipolarnie. Każda kolumna odpowiada większemu rzędowi ciągu.

Ponownie uwidacznia się fakt, iż fala trójkątna jest dobrze aproksymowana małym rzędem ciągu. Z kolei fala piłokształtna wymaga wielu składowych w celu odwzorowania nagłych załamań.

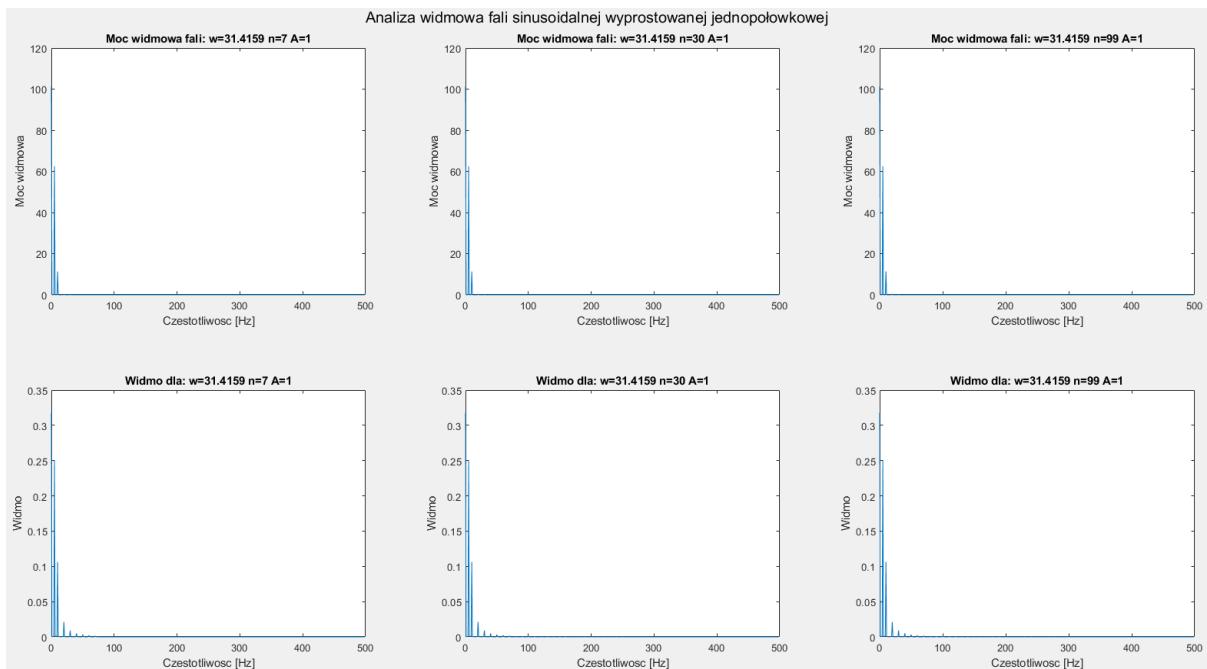


Rys. 5.7. Moc widmowa (1 rzad wykresów) i widmo (2 rzad wykresów) dla fali trójkątnej unipolarnie piłokształtnej. Każda kolumna odpowiada większemu rzędowi ciągu.



Rys. 5.8. Moc widmowa (1 rzad wykresów) i widmo (2 rzad wykresów) dla fali sinusoidalnej dwupołówkowej wyprostowanej. Każda kolumna odpowiada większemu rzędowi ciągu.

Sygnały sinusoidalnie pochodne aproksymowana są już w pierwszych elementach ciągu.



Rys. 5.9. Moc widmowa (1 rzad wykresów) i widmo (2 rzad wykresów) dla fali sinusoidalnej jednopółkowej wyprostowanej. Każda kolumna odpowiada większemu rzędowi ciągu.

Następnie po analizie sygnałów podstawowych przeprowadzona została analiza układów RC i RLC.

Tab. 5.2. Kod programu "Krupnik_Lab_5_2.m".

```
% Lab 5. Analiza układów RC i RLC
%
%% Analiza układów RC
```

Mateusz Krupnik

```
% Dane układu
R=1000; C=10^(-6);
L=[1]; M=[(R*C) 1]; % Licznik, Mianownik
sys=tf(L,M) % Transfer function (licznik, mianownik)
% Wykresy analizy układu
figure(1)
freqs(L,M) % analiza amplitudowo i fazowo - częstotliwościowa
figure(2)
impulse(L,M) % odpowiedź impulsowa układu
figure(3)
step(L,M) % odpowiedź skokowa układu
figure(4)
iopzplot(sys) % wykres zer i biegunów dla układów wej/ wyj
[z,p,k]=tf2zp(L,M) % konwersja Transfer Function na zera i bieguny

%% Analiza układu RLC
% Dane układu
R=1000; C=10^(-6); L=1;
L=[1]; M=[(L*C) (R*C) 1]; % licznik i mianownik
sys=tf(L,M) % Transfer function (licznik, mianownik)
% Wykresy analizy układu
figure(5)
freqs(L,M) % analiza amplitudowo i fazowo - częstotliwościowa
figure(6)
impulse(L,M) % odpowiedź impulsowa układu
figure(7)
step(L,M) % odpowiedź skokowa układu
figure(8)
iopzplot(sys) % wykres zer i biegunów dla układów wej/ wyj
[z,p,k]=tf2zp(L,M) % konwersja Transfer Function na zera i bieguny
```

Przyjęto parametry dla układu RC: $R=1000[\Omega]$, $C=10^{-6}[F]$ oraz dla układu RLC: $R=1000[\Omega]$, $C=10^{-6}[F]$, $L=1[H]$. Za pomocą tych wartości zdefiniowane zostały liczniki i mianowniki transmitancji układów danymi wzorami poniżej.

$$H(s) = \frac{1}{RC \cdot s + 1}, H(s) = \frac{1}{LC \cdot s^2 + RC \cdot s + 1}$$

Licznik i mianownik przyjmują wartości kolejny współczynników wielomiany zmiennej s , przy czym zaczyna się od potęgi zerowej. Funkcja tf od Transfer Function zwraca strukturę będącą opisem modelowanego systemu. Wynik polecenia wyświetlany jest w oknie poleceń.

```
sys =
1
-----
0.001 s + 1

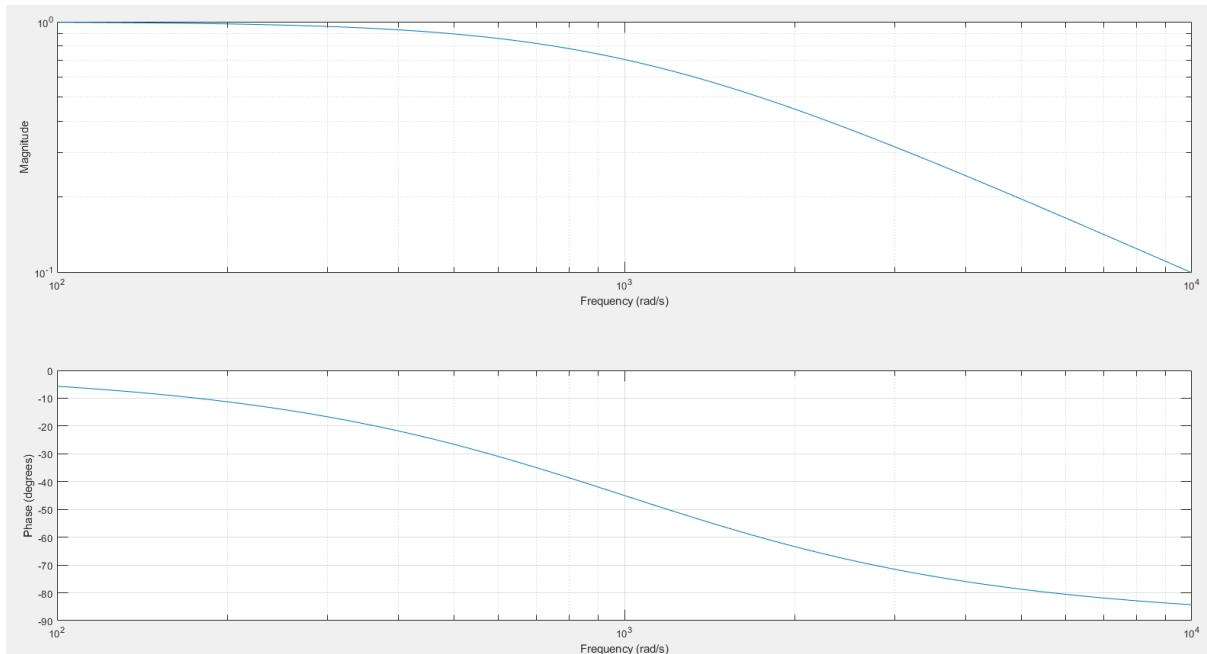
sys =
1
-----
1e-06 s^2 + 0.001 s + 1

Continuous-time transfer function. Continuous-time transfer function.
```

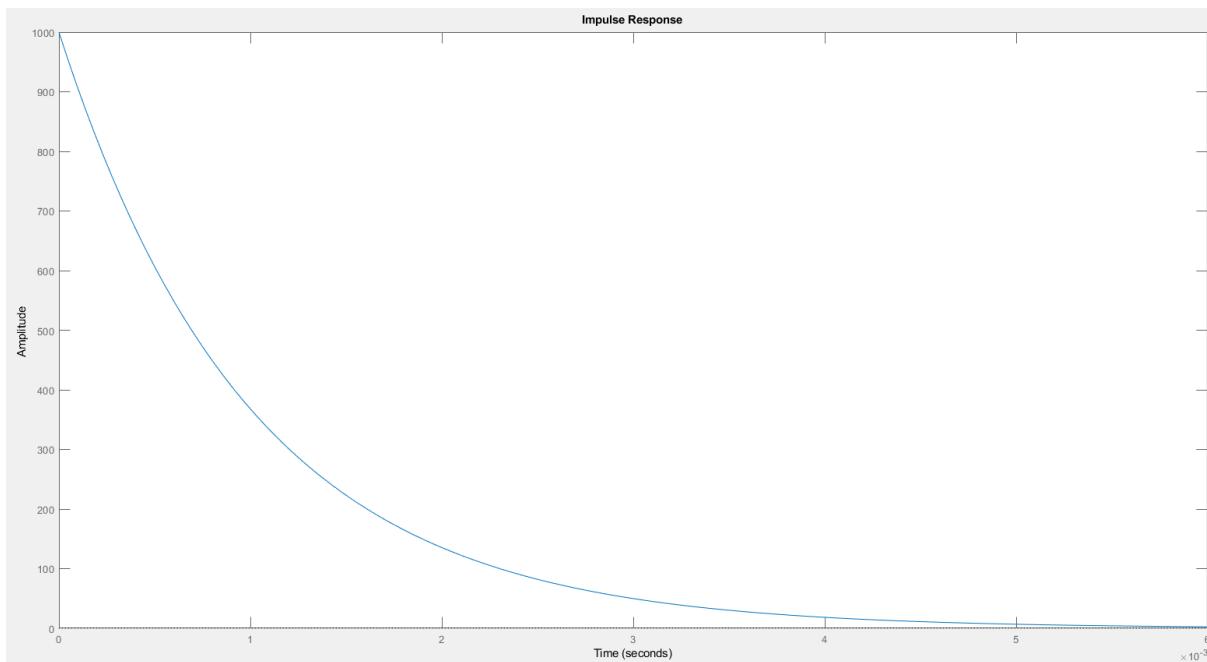
Rys. 5.10. Wynik z funkcji tf(L, M).

Następnie dla stworzonych modeli wyznaczone zostały charakterystyki częstotliwościowe i czasowe za pomocą funkcji freqs(L, M), step(L, M), impulse(L, M). Są to funkcje zwracające charakterystyki

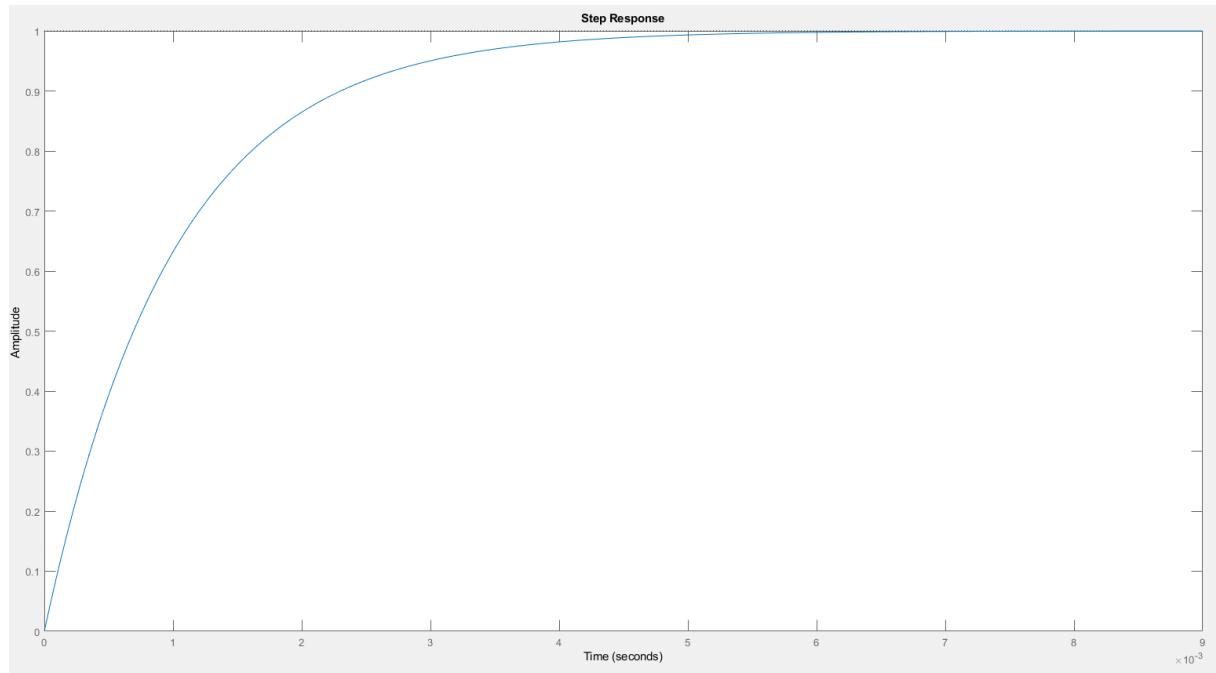
Bodego (amplitudowo i fazowo-częstotliwościowe), skokową i impulsową. Ostatnim etapem było wykreślenie zer i biegunów układu za pomocą funkcji iopzplot(sys) oraz transformacja układu opisanego wielomianami na postać biegunoową. Służy do tego funkcja tf2zp(sys).



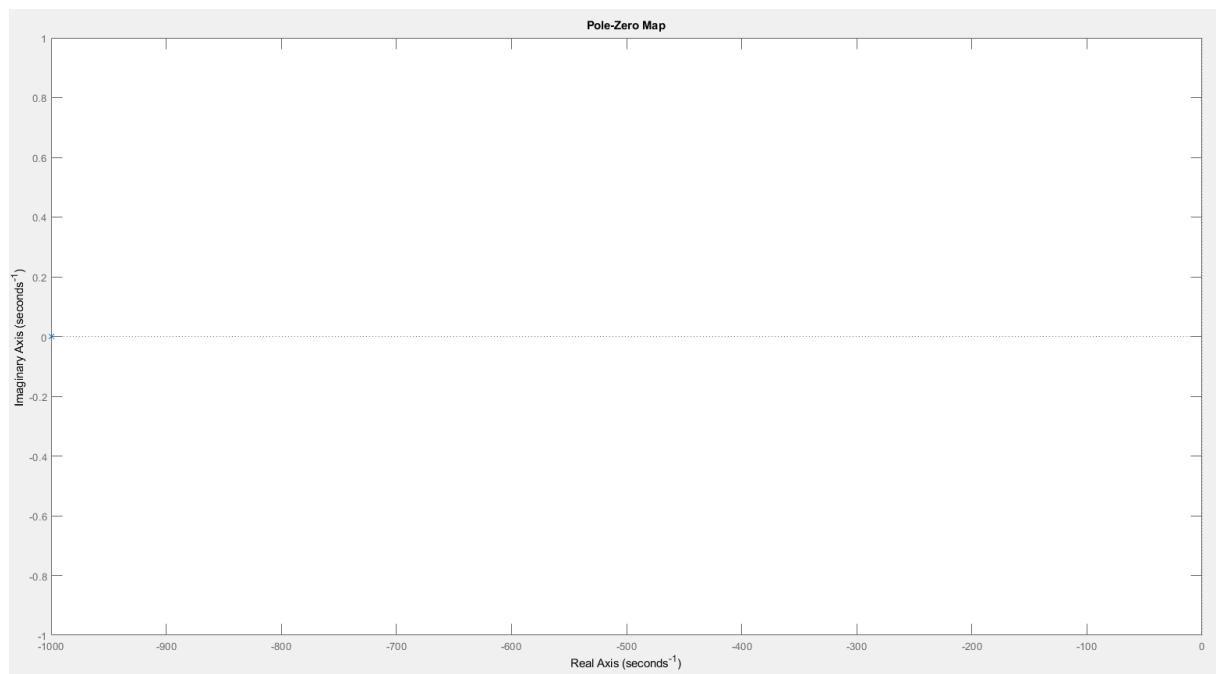
Rys. 5.11. Charakterystyki Bodego dla układu RC.



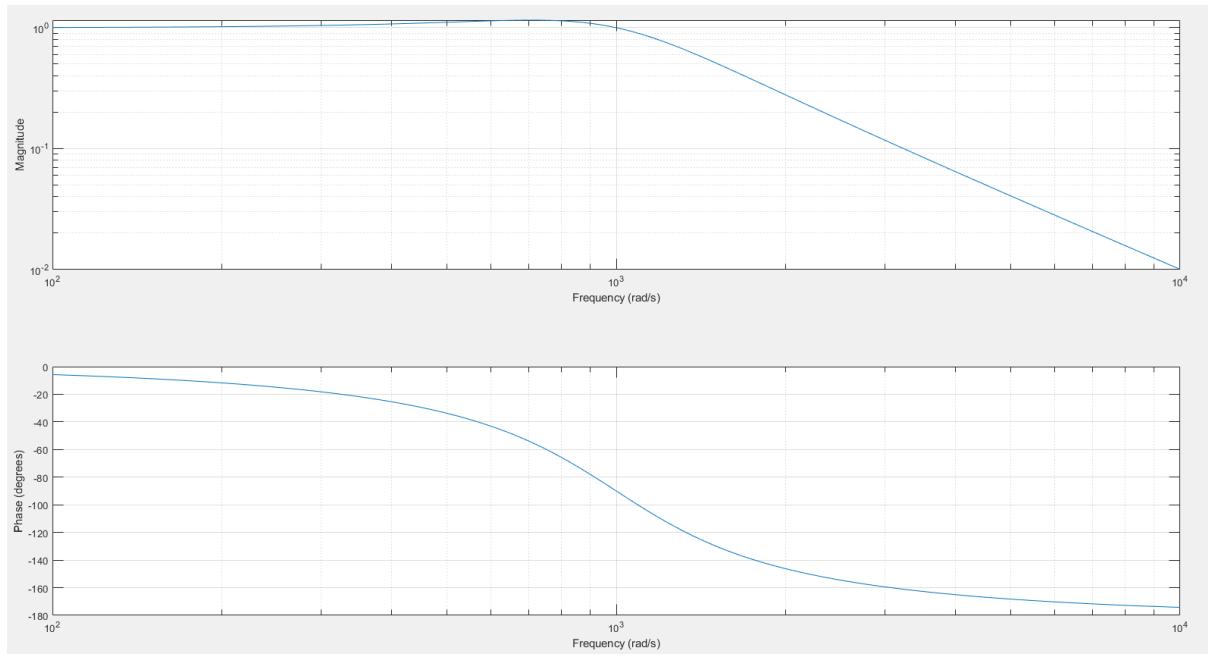
Rys. 5.12. Odpowiedź impulsowa układu RC.



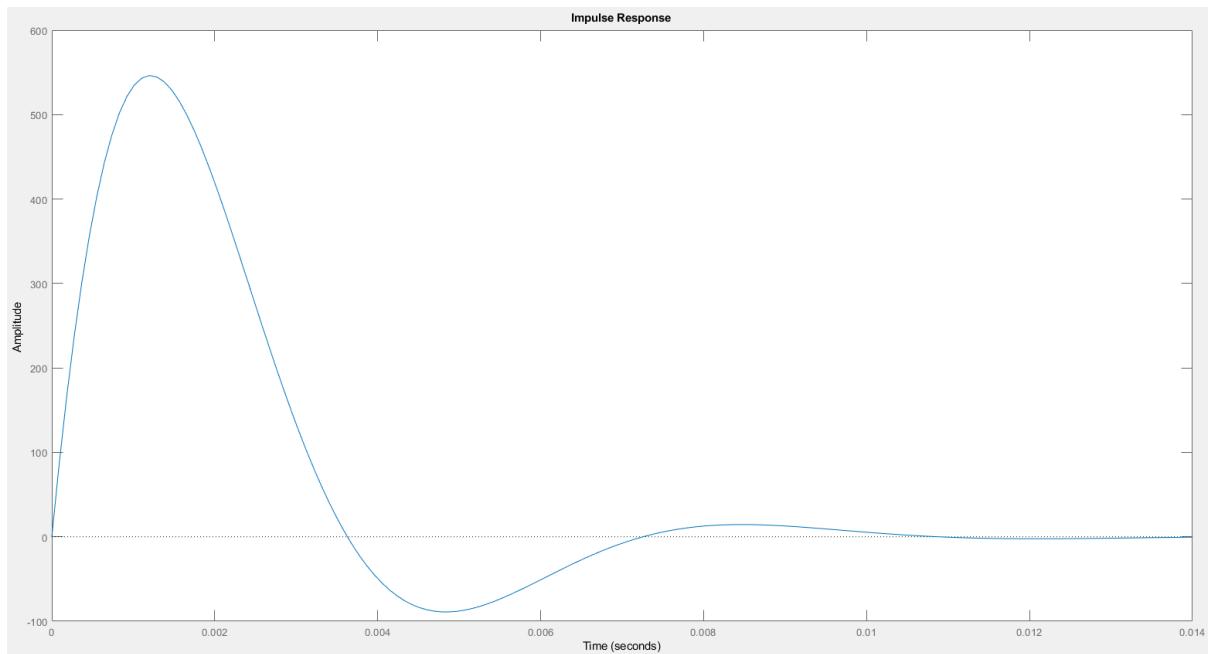
Rys. 5.13. Odpowiedź skokowa układu RC.



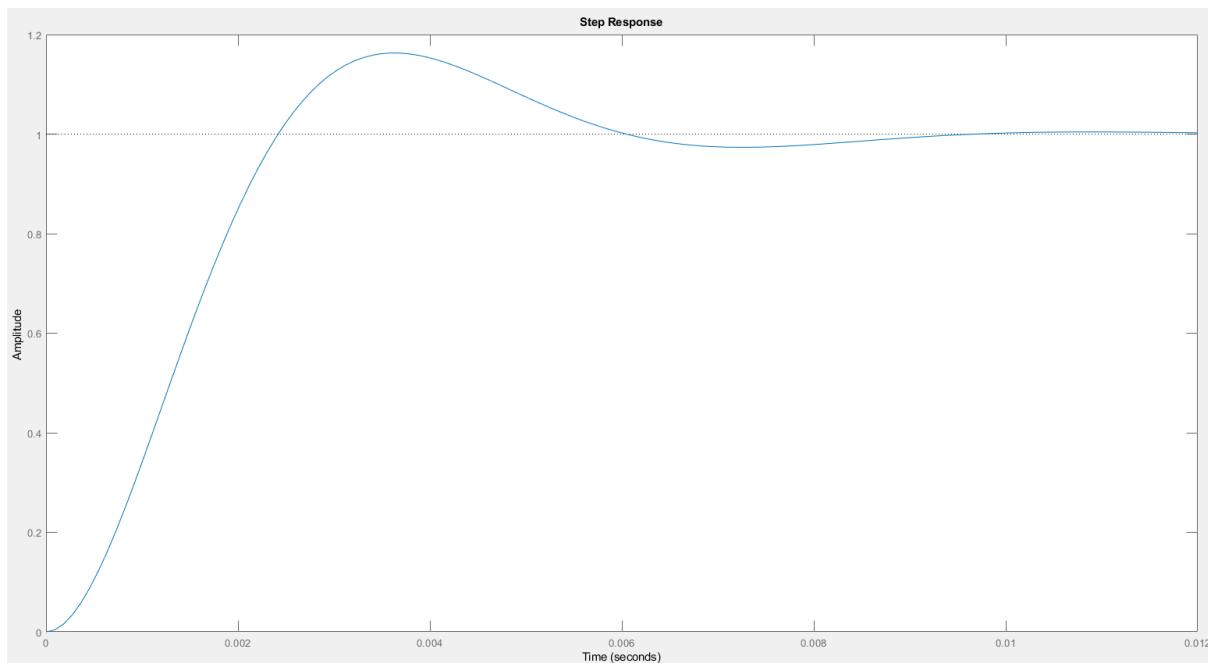
Rys. 5.14. Biegun układu RC (prawie w zerze).



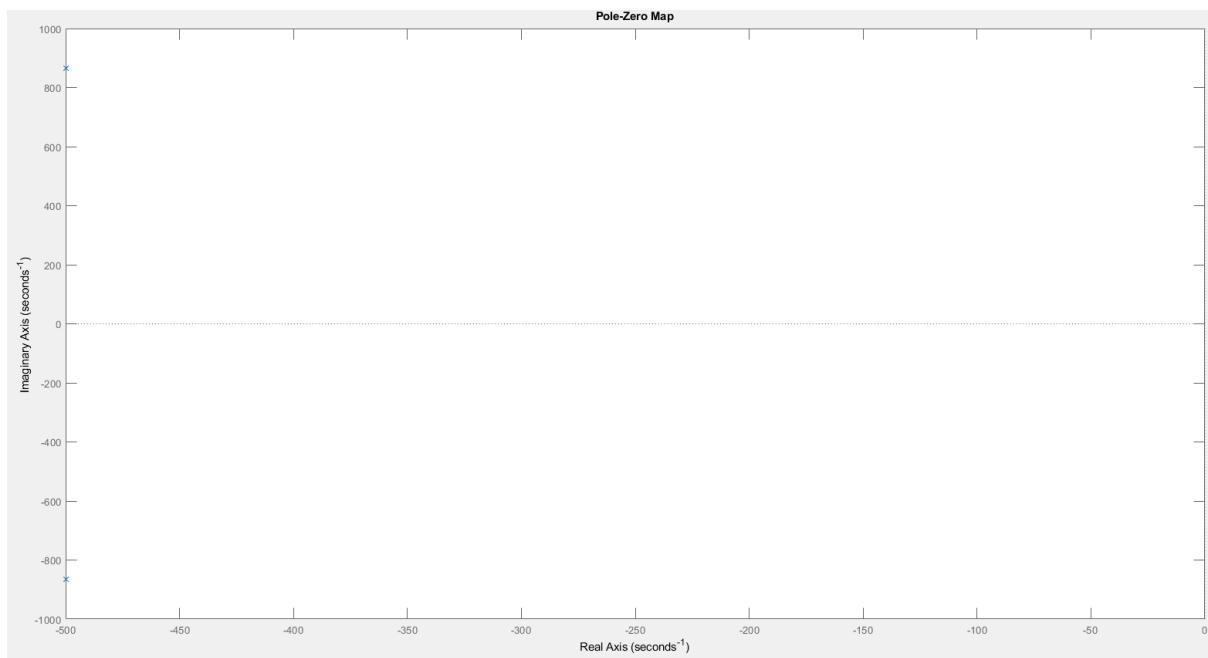
Rys. 5.15. Charakterystyki Bodego dla układu RLC.



Rys. 5.16. Odpowiedź impulsowa układu RLC.



Rys. 5.17. Odpowiedź skokowa układu RLC.



Rys. 5.18. Biegunki układu RLC, wzajemnie sprzężone.

Na podstawie analizy transmitancji układu możemy określić jego stabilność. Biegunki układu muszą znajdować się w lewej półpłaszczyźnie lub na osi urojonej. Odpowiedź impulsowa musi być asymptotycznie zbieżna do zera, podobnie skokowa do amplitudy sygnału skokowego na wejściu.

6. Laboratorium 6

W ramach tego ćwiczenia laboratoryjnego wykreślano zostały charakterystyki amplitudowe i fazowe, impulsowe, skokowe oraz dokonana została ocena stabilności układów o zadanych transmitancjach w dziedzinie Z. Następnie dla filtra o skończonej odpowiedzi impulsowej (FIR) zostało

sprawdzone jego działanie, oraz został zaprojektowany filtr cyfrowy w narzędzi Simulink, który wykorzystuje blokowy język programowania.

Tab. 6.1. Kod programu „Laboratoria_nr_6_1.m”.

```
% Lab 6. Dla zadanych transmitancji wykreślić ich
% odpowiedzi impulsowe, skokowe, charakterystyki
% amplitudowe i fazowe oraz zbadac stabilność.
%
% Mateusz Krupnik

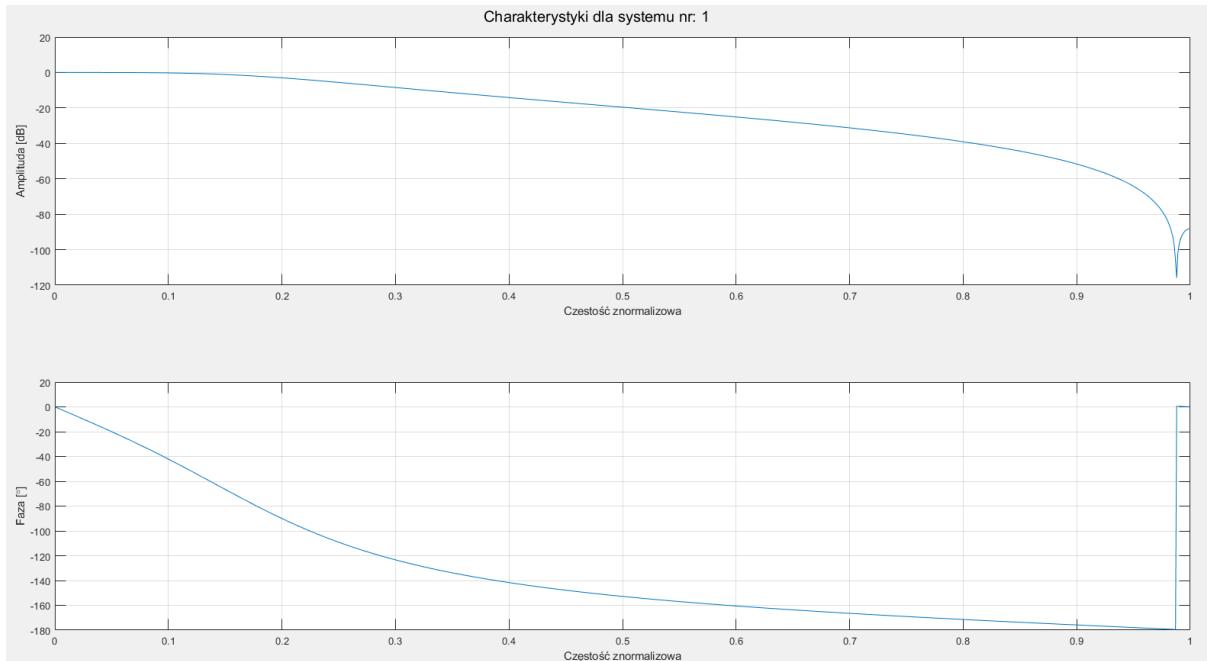
% CZESC 1: Wykreślenie charakterystyk dla 4 zestawów współczynników
% transmitacji układów (KODY 1-4 w Lab. 6)
clc; clear all; close all;
L=[0.0675 0.1349 0.0675; 0.0412 0.0824 0.0412; ...
    0.0996 0.1297 0.0996; 0.1239 0.0662 0.1239]; % Licznik
M=[1 -1.1430 0.4128; 1 -1.4409 0.6737; 1 -1.6099 0.6794; ...
    1 -1.4412 0.6979]; % Mianownik
d_k = zeros(1, 1001); d_k(1,1)=1;
fs=1000; f=100; t=0:(1/fs):1;
for i=1:size(L,1)
    disp(['System nr: ' num2str(i)]);
    l = (L(i, :)); m = (M(i, :));
    sys=tf(l, m) % Transfer Function
    % Wykresy systemu
    [h, w] = freqz(l, m, fs); % Char. ampl i fazowa
    Mag = 20*log10(abs(h)); % Amplituda w skali log
    F = phase(h)*180/pi; % Faza w stopiach
    w = w/pi; % skalowanie
    figure(4*i-3)
    sgtitle(['Charakterystyki dla systemu nr: ' num2str(i)]);
    subplot(211); plot(w, Mag);
    ylabel('Amplituda [dB]'); grid;
    xlabel('Częstość znormalizowana');
    subplot(212); plot(w, F);
    ylabel('Faza [\circ]'); grid;
    xlabel('Częstość znormalizowana');
    figure(4*i-2)
    subplot(211);
    impulse(l, m); grid; % impuls dla Z transmitacji
    subplot(212);
    dstep(l, m); grid; % odp skokowa

    y = filter(l, m, d_k); % odpowiedz filtra o Z transmitacji na x
    figure(4*i-1)
    plot(t(1:50),d_k(1:50), t(1:50), y(1:50));
    title(['Odpowiedź dla systemu nr: ' num2str(i)]);
    xlabel('Czas [s]'); grid;
    ylabel('Amplituda'); legend('Wymuszenie', 'Odpowiedź');

    a=0; b=0; r=1; %
    x = linspace(a-r,a+r,100);
    y1=sqrt(r^2-(x-a) .^2)+b;
    y2=-sqrt(r^2-(x-a) .^2)+b;
    figure(4*i)
    plot(x,[y1; y2], 'b'); grid on; axis equal; hold on;
    pzmap(l, m); % Wyrysowanie zer i biegunkow
    [z, p, k] = tf2zp(l, m);
end
```

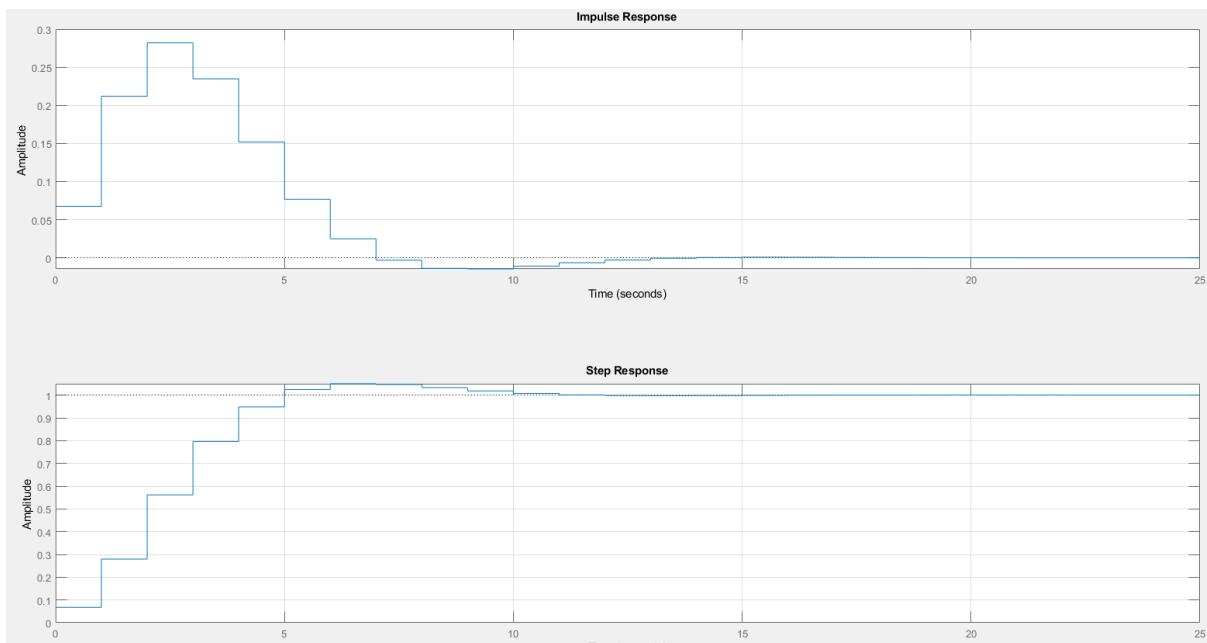
Poniżej pokazane zostaną charakterystyki dla każdego z badanych układów. Transmitancja pierwszego układu:

$$H(z) = \frac{0,0675 + 0,1349z^{-1} + 0,0675z^{-2}}{1 - 1,1430z^{-1} + 0,4128z^{-2}}$$

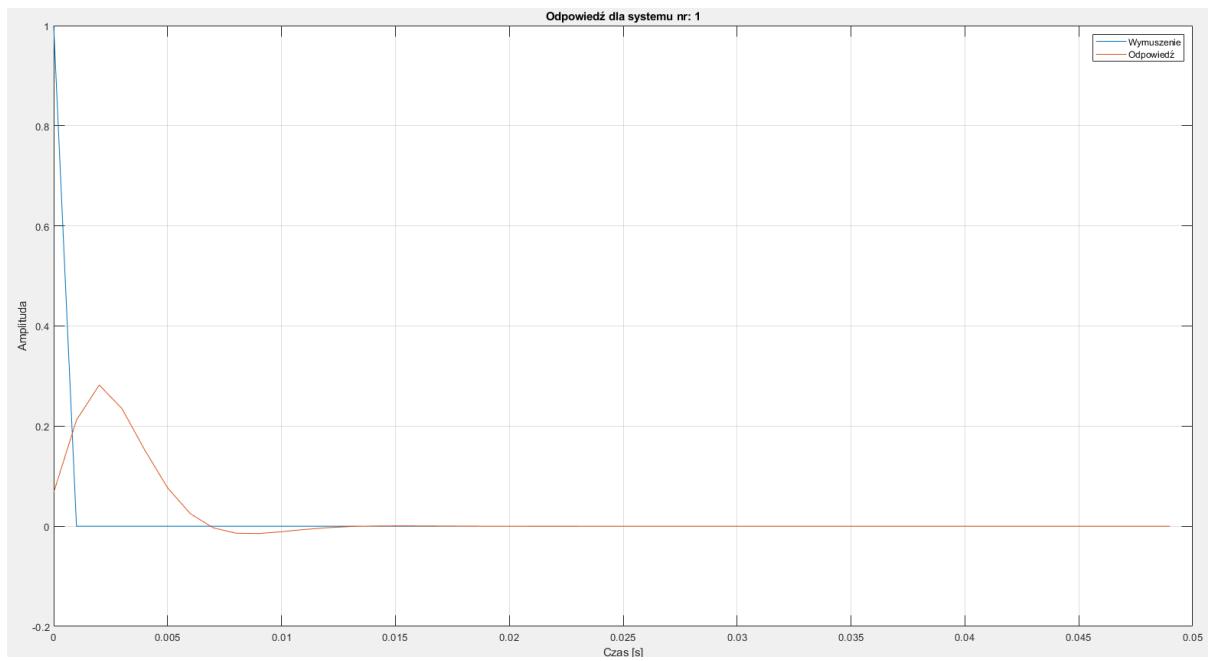


Rys. 6.1. Charakterystyki amplitudowo-częstotliwościowe i fazowo-częstotliwościowe dla układu nr 1.

Jak można zauważyć charakterystyki otrzymane za pomocą funkcji freqz(licznik, mianownik), która wyznacza odpowiedź układu dyskretnego o transmitancji podanej w postaci współczynników wielomianów licznika i mianownika.

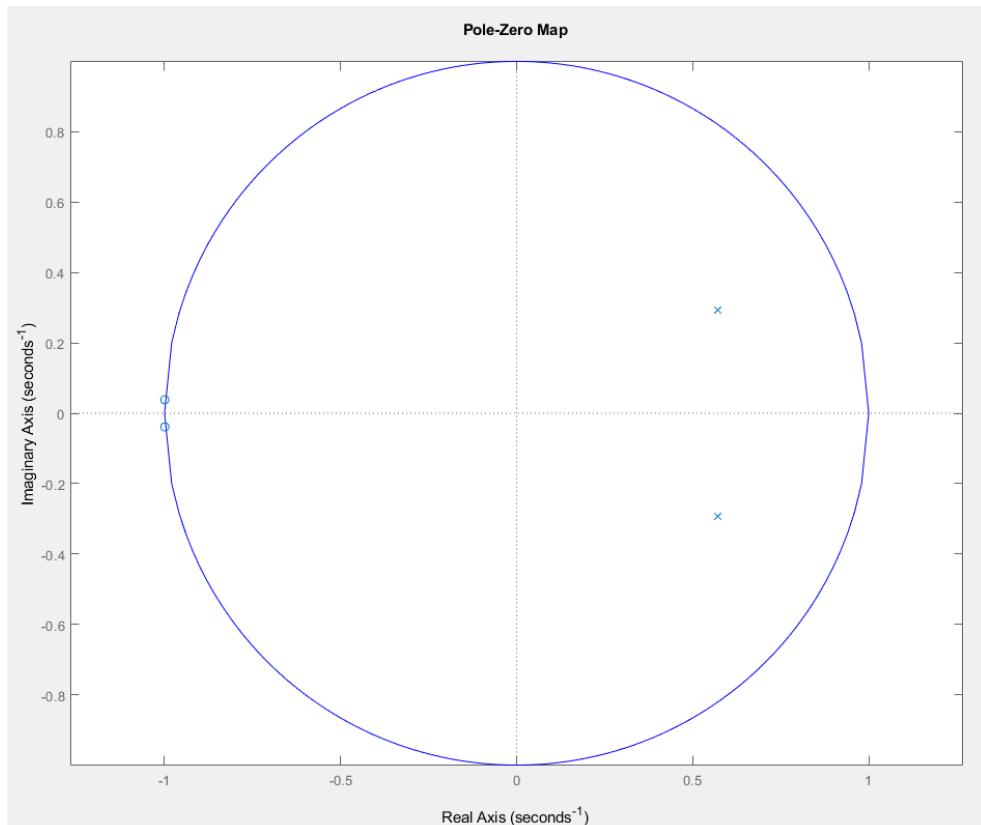


Rys. 6.2. Odpowiedź impulsowa i skokowa układu nr 1.



Rys. 6.3. Odpowiedź układu nr 1 na pobudzenie deltą Kroneckera.

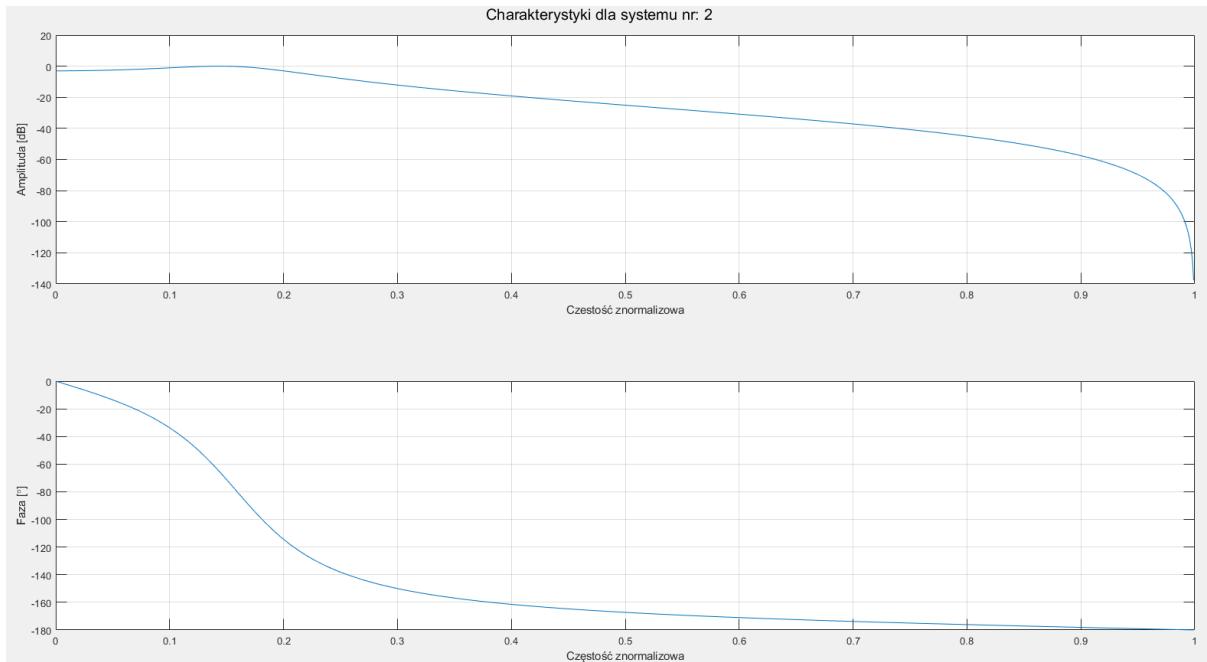
Jak można zauważyć układ po wzbudzeniu impulsem powraca do stanu początkowego co świadczy o jego stabilności, dodatkowo można zauważyć to na podstawie odpowiedzi skokowej. Poniższy rysunek przedstawia rozkład zer i biegunków na płaszczyźnie zespolonej po transformacji bilinowej. Jak widać zera mianownika leżą na okręgu jednostkowym, a więc układ jest stabilny.



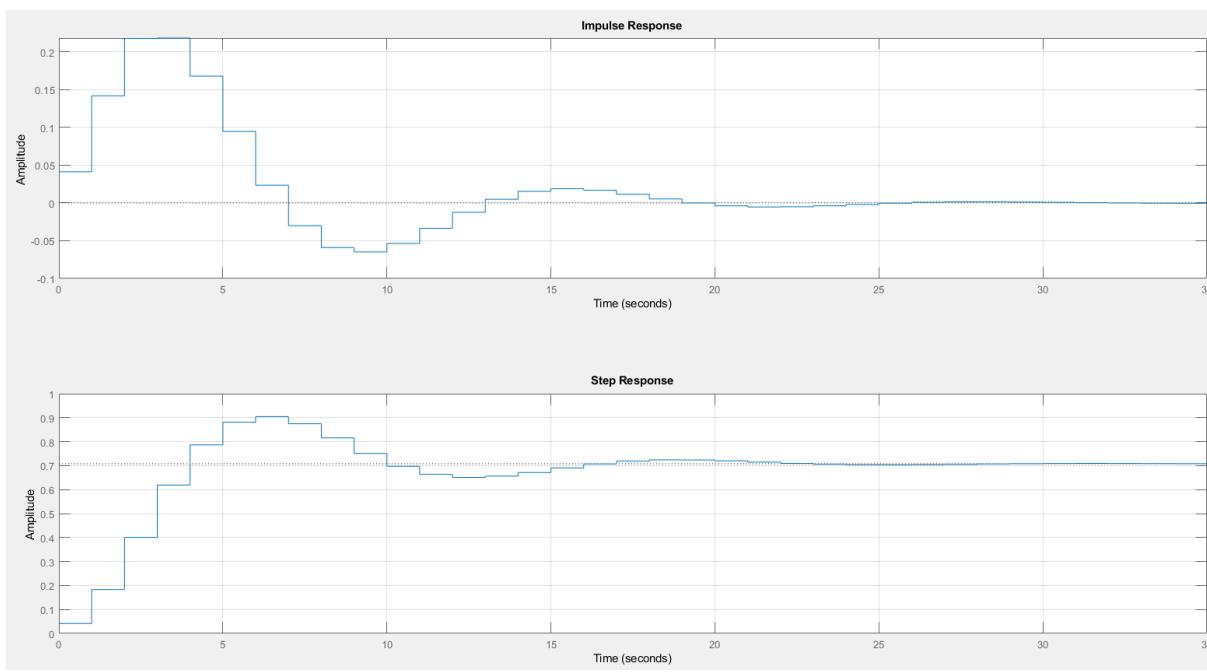
Rys. 6.4. Wykres położenia zer (x) i biegunków (o) układu nr 1.

Transmitancja drugiego układu:

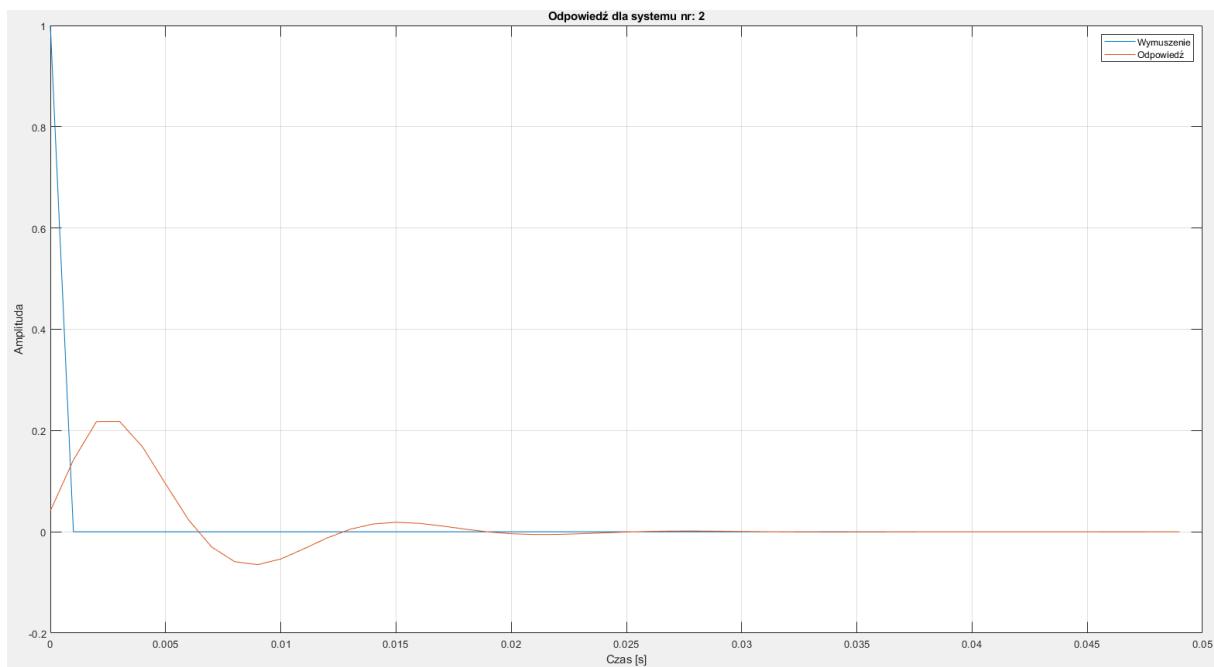
$$H(z) = \frac{0,0412 + 0,0824z^{-1} + 0,0412z^{-2}}{1 - 1,14409z^{-1} + 0,6737z^{-2}}$$



Rys. 6.5. Charakterystyki amplitudowo-częstotliwościowe i fazowo-częstotliwościowe dla układu nr 2.

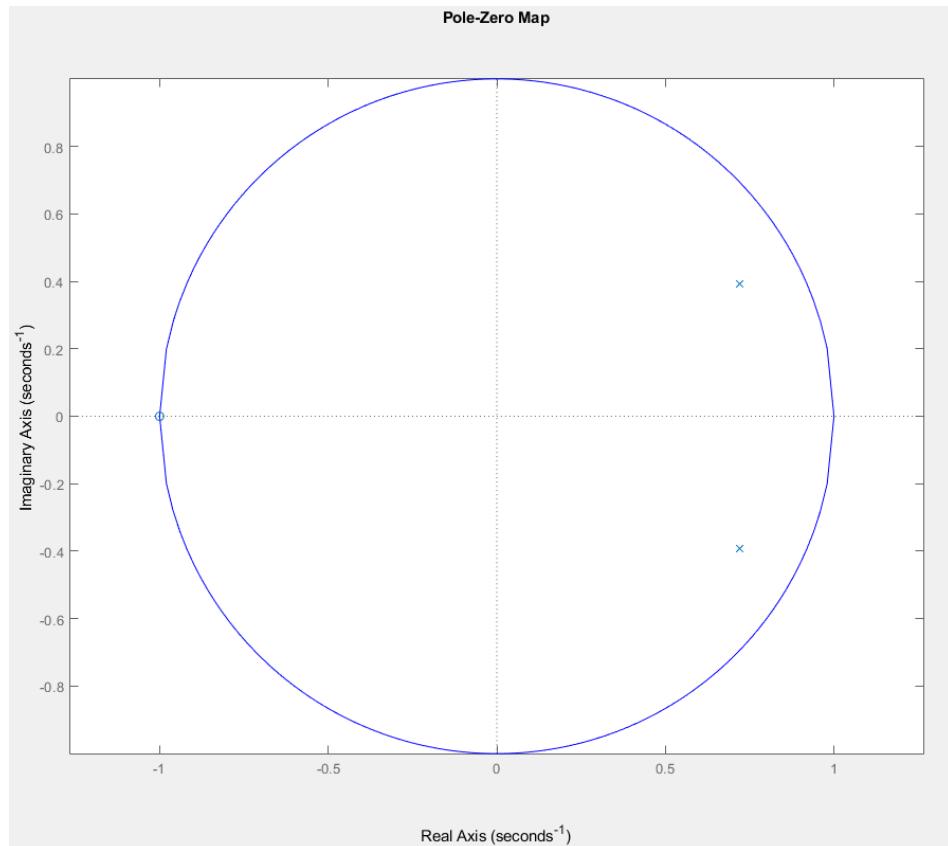


Rys. 6.6. Odpowiedź impulsowa i skokowa układu nr 2.



Rys. 6.7. Odpowiedź układu nr 2 na pobudzenie deltą Kroneckera.

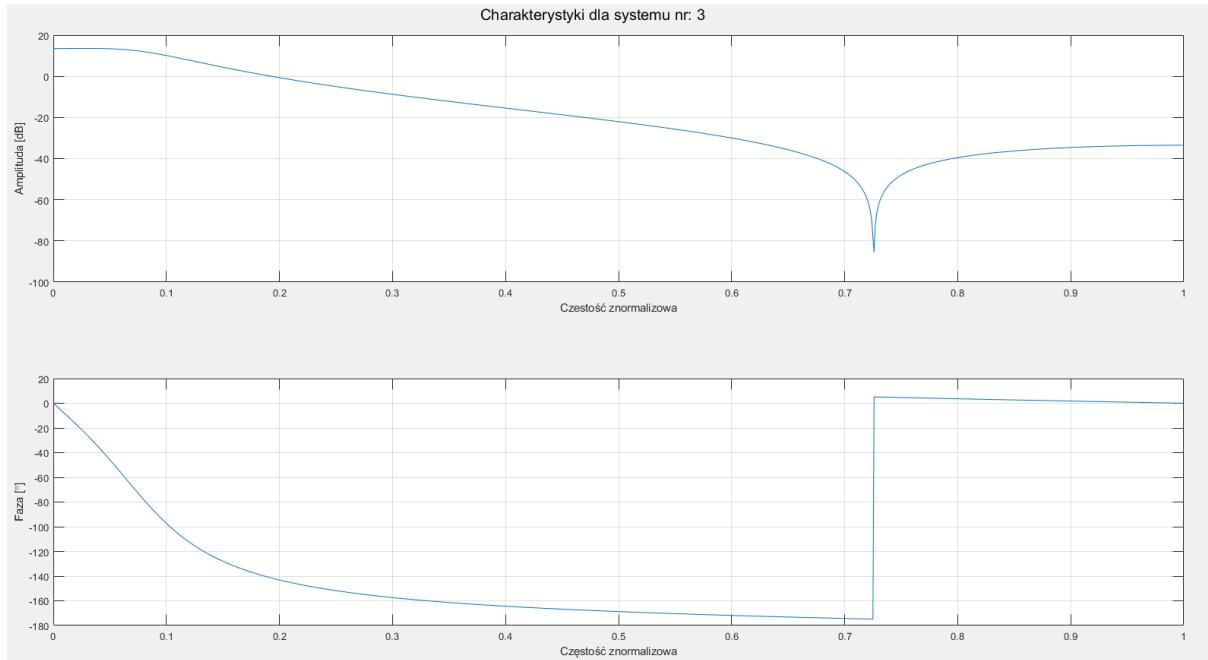
Podobnie jak w poprzednim przypadku układ jest stabilny, jednak w tym przypadku biegun jest podwójny. W odpowiedzi impulsowej jak i skokowej i widać dłuższe oscylacje (pojawia się druga dodatnia góruka). Z kolei na charakterystyce amplitudowej widoczne jest wzmacnianie składowych o częstotliwości znormalizowanej około 0,15.



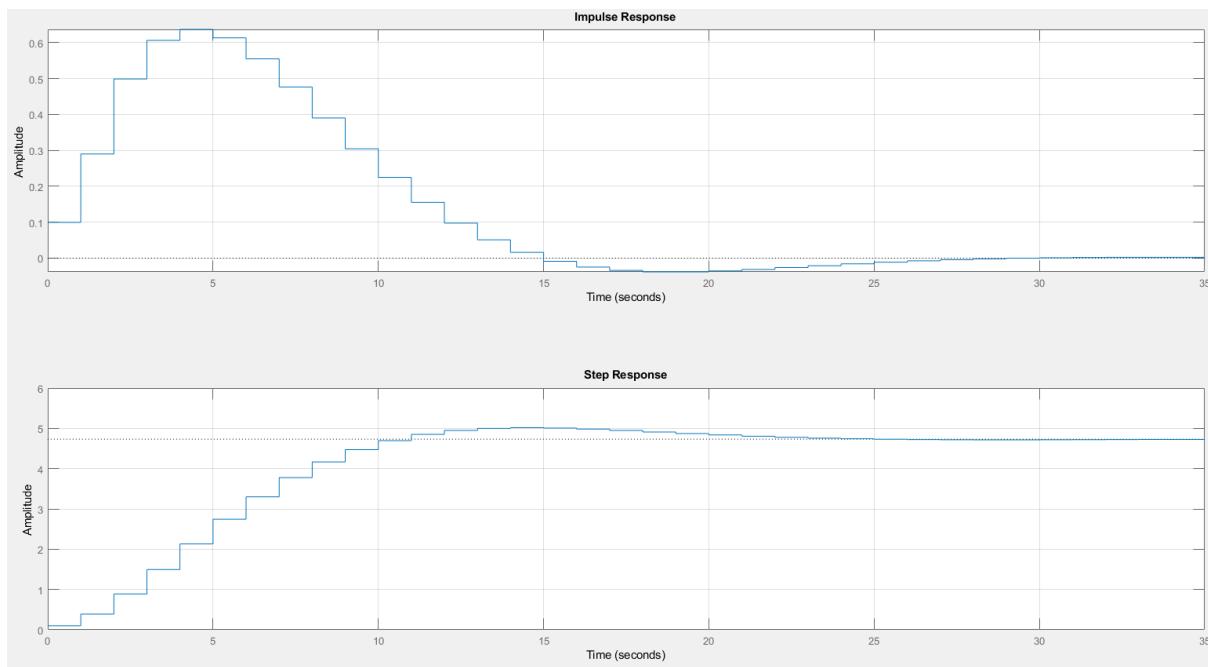
Rys. 6.8. Wykres położenia zer (x) i biegunków (o) układu nr 2.

Transmitancja trzeciego układu:

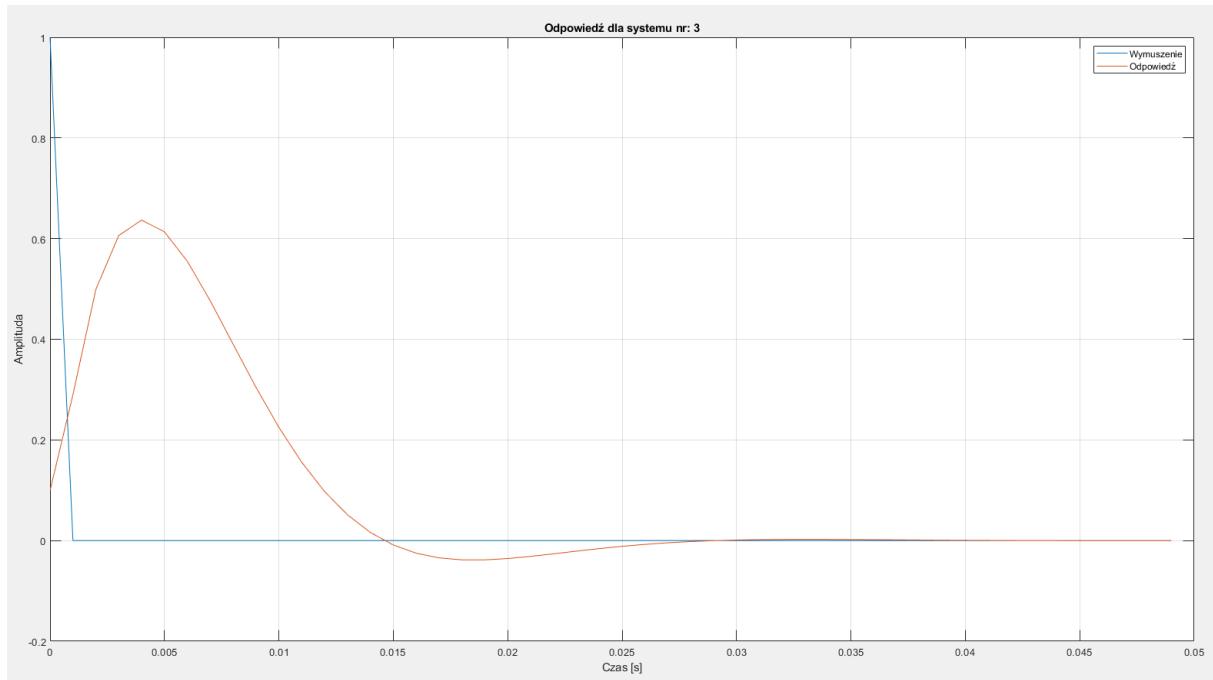
$$H(z) = \frac{0,0996 + 0,1297z^{-1} + 0,0996z^{-2}}{1 - 1,6099z^{-1} + 0,6794z^{-2}}$$



Rys. 6.9. Charakterystyki amplitudowo-częstotliwościowe i fazowo-częstotliwościowe dla układu nr 3.

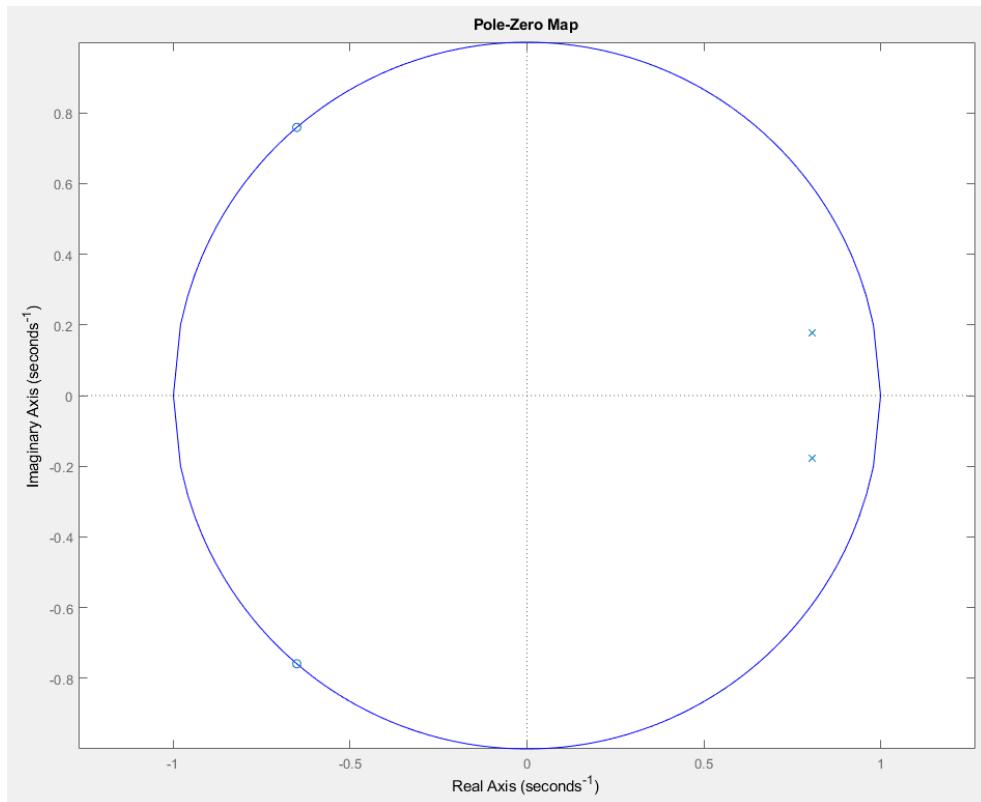


Rys. 6.10. Odpowiedź impulsowa i skokowa układu nr 3.



Rys. 6.11. Odpowiedź układu nr 3 na pobudzenie deltą Kroneckera.

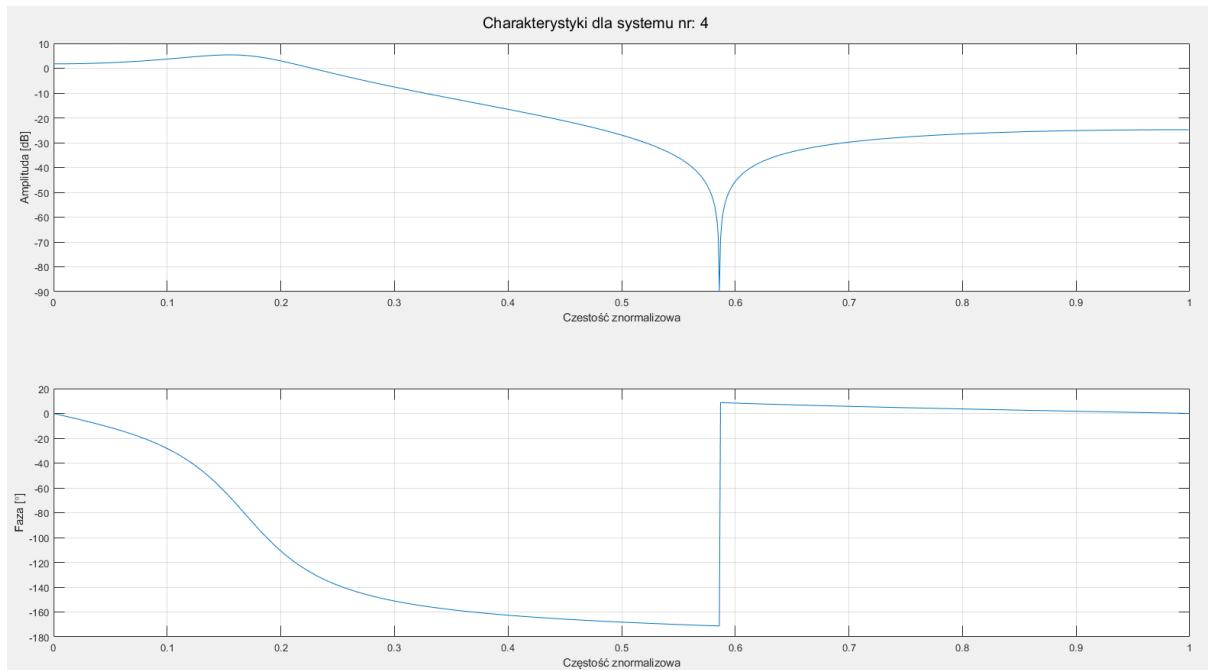
Trzeci układ również wykazuje się stabilnością, widoczne są dobrze sprzężone ze sobą zera i biegunki.



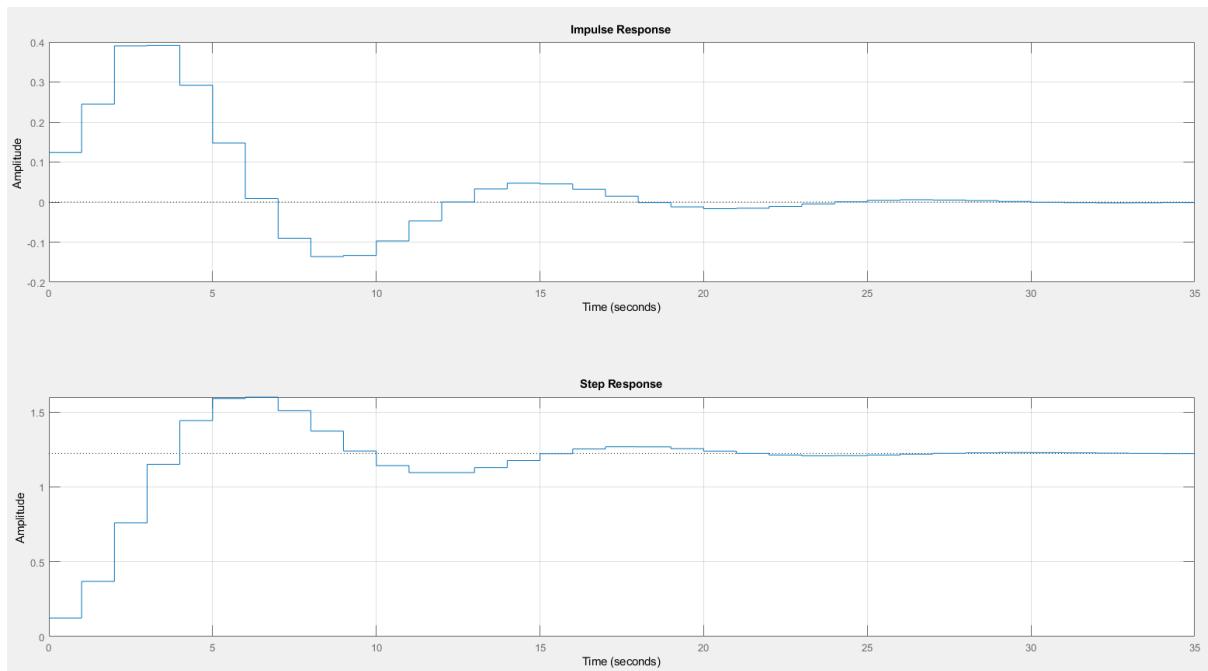
Rys. 6.12. Wykres położenia zer (x) i biegunków (o) układu nr 3.

Transmitancja ostatniego układu:

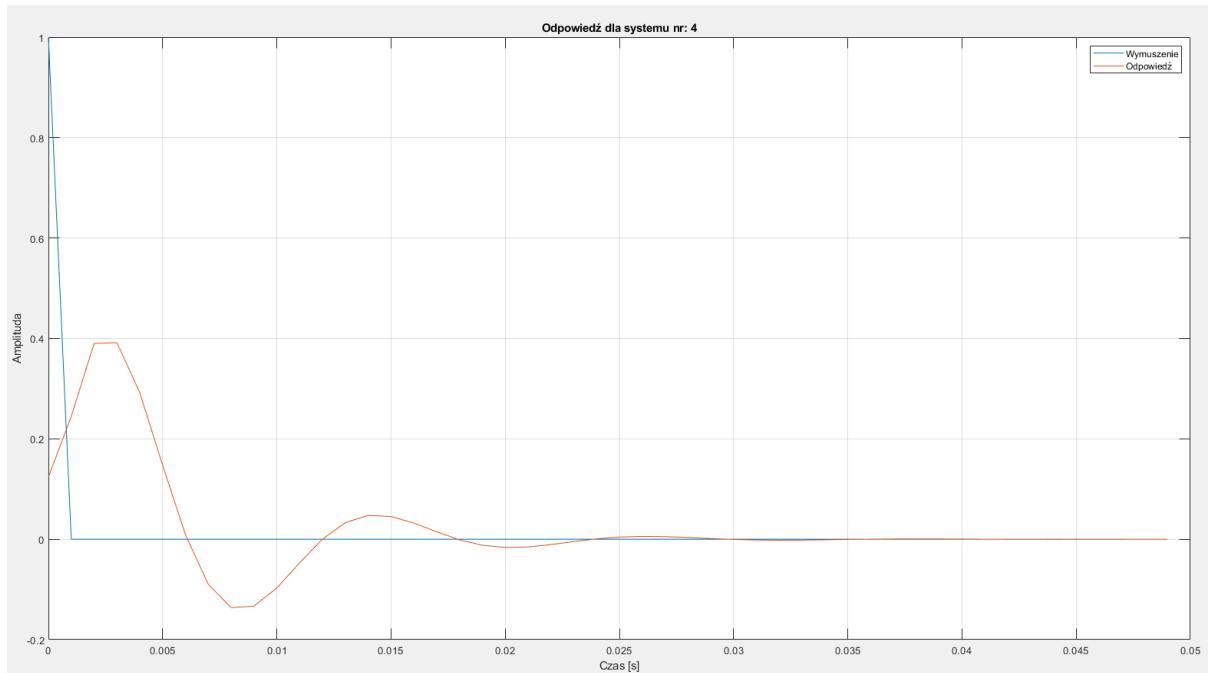
$$H(z) = \frac{0,1239 + 0,0662z^{-1} + 0,1239z^{-2}}{1 - 1,4412z^{-1} + 0,6979z^{-2}}$$



Rys. 6.13. Charakterystyki amplitudowo-częstotliwościowe i fazowo-częstotliwościowe dla układu nr 4.

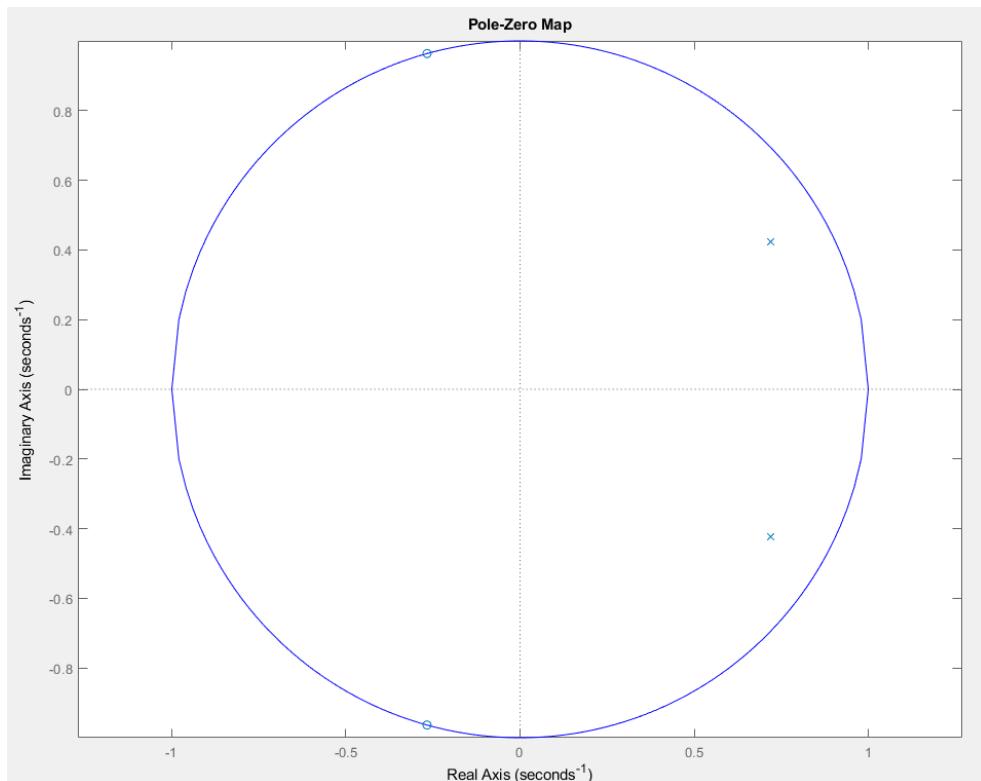


Rys. 6.14. Odpowiedź impulsowa i skokowa układu nr 4.



Rys. 6.15. Odpowiedź układu nr 4 na pobudzenie deltą Kroneckera.

Ostatni układ również wykazuje stabilność, jednak jak można zauważyć posiada on największe oscylacje. Odpowiedź impulsowa ewidentnie posiada trzecią dodatnią górkę.



Rys. 6.16. Wykres położenia zer (x) i biegunków (o) układu nr 4.

Następnie dla sygnału o składającym się z 3 sygnałów sinusoidalnych o różnych częstotliwościach dokonano filtracji filtrem dolnoprzepustowym oraz średkowo przepustowym. W tym celu wykorzystano funkcję butter(rząd, częstotliwość_znormalizowana) do zaprojektowania filtra Butterwortha o zadanych rzędzie i znormalizowanej częstotliwości odcięcia. Filtracje dolnoprzepustową dla rzędu równego 8 i częstotliwości odcięcia równej odpowiednio 300 i 200 Hz przedstawiono poniżej tabeli kodu.

Tab. 6.2. Kod programu "Krupnik_Lab_6_2.m".

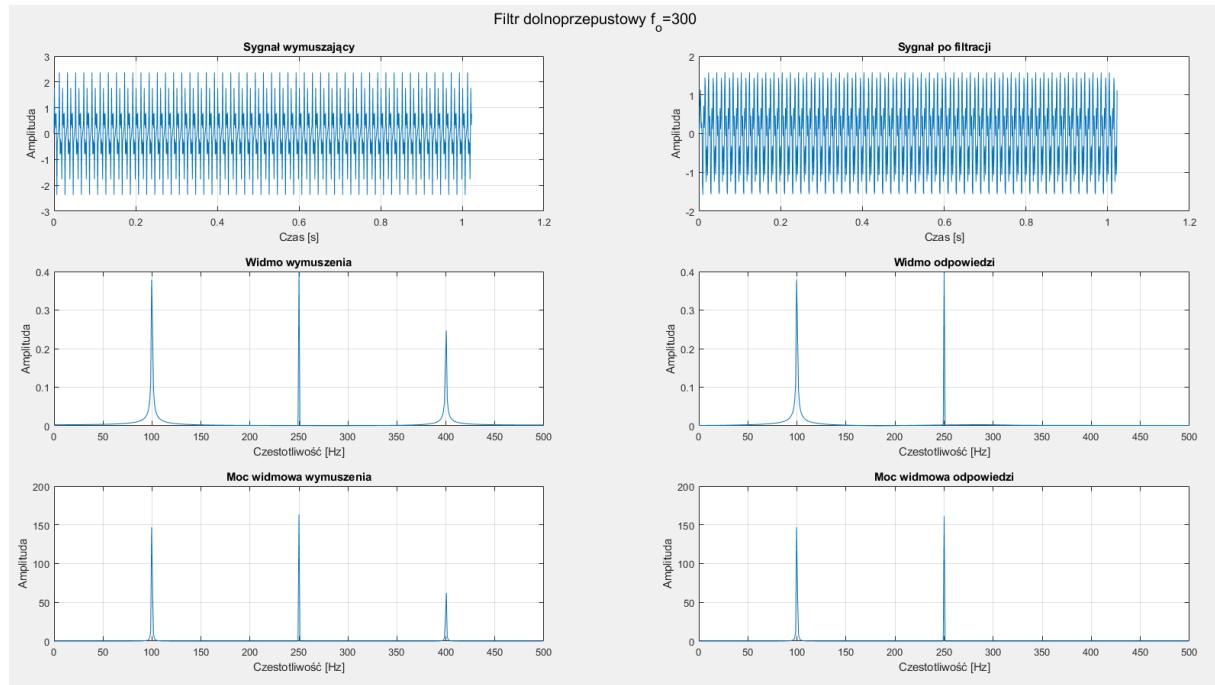
```
% Lab 6.
% Dla filtru FIR sprawdzić jego działanie.
%                                         Mateusz Krupnik
clear all; close all; clc;
%% Sprawdzenie działania filtrów cyfrowych
% Dane sygnałów
f1=100; f2=250; f3=400; fs=1000;      % Częstotliwości składowych i Nyquista
A1=1; A2=0.8; A3=0.65;                  % Aplitudy składowych
t=0:(1/fs):1.023;                      % Wektor czasu
% Sygnał wymuszenia - składowa 3 harmonicznych
x=A1*sin(2*pi*f1*t)+A2*sin(2*pi*f2*t)+A3*sin(2*pi*f3*t);

% Parametry
fo1 = 300; fo2 = 200;                   % Częstotliwości odcięcia
wn1 = fo1*2/fs; wn2 = fo2*2/fs; % Znormalizowane częstotliwości
wn = [wn1, wn2]; fo = [fo1, fo2];
% Filtre Butter - dolnoprzepustowy
N = [2, 4 ,8]; rząd = 3;    % wybór rzędu filtra

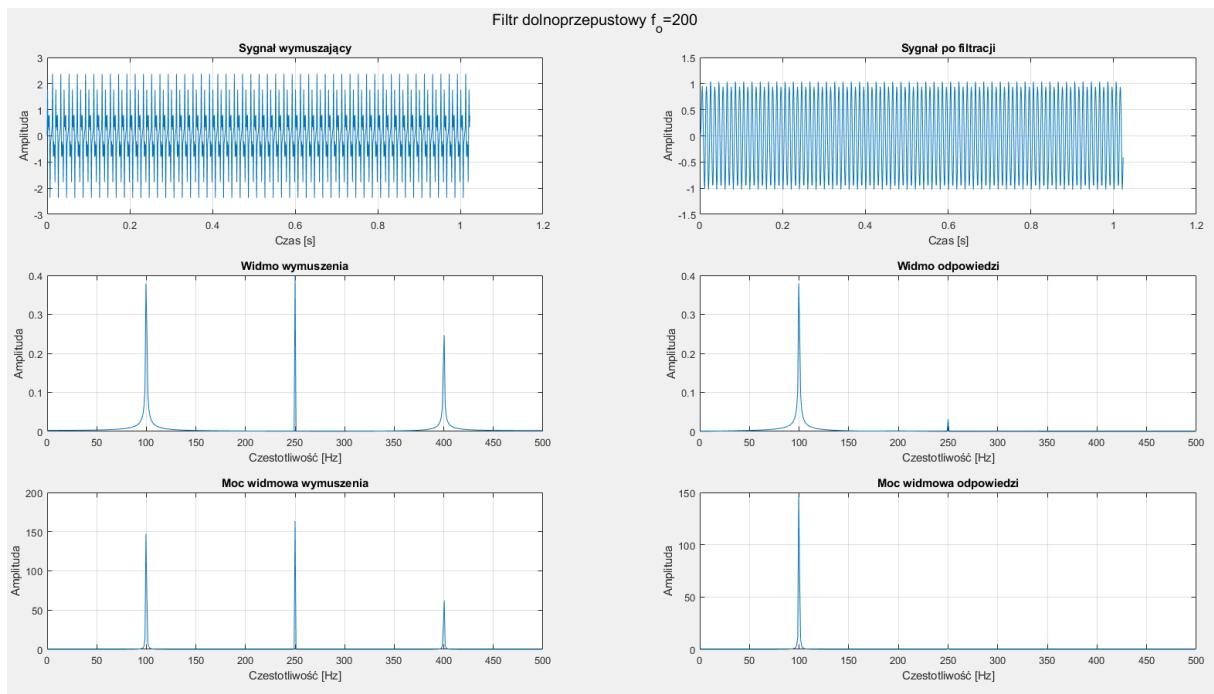
%% Filtre dolnoprzepustowy Butterwatha

for i=1:length(wn)
    [l, m] = butter(N(1,rząd), wn(i));      % Rząd oraz częst. odcięcia
    y = filter(l, m, x);                     % Odpowiedź filtra
    % Obliczenie transformaty Fouriera
    [f_w, Moc, Wid] = fft_from_signal([x; y], fs);    % Funkcja z Lab 4 i 5
    % Dźwięk
    sound(x); pause(t(end)); sound(y); pause(t(end));
    % Wykresy
    figure(16+i);
    sgtitle(['Filtr dolnoprzepustowy f_o=' num2str(fo(i))]);
    subplot(321); plot(t, x);
    xlabel('Czas [s]'); ylabel('Amplituda'); grid;
    title('Sygnał wymuszający');
    subplot(322); plot(t, y);
    xlabel('Czas [s]'); ylabel('Amplituda'); grid;
    title('Sygnał po filtracji');
    subplot(323); plot(f_w, Wid(1,:));
    xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
    title('Widmo wymuszenia');
    subplot(324); plot(f_w, Wid(2,:));
    xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
    title('Widmo odpowiedzi');
    subplot(325); plot(f_w, Moc(1,:));
    xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
    title('Moc widmowa wymuszenia');
    subplot(326); plot(f_w, Moc(2,:));
    xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
    title('Moc widmowa odpowiedzi');
end
```

```
% DEFINICJA FUNKCI %%%%%%
function [f, M, W] = fft_from_signal(y, fs)
% fft_from_signal
% Summary of this function goes here
% Detailed explanation goes here
% y - signal matrix, with signals as rows
% f - frequency, M - power spectrum, W - spectrum
N = length(y);
for i=1:size(y, 1)
    fft_moc=fft(y(i, 1:N));
    moc_wid=fft_moc.*conj(fft_moc)/N;
    widmo=sqrt(fft_moc.*conj(fft_moc))/N;
    f=fs*(0:N/2-1)/N;
    M(i,:)=moc_wid;
    W(i,:)=widmo;
end
M = M(:, floor(1:N/2));
W = W(:, floor(1:N/2));
end
```



Rys. 6.17. Filtracja filtrem dolnoprzepustowym Butterwortha o częstotliwości odcięcia 300 Hz i rzędzie równym 8.



Rys. 6.18. Filtracja filtrem dolnoprzepustowym Butterwortha o częstotliwości odcięcia 200 Hz i rzędzie równym 8.

Jak można zauważyć filtr o częstotliwości odcięcia 300 Hz dobrze usnął składową 400 Hz jednak filtr o częstotliwości odcięcia 200 Hz pozostawił część składowej 250 Hz. Aby to poprawić należałyby zwiększyć rzad filtra lub przesunąć granice odcięcia. Następnie przetestowana została filtracja śródkowoprzepustowa, kod programu i wynik znajdują się poniżej.

Tab. 6.3. Kod programu "Krupnik_Lab_6_3.m".

```
% Lab 6.
% Dla filtra FIR sprawdzić jego działanie.
%
% Mateusz Krupnik
clear all; close all; clc;
%% Sprawdzenie działania filtrów cyfrowych
% Dane sygnałów
f1=100; f2=250; f3=400; fs=1000;      % Częstotliwości składowych i Nyquista
A1=1; A2=0.8; A3=0.65;                  % Aplitudy składowych
t=0:(1/fs):1.023;                      % Wektor czasu
% Sygnał wymuszenia - składowa 3 harmonicznych
x=A1*sin(2*pi*f1*t)+A2*sin(2*pi*f2*t)+A3*sin(2*pi*f3*t);

% Parametry
fo1 = 300; fo2 = 200;                   % Częstotliwości odcięcia
wn1 = fo1*2/fs; wn2 = fo2*2/fs; % Znormalizowane częstotliwości
wn = [wn1, wn2]; fo = [fo1, fo2];
% Filtr Butter - dolnoprzepustowy
N = [2, 4 ,8]; rzad = 3;      % wybór rzędu filtra

%% Filtr śródkowoprzepustowy Butterwortha
% Parametry z poprzedniego filtru
[l, m] = butter(N,1,rzad); fliplr(wn)); % Odwrócenie kolejności cz. odc.
y = filter(l, m, x);                  % Odpowiedź filtra
[f_w, Moc, Wid] = fft_from_signal([x; y], fs); % Funkcja z Lab 4 i 5

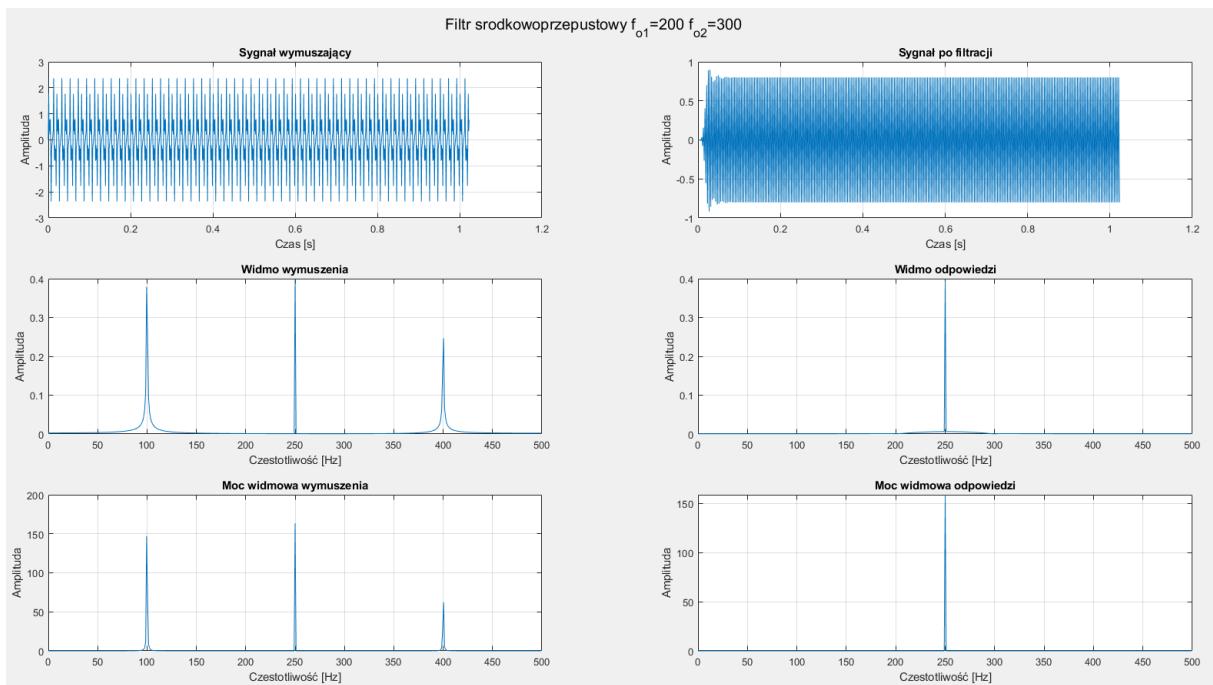
% Dzwiek
```

```

sound(x); pause(t(end)); sound(y); pause(t(end));
% Wykresy
figure(19);
sgtitle(['Filtr srodkowoprzepustowy f_{o1}=' ...
    num2str(fo(2)) ' f_{o2}=' num2str(fo(1))]);
subplot(321); plot(t, x);
xlabel('Czas [s]'); ylabel('Amplituda'); grid;
title('Sygnał wymuszający');
subplot(322); plot(t, y);
xlabel('Czas [s]'); ylabel('Amplituda'); grid;
title('Sygnał po filtracji');
subplot(323); plot(f_w, Wid(1,:));
xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
title('Widmo wymuszenia');
subplot(324); plot(f_w, Wid(2,:));
xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
title('Widmo odpowiedzi');
subplot(325); plot(f_w, Moc(1,:));
xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
title('Moc widmowa wymuszenia');
subplot(326); plot(f_w, Moc(2,:));
xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
title('Moc widmowa odpowiedzi');

% DEFINICJA FUNKCI %%%%%%%%%%%%%%
function [f, M, W] = fft_from_signal(y, fs)
% fft_from_signal
% Summary of this function goes here
% Detailed explanation goes here
% y - signal matrix, with signals as rows
% f - frequency, M - power spectrum, W - spectrum
N = length(y);
for i=1:size(y, 1)
    fft_moc=fft(y(i, 1:N));
    moc_wid=fft_moc.*conj(fft_moc)/N;
    widmo=sqrt(fft_moc.*conj(fft_moc))/N;
    f=fs*(0:N/2-1)/N;
    M(i,:)=moc_wid;
    W(i,:)=widmo;
end
M = M(:, floor(1:N/2));
W = W(:, floor(1:N/2));
end

```



Rys. 6.19. Filtracja filtrem średzkowoprzepustowym Butterwortha 8 rzędu o częstotliwościach granicznych 200 i 300 Hz.

Jak można zauważyć filtr średzkowoprzepustowy poradził sobie z filtracją składowych poza pasmem przepustowym. Brak widocznych składowych wynika z odpowiedniej odległości granic od pozostałych składowych. Kolejno podobnie postąpiono dla filtra górnoprzepustowego.

Tab. 6.4. Kod programu "Krupnik_Lab_6_4.m".

```
% Lab 6.
% Dla filtra FIR sprawdzić jego działanie.
%
% Sprawdzenie działania filtrów cyfrowych
% Dane sygnałów
f1=100; f2=250; f3=400; fs=1000; % Częstotliwości składowych i Nyquista
A1=1; A2=0.8; A3=0.65; % Aplitudy składowych
t=0:(1/fs):1.023; % Wektor czasu
% Sygnał wymuszenia - składowa 3 harmonicznych
x=A1*sin(2*pi*f1*t)+A2*sin(2*pi*f2*t)+A3*sin(2*pi*f3*t);

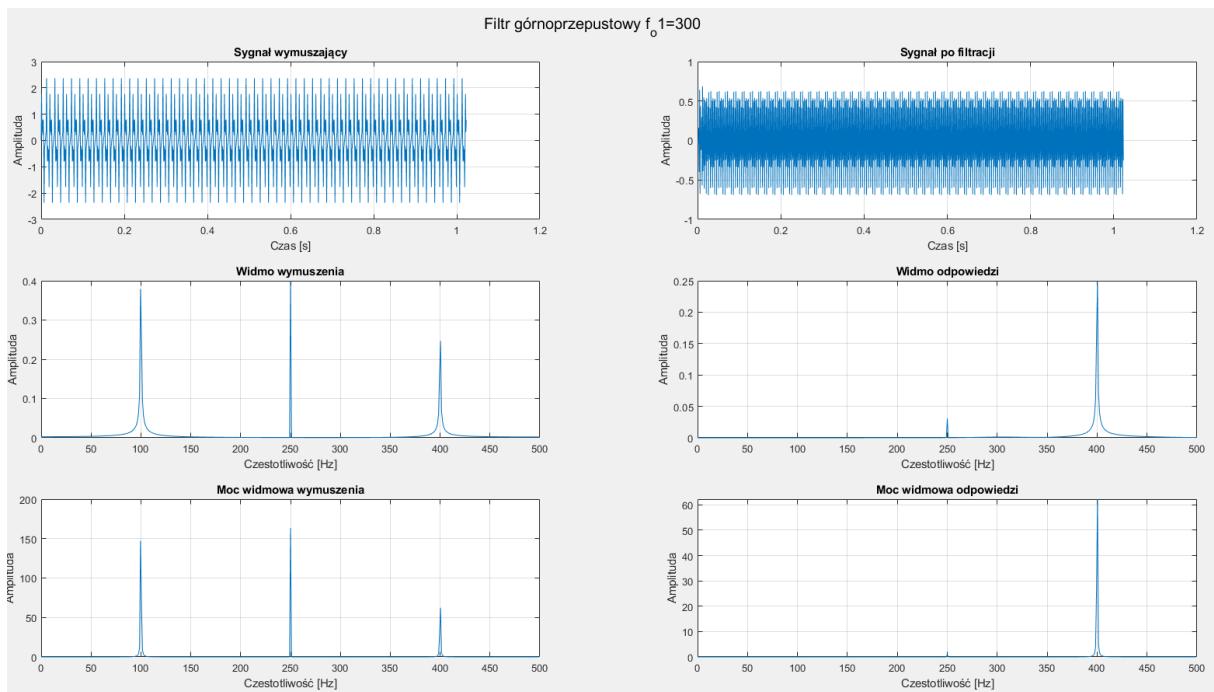
%
% Parametry
fo1 = 300; fo2 = 200; % Częstotliwości ociecia
wn1 = fo1*2/fs; wn2 = fo2*2/fs; % Znormalizowane częstotliwości
wn = [wn1, wn2]; fo = [fo1, fo2];
% Filtr Butter - dolnoprzepustowy
N = [2, 4 ,8]; rzad = 3; % wybór rzędu filtra

%
% Filtr górnoprzepustowy
% Parametry z poprzednich filtrów
[l, m] = butter(N(1,3), wn(1), 'high');
y = filter(l, m, x); % Odpowiedź filtra
[f_w, Moc, Wid] = fft_from_signal([x; y], fs); % Funkcja z Lab 4 i 5

%
% Dźwięk
sound(x); pause(t(end)); sound(y); pause(t(end));
```

```
% Wykresy
figure(20); sgtitle(['Filtr górnoprzepustowy f_ol=' num2str(fo(1))]);
subplot(321); plot(t, x);
xlabel('Czas [s]'); ylabel('Amplituda'); grid;
title('Sygnał wymuszający');
subplot(322); plot(t, y);
xlabel('Czas [s]'); ylabel('Amplituda'); grid;
title('Sygnał po filtracji');
subplot(323); plot(f_w, Wid(1,:));
xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
title('Widmo wymuszenia');
subplot(324); plot(f_w, Wid(2,:));
xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
title('Widmo odpowiedzi');
subplot(325); plot(f_w, Moc(1,:));
xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
title('Moc widmowa wymuszenia');
subplot(326); plot(f_w, Moc(2,:));
xlabel('Częstotliwość [Hz]'); ylabel('Amplituda'); grid;
title('Moc widmowa odpowiedzi');

% DEFINICJA FUNKCI %%%%%%
function [f, M, W] = fft_from_signal(y, fs)
% fft_from_signal
% Summary of this function goes here
% Detailed explanation goes here
% y - signal matrix, with signals as rows
% f - frequency, M - power spectrum, W - spectrum
N = length(y);
for i=1:size(y, 1)
    fft_moc=fft(y(i, 1:N));
    moc_wid=fft_moc.*conj(fft_moc)/N;
    widmo=sqrt(fft_moc.*conj(fft_moc))/N;
    f=fs*(0:N/2-1)/N;
    M(i,:)=moc_wid;
    W(i,:)=widmo;
end
M = M(:, floor(1:N/2));
W = W(:, floor(1:N/2));
end
```



Rys. 6.20. Filtracja filtrem górnoprzepustowym Butterwortha 8 rzędu o częstotliwości odcięcia 300 Hz.

Filtr górnoprzepustowy zadziałał odwrotnie niż dolnoprzepustowy. Podobnie widoczny jest mały prążek częstotliwości 250 Hz. Należało przesunąć granice lub zwiększyć rząd filtra.

Kolejno dokonane zostało porównanie metod projektowania filtrów cyfrowych. Porównano metodę Yula-Walkera polegającą na minimalizacji błędu średniokwadratowego z metodą próbkowania w dziedzinie częstotliwości (fir2). Filtr yulewalk jest filtrem 8 rzędu dla 6 próbek w dziedzinie częstotliwości z kolei filtr fir2 dla tych samych próbek posiada 128 rząd. Kod oraz wyniki przedstawione zostały poniżej.

Tab. 6.5. Kod programu "Krupnik_Lab_6_5.m".

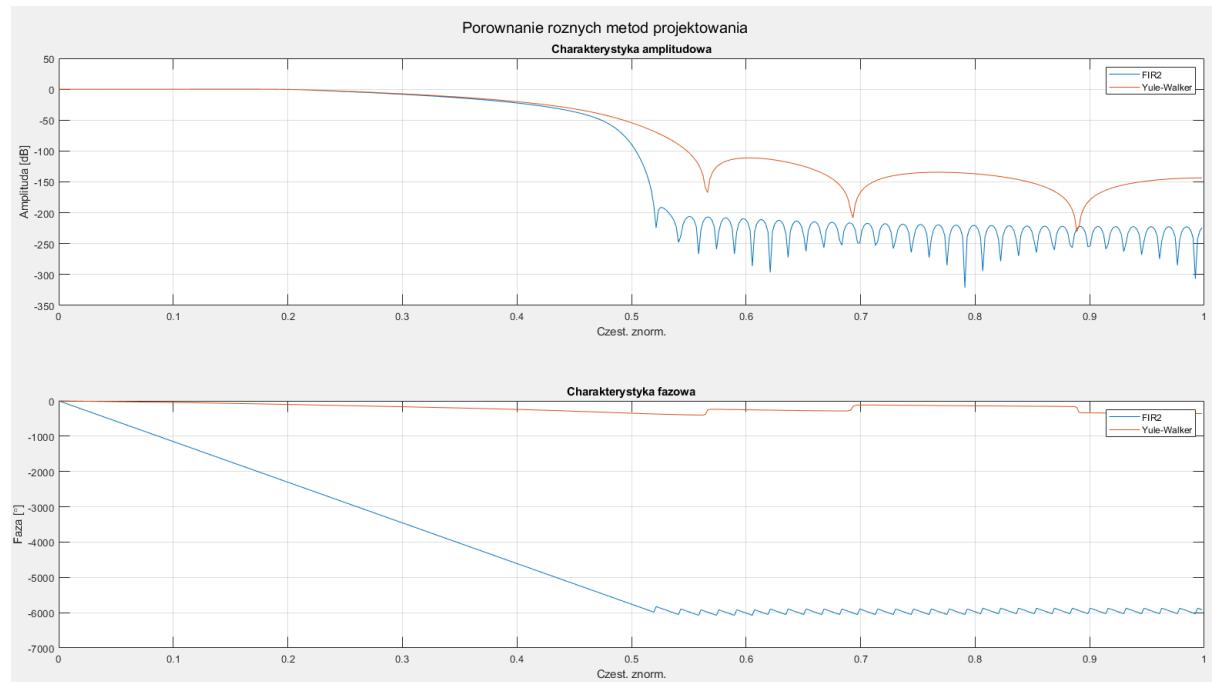
```
% Lab 6.
% Dla filtra FIR sprawdzić jego działanie.
%
% Mateusz Krupnik
clear all; close all; clc;
%% Sprawdzenie działania filtrów cyfrowych
% Dane sygnałów
f1=100; f2=250; f3=400; fs=1000;      % Częstotliwości składowych i Nyquista
A1=1; A2=0.8; A3=0.65;                  % Aplitudy składowych
t=0:(1/fs):1.023;                      % Wektor czasu
% Sygnał wymuszenia - składowa 3 harmonicznych
x=A1*sin(2*pi*f1*t)+A2*sin(2*pi*f2*t)+A3*sin(2*pi*f3*t);

%
% Parametry
fo1 = 300; fo2 = 200;                   % Częstotliwości odcięcia
wn1 = fo1*2/fs; wn2 = fo2*2/fs; % Znormalizowane częstotliwości
wn = [wn1, wn2]; fo = [fo1, fo2];
% Filtr Butter - dolnoprzepustowy
N = [2, 4 ,8]; rząd = 3;           % wybór rzędu filtra
%
% Projekt filtru cyfrowego i jego działanie
```

```
% Parametry, niektóre z poprzednich filtrów
format long e; % Typ formatowania zmiennych, 16 miejsc, wykładniczo
F_=[0 0.1 0.2 0.5 0.7 1];
M_=[1 1 1 0 0 0];
% Parametr rzędu z poprzednich sekcji
[l, m] = yulewalk(N(1, rząd), F_, M_); % Metoda Yule-Walkera
b = fir2(128, F_, M_); fir2(128, F_, M_); % Metoda próbkowania w d. częst.

[h, w] = freqz(b, 1);
Mag = 20*log10(abs(h)); Fi = phase(h)*180/pi; w = w/pi;
[h, w_] = freqz(l, m);
Mag_ = 20*log10(abs(h)); Fi_ = phase(h)*180/pi; w_ = w_/pi;

% Wykresy
figure(21);
sgtitle('Porównanie różnych metod projektowania');
subplot(211); plot(w, Mag, w_, Mag_);
legend('FIR2', 'Yule-Walker'); grid;
xlabel('Czest. znorm.'); ylabel('Amplituda [dB]');
title('Charakterystyka amplitudowa');
subplot(212); plot(w, Fi, w_, Fi_);
legend('FIR2', 'Yule-Walker'); grid;
xlabel('Czest. znorm.'); ylabel('Faza [\circ]');
title('Charakterystyka fazowa');
```

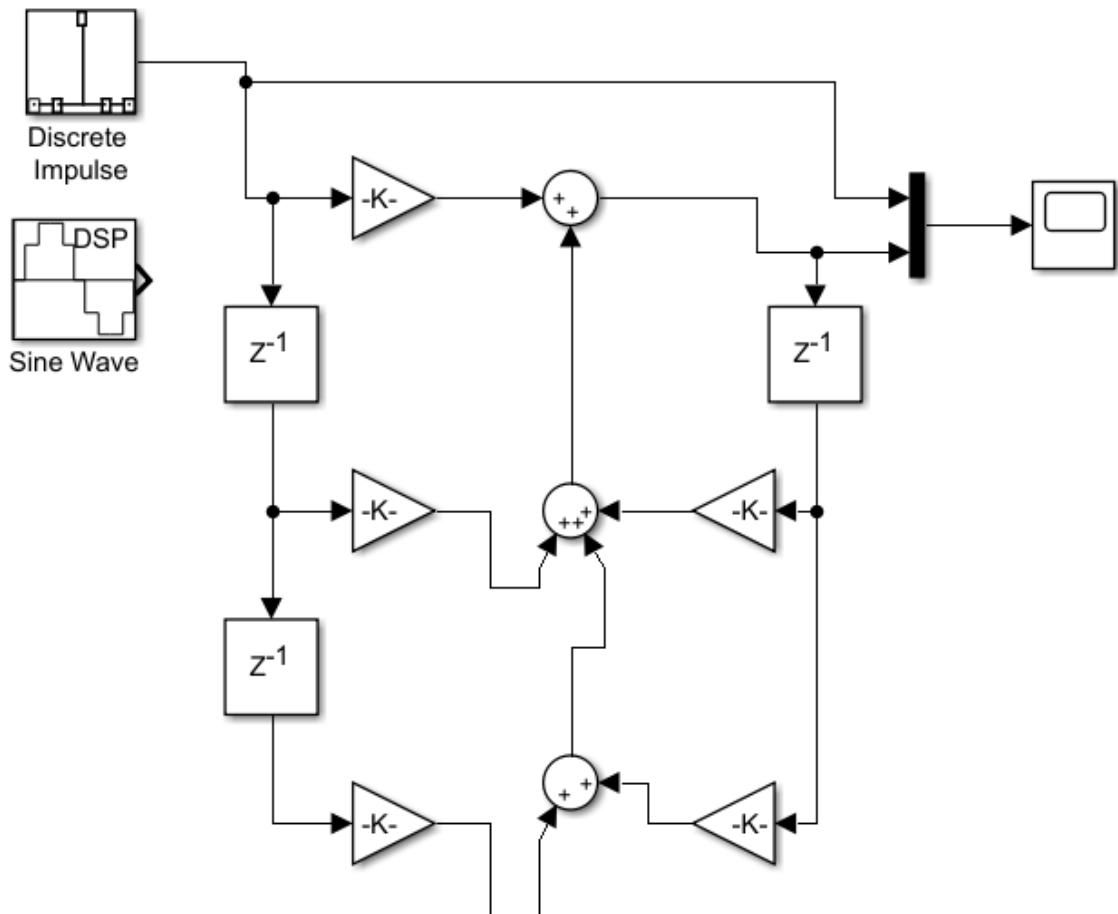


Rys. 6.21. Porównanie metod projektowania filtrów cyfrowych.

Jak można zauważyć filtry te posiadają o znaczące różnice w budowie. Filtr uzyskany metodą próbkowania w dziedzinie częstotliwości posiada o wiele większy rozmiar, ale jego faza jest liniowa w paśmie przepustowym. W paśmie zaporowym wykazuje się pulsacjami o stałym poziomie wysokości listków. Filtr Yule-Walkera z kolei mimo małych rozmiarów posiada podobną charakterystykę w zakresie przepustowym jednak jego pasmo przejściowe jest mniej strome, a odstęp amplitudy pasma

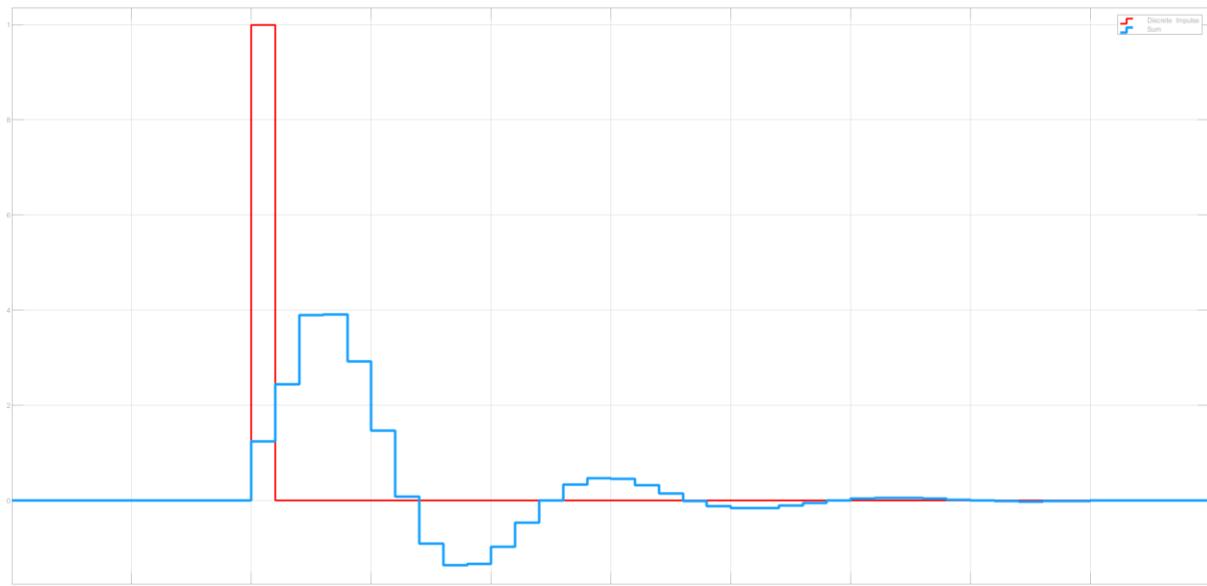
przejściowego i zaporowego jest mniejszy niż w przypadku metody FIR2 (należy pamiętać o skali logarytmicznej).

Na koniec stworzony został filtr cyfrowy w narzędziu Simulink. Schemat filtra rekursywnego pokazany jest na rysunku poniżej. Współczynnik dobrano według transmitancji czwartego analizowanego filtra.

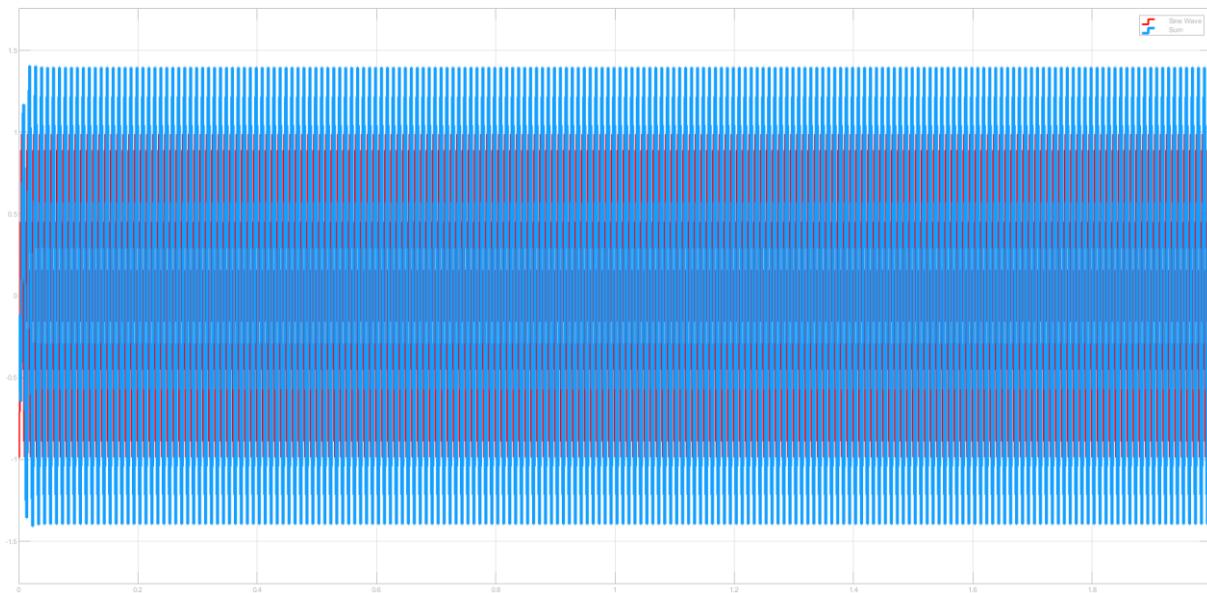


Rys. 6.22. Schemat blokowy filtru IIR. Plik „Krupnik_Lab_6_6_SIMULINK.slx”.

Następnie wyznaczono odpowiedź impulsową i filtrację cyfrowego sygnału sinusoidalnego.

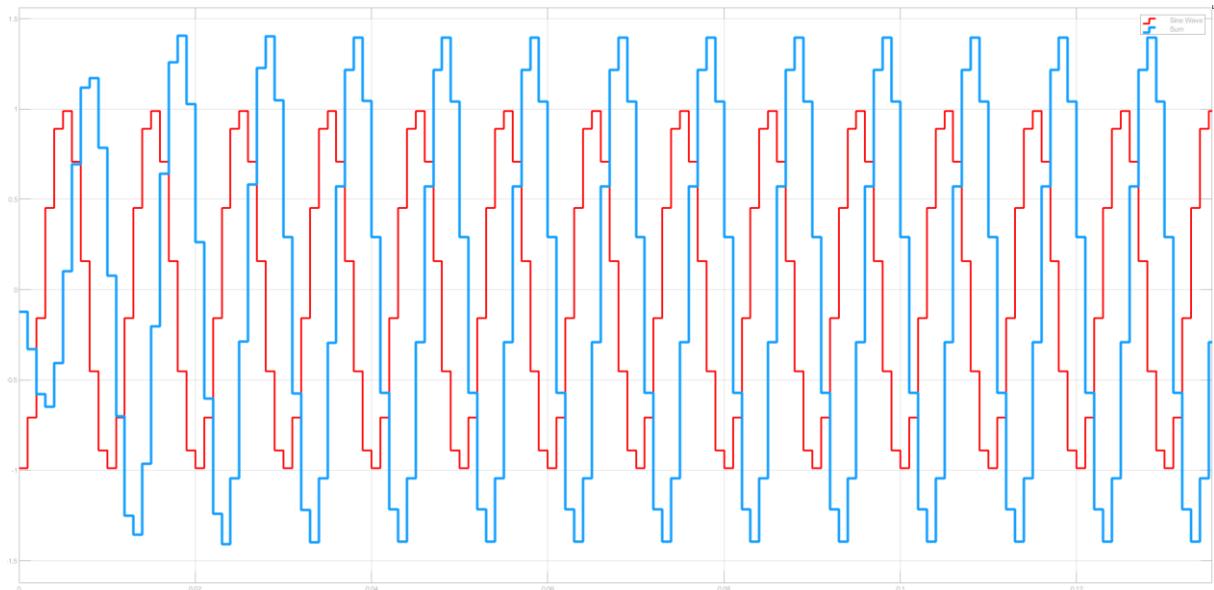


Rys. 6.23. Odpowiedź na impuls dyskretny.



Rys. 6.24. Filtracja sygnału sinusoidalnego.

Jak widać wyniki filtru zbudowanego w narzędziu Simulink w postaci blokowej o takich samych parametrach transmitancji jak ostatni analizowany układ daje identyczne wyniki. Widoczne jest zmniejszenie amplitudy oraz przesunięcie fazowe sygnału sinusoidalnego.



Rys. 6.25. Przybliżenie na początkowy fragment filtracji.

7. Laboratorium 7

Celem tych ćwiczeń było porównanie działania filtrów FIR oraz IIR. Wykorzystane zostały metody butter(rząd, częstotliwość_odcięcia), która służy do projektowania filtrów rekursywnych analogowych oraz cyfrowych a także metodę fir1(rząd, częstotliwość_odcięcia) która z kolei służy do projektowania cyfrowych filtrów nierekursywnych metodą okien. Podobnie jak w poprzednim ćwiczeniu wykorzystany zostanie sygnał o 3 składowych harmonicznych. Zaprojektowane filtry zostaną przedstawione w postaci charakterystyk częstotliwościowych oraz przebiegów czasowych. Jako pierwszy zaprojektowano filtr dolnoprzepustowy Butterwortha 32 rzędu i częstotliwości odcięcia 300 Hz. Następnie zaprojektowany został filtr nierekursywny 16 rzędu o tej samej częstotliwości odcięcia.

Tab. 7.1. Kod programu "Krupnik_Lab_7.m".

```
% Lab 7. Porównanie działania filtra rekursywnego i nierekursywnego.
% Metody fir1 i butter.
%                                         Mateusz Krupnik'
clc; clear all; close all;

% Dane sygnałów z lab. 6
f1=100; f2=250; f3=400; fs=1000;      % Czest. składowych i Nyquista
A1=1; A2=0.8; A3=0.65;                  % Aplitudy składowych
t=0:(1/fs):1.023;                      % Wektor czasu
% Sygnał wymuszenia - składowa 3 harmonicznych
x=A1*sin(2*pi*f1*t)+A2*sin(2*pi*f2*t)+A3*sin(2*pi*f3*t);
% Parametry odcięcia
fo1 = 300; fo2 = 200;                   % Czestotliwosci ociecia
wn1 = fo1*2/fs; wn2 = fo2*2/fs;        % Znormalizowane czestotliwosci

%% Filtr dolnoprzepustowy
% Filtr Butterwortha
rzad = 32;                            % rząd filtra
[B, A] = butter(rzad, wn1);           % dla częstotliwości odcięcia nr 1
x_filtered = filter(B, A, x);          % filtracja syg. przez układ
% Obliczenie widma i mocy za pomocą funkcji z lab 4.
```

```
[f_w, Moc, Wid] = fft_from_signal([x; x_filtered], fs);

% Wykresy
figure(1);
sgtitle(['Filtr dolnoprzepustowy Butterworth, rzad=' num2str(rzad) ...
    ', f_o=' num2str(f0)]);
subplot(211);
plot(t, x); title('Sygnał wymuszający');
xlabel('Czas [s]'); ylabel('Amp.'); grid;
subplot(212);
plot(t, x_filtered); title('Sygnał po filtracji');
xlabel('Czas [s]'); ylabel('Amp.'); grid;

figure(2);
sgtitle(['Filtr dolnoprzepustowy Butterworth, rzad=' num2str(rzad) ...
    ', f_o=' num2str(f0)]);
subplot(211);
plot(f_w, Wid(1,:)); title('Widmo sygnału wymuszającego');
xlabel('Częstotliwość [Hz]'); ylabel('Amp.'); grid;
subplot(212);
plot(f_w, Wid(2,:)); title('Widmo po filtracji');
xlabel('Częstotliwość [Hz]'); ylabel('Amp.'); grid;

% Filtr metoda probkowania w dziedzinie częstotliwości
% filtr nierekursywny
rzad = 16; % rzad filtra
b1 = fir1(rzad, wn1);
x_fir2 = filter(b1, 1, x);
% Obliczenie widma i mocy za pomocą funkcji z lab 4.
[f_w, Moc, Wid] = fft_from_signal([x; x_fir2], fs);

% Wykresy
figure(3);
sgtitle(['Filtr dolnoprzepustowy FIR2, rzad=' num2str(rzad) ...
    ', f_o=' num2str(f0)]);
subplot(211);
plot(t, x); title('Sygnał wymuszający');
xlabel('Czas [s]'); ylabel('Amp.'); grid;
subplot(212);
plot(t, x_fir2); title('Sygnał po filtracji');
xlabel('Czas [s]'); ylabel('Amp.'); grid;

figure(4);
sgtitle(['Filtr dolnoprzepustowy FIR2, rzad=' num2str(rzad) ...
    ', f_o=' num2str(f0)]);
subplot(211);
plot(f_w, Wid(1,:)); title('Widmo sygnału wymuszającego');
xlabel('Częstotliwość [Hz]'); ylabel('Amp.'); grid;
subplot(212);
plot(f_w, Wid(2,:)); title('Widmo po filtracji');
xlabel('Częstotliwość [Hz]'); ylabel('Amp.'); grid;

% Wykresy porównawcze filtracji
figure(5);
spectrogram(x, 'yaxis'); title('Sygnał wymuszający');
figure(6);
spectrogram(x_filtered, 'yaxis'); title('Sygnał po filtracji - IIR');
figure(7);
spectrogram(x_fir2, 'yaxis'); title('Sygnał po filtracji - FIR');
```

```

% Filtrowanie i krótkoczasowa analiza częstotliwościowa
% sygnałów o zmiennej częstotliwości
T = 0:(1/fs):1.023;
% chirp(wektor czasu, częstotliwość start, koniec narastania,
% częstotliwość docelowa)
X = chirp(T, 50, 1.023, 450); % funkcja generująca sygnał
% filtracja przez filtry FIR i IIR
X_fir2 = filter(b1, 1, X);
X_iir = filter(B, A, X);

figure(8)
subplot(311); plot(T, X);
title('Sygnał wymuszający'); xlabel('Czas [s]'); ylabel('Amp.');
```

```
grid;
```

```
subplot(312); plot(T, X_fir2);
title('Filtracja FIR'); xlabel('Czas [s]'); ylabel('Amp.');
```

```
grid;
```

```
subplot(313); plot(T, X_iir);
title('Filtracja IIR'); xlabel('Czas [s]'); ylabel('Amp.');
```

```
grid;
```

```
figure(9)
% SFFT od sygnału X, z oknem 256, zakładką okien 250, i 256 punktową FFT
spectrogram(X, 256, 250, 256, 1E3, 'yaxis');
title('Sygnał wymuszający');
```

```
figure(10)
spectrogram(X_fir2, 256, 250, 256, 1E3, 'yaxis');
```

```
title('Sygnał po filtracji FIR');
```

```
figure(11)
spectrogram(X_iir, 256, 250, 256, 1E3, 'yaxis');
```

```
title('Sygnał po filtracji IIR');
```

```
%% Filtr średkowoprzepustowy
```

```
% Filtr Butterwortha
```

```
rzad1 = 8; % rzad filtra
```

```
[B, A] = butter(rzad1, [wn2 wn1]); % dla częstotliwości odciecia nr 1
x_filtered = filter(B, A, x); % filtracja syg. przez układ
```

```
% Filtr metoda probkowania w dziedzinie częstotliwości
```

```
% filtr nierekursywny
```

```
rzad2 = 16; % rzad filtra
```

```
b1 = fir1(rzad2, [wn2 wn1]);
```

```
x_fir2 = filter(b1, 1, x);
```

```
% Obliczenie widma i mocy za pomocą funkcji z lab 4.
```

```
[f_w, Moc, Wid] = fft_from_signal([x; x_filtered; x_fir2], fs);
```

```
% Wykresy
```

```
figure(12);
```

```
sgtitle(['Filtr średkowoprzepustowy f_o=[ num2str(f01) ...
', ' num2str(f02) ']));
```

```
subplot(311);
```

```
plot(t, x); title('Sygnał wymuszający');
```

```
xlabel('Czas [s]'); ylabel('Amp.');
```

```
grid;
```

```
subplot(312);
```

```
plot(t, x_filtered);
```

```

title(['Sygnał po filtracji, Butterworth rzad=' num2str(rzad1)]);
xlabel('Czas [s]'); ylabel('Amp.); grid;
subplot(313);
plot(t, x_fir2);
title(['Sygnał po filtracji, FIR2 rzad=' num2str(rzad2)]);
xlabel('Czas [s]'); ylabel('Amp.); grid;

figure(14);
sgtitle(['Filtr srodkowoprzepustowy f_o=[' num2str(f0) ...
    ', ' num2str(fo2) ']]);
subplot(311);
plot(f_w, Wid(1,:)); title('Widmo sygnału wymuszającego');
xlabel('Czestotliwosc [Hz]'); ylabel('Amp.); grid;
subplot(312);
plot(f_w, Wid(2,:));
title(['Widmo po filtracji, Butterworth rzad=' num2str(rzad2)]);
xlabel('Czestotliwosc [Hz]'); ylabel('Amp.); grid;
subplot(313);
plot(f_w, Wid(3,:));
title(['Widmo po filtracji, FIR2 rzad=' num2str(rzad2)]);
xlabel('Czestotliwosc [Hz]'); ylabel('Amp.); grid;

% Wykresy porównawcze filtracji
figure(15);
spectrogram(x, 'yaxis'); title('Sygnał wymuszający');
figure(16);
spectrogram(x_filtered, 'yaxis'); title('Sygnał po filtracji - IIR');
figure(17);
spectrogram(x_fir2, 'yaxis'); title('Sygnał po filtracji - FIR');

% Filtrowanie i krótkoczasowa analiza częstotliwościowa
% sygnałów o zmiennej częstotliwości
T = 0:(1/fs):1.023;
% chirp(wektor czasu, częstotliwość start, koniec narastania,
% częstotliwość docelowa)
X = chirp(T,50,1.023,450); % funkcja generująca sygnał
% filtracja przez filtry FIR i IIR
X_fir2 = filter(b1, 1, X);
X_iir = filter(B, A, X);

figure(18)
subplot(311); plot(T, X);
title('Sygnał wymuszający'); xlabel('Czas [s]'); ylabel('Amp.); grid;

subplot(312); plot(T, X_fir2);
title('Filtracja FIR'); xlabel('Czas [s]'); ylabel('Amp.); grid;

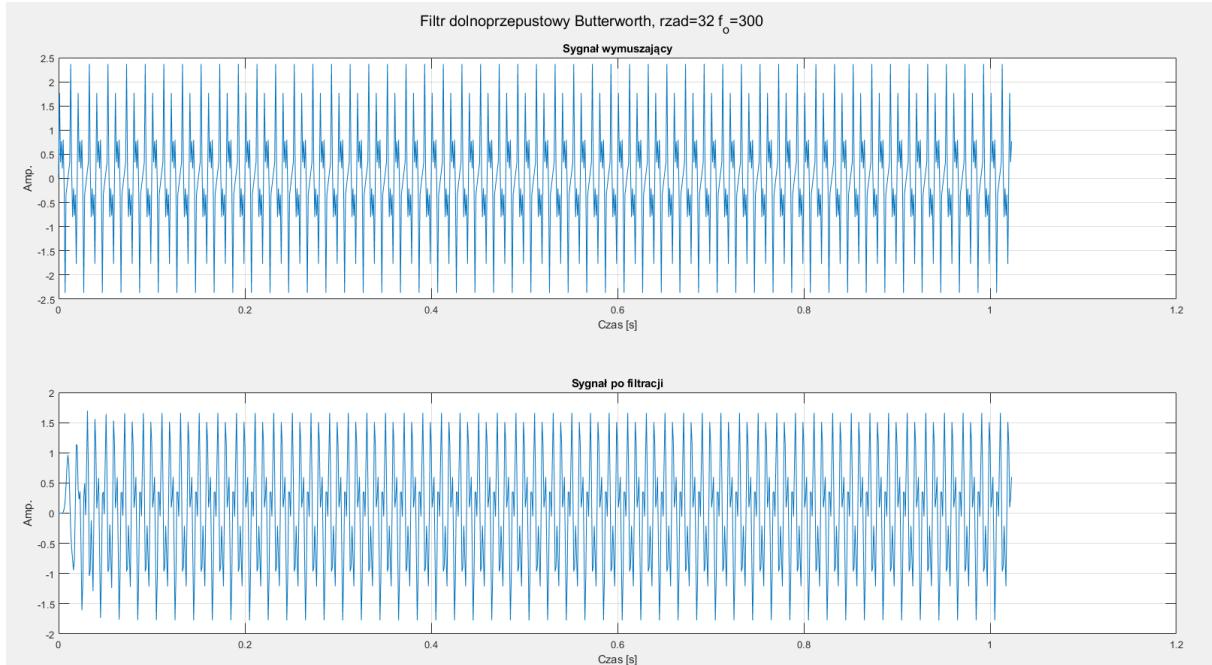
subplot(313); plot(T, X_iir);
title('Filtracja IIR'); xlabel('Czas [s]'); ylabel('Amp.); grid;

figure(19)
% SFFT od sygnału X, z oknem 256, zakładką okien 250, i 256 punktową FFT
spectrogram(X, 256, 250, 256, 1E3, 'yaxis'); title('Sygnał wymuszający');

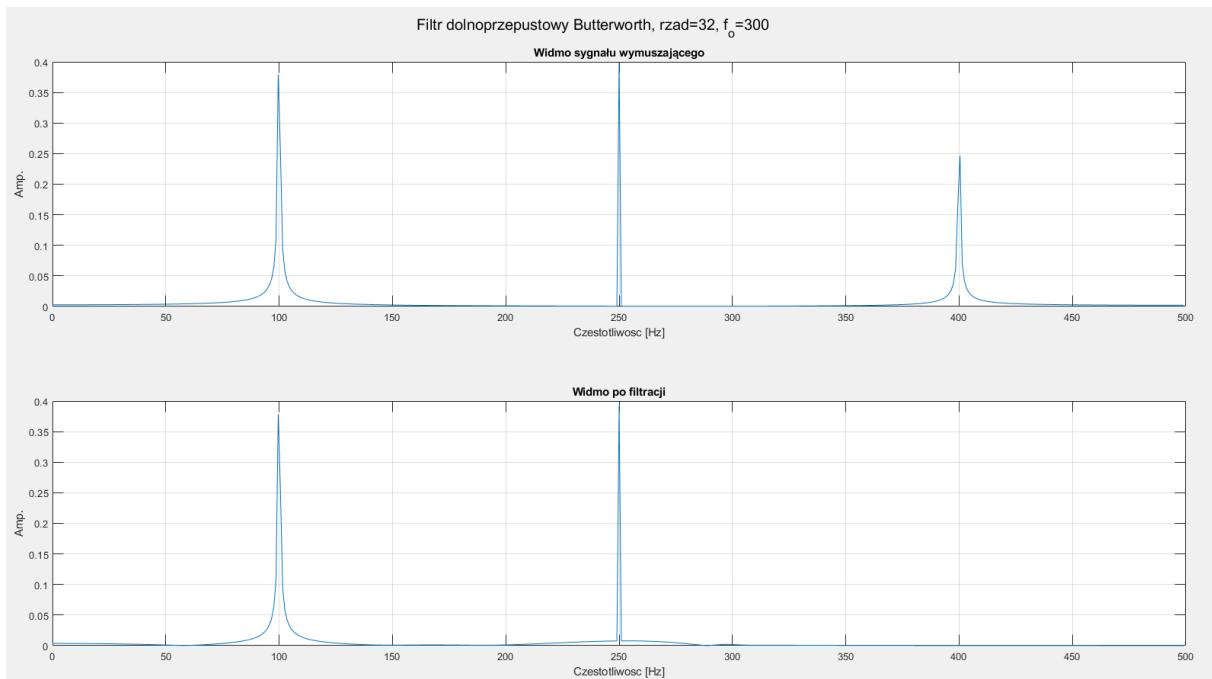
figure(20)
spectrogram(X_fir2, 256, 250, 256, 1E3, 'yaxis');
title('Sygnał po filtracji FIR');

```

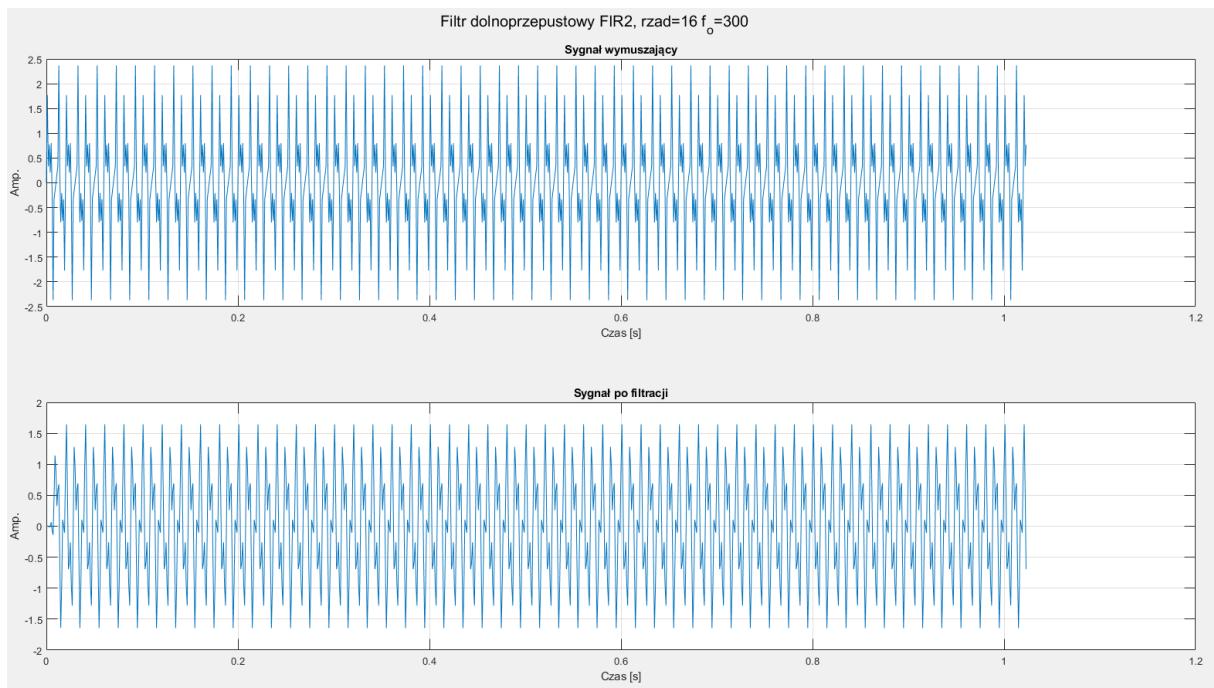
```
figure(21)
spectrogram(X_iir, 256, 250, 256, 1E3, 'yaxis');
title('Sygnał po fitlracji IIR');
```



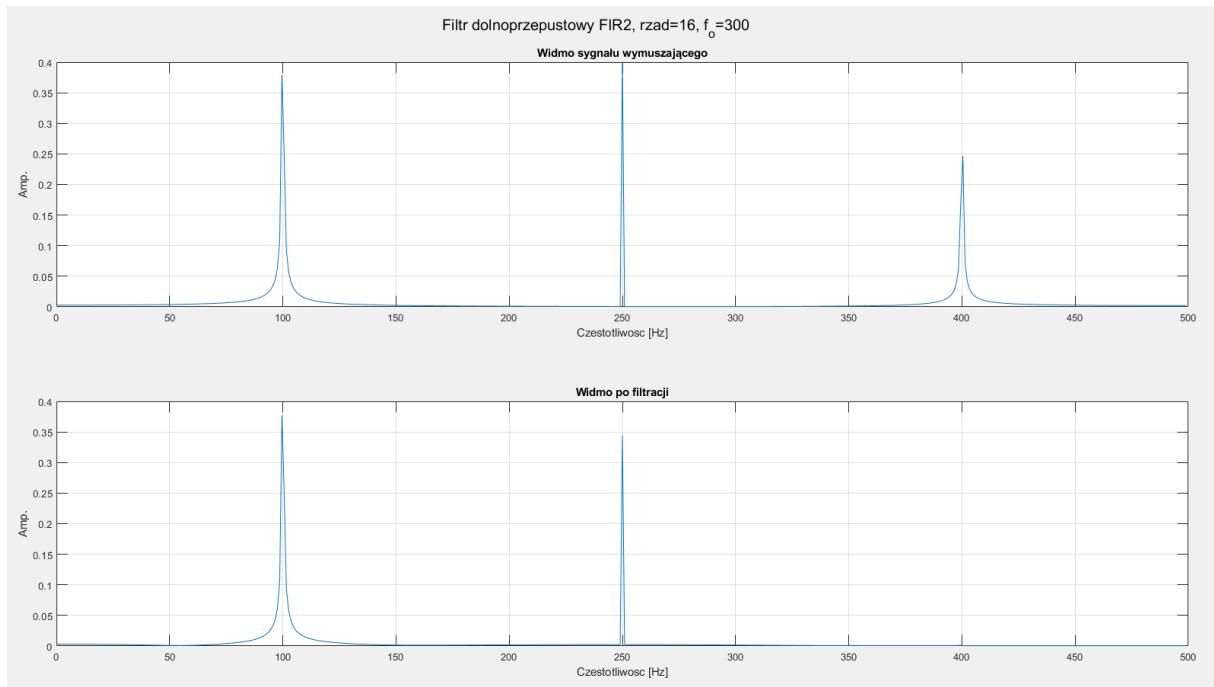
Rysunek 7.1. Sygnał wymuszenia oraz po filtracji. Zastosowano filtr Butterwortha 32 rzędu, częstotliwość odcięcia 300Hz.



Rysunek 7.2. Widmo sygnału wymuszającego oraz po filtracji filtrem Butterwortha.

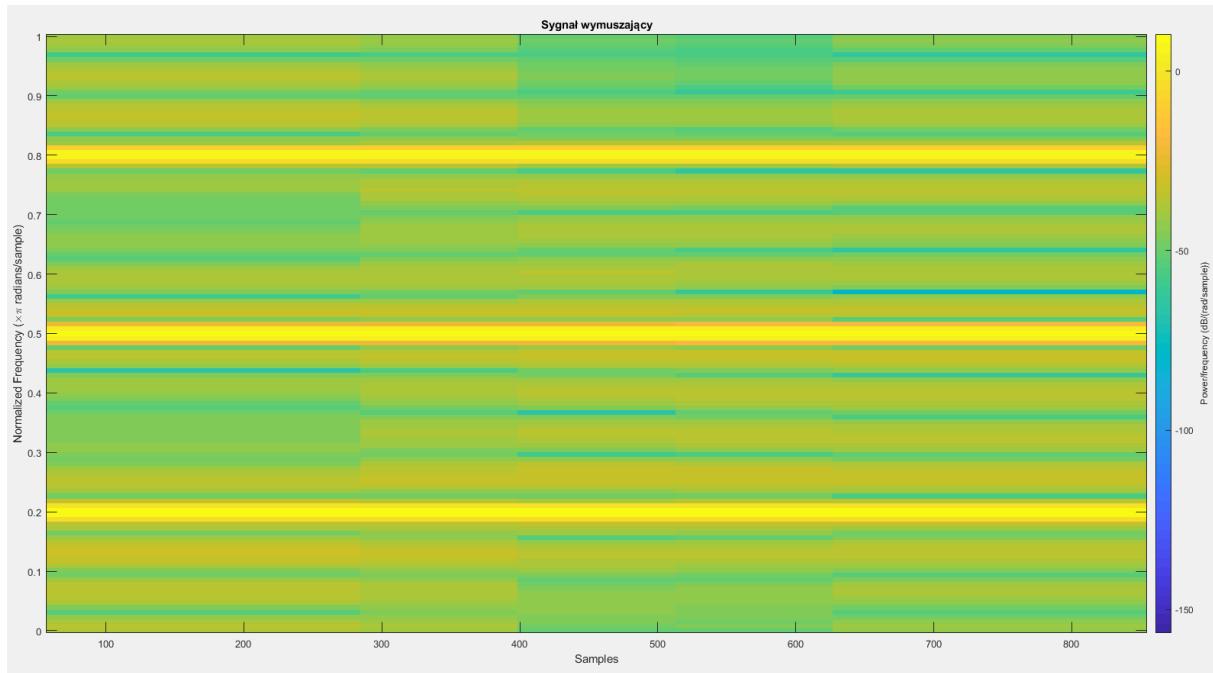


Rysunek 7.3. Sygnał wymuszenia oraz po filtracji. Zastosowano filtr cyfrowy fir1 16 rzędu, częstotliwość odcięcia 300Hz.

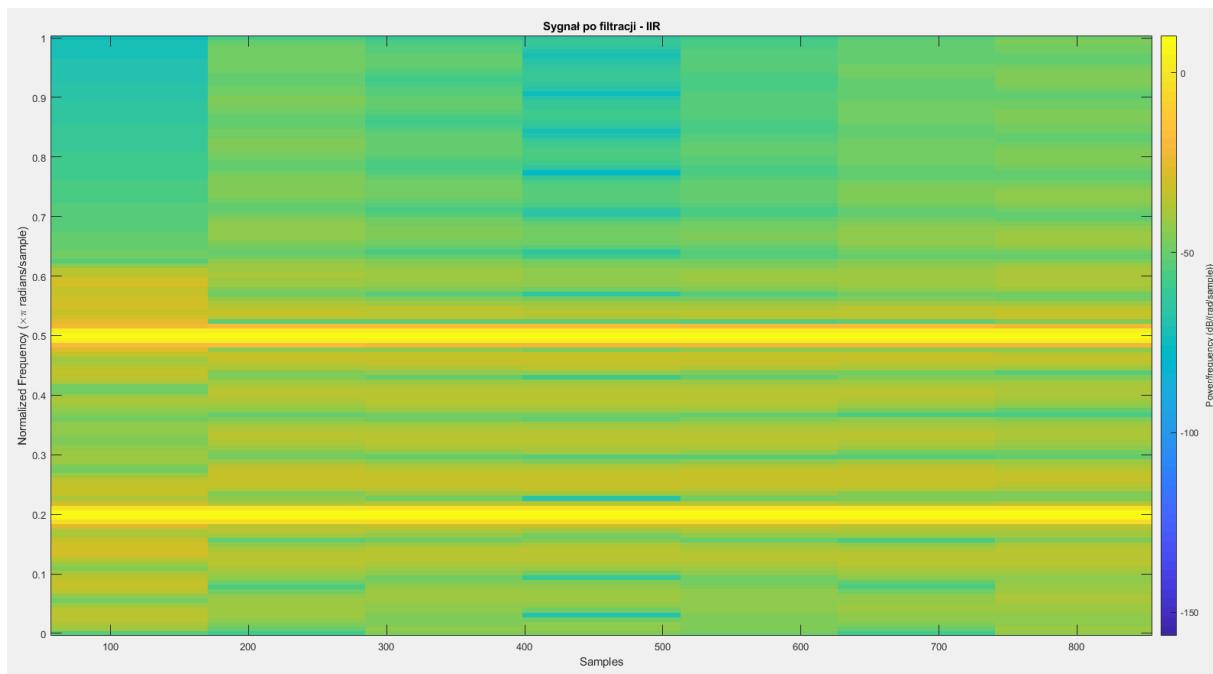


Rysunek 7.4. Widmo sygnału wymuszającego i po filtracji filtrem cyfrowym nerekursywnym.

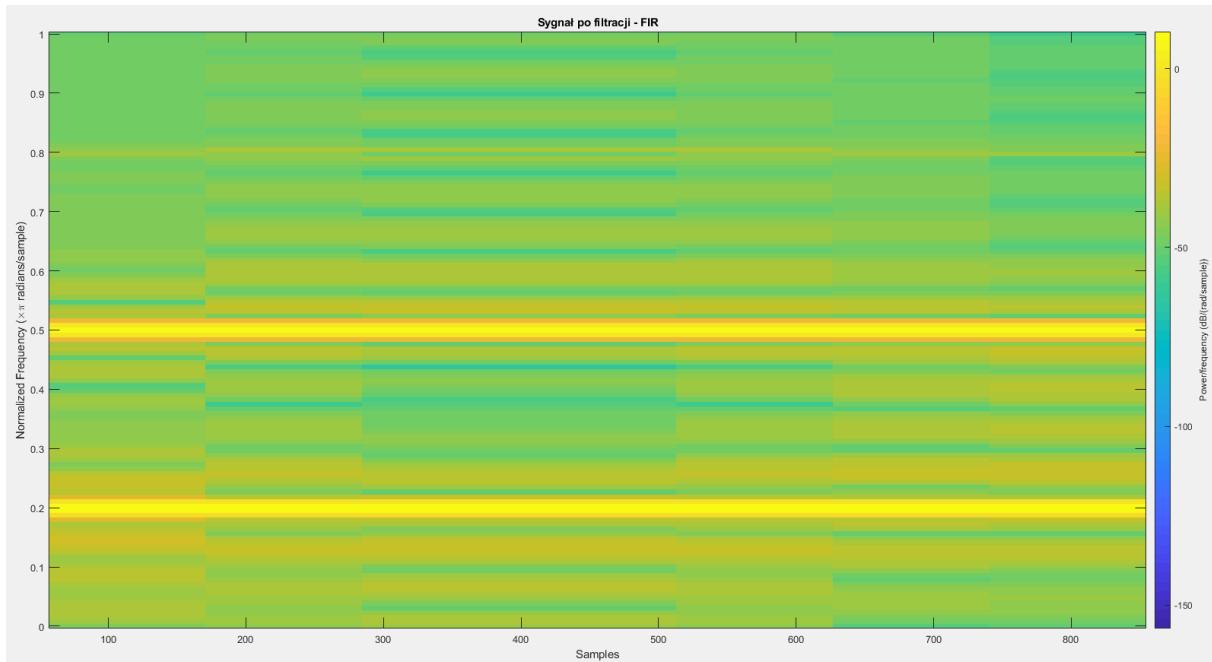
W przypadku analizy częstotliwościowej widoczne jest usunięcie wszystkich składowych powyżej częstotliwości odcięcia. Przeprowadzona jednak została analiza krótko-czasową za pomocą polecenia spectrogram, które zwraca charakterystykę czasowo-częstotliwościową.



Rysunek 7.5. Spektrogram sygnału wymuszającego, widoczne 3 składowe harmoniczne.

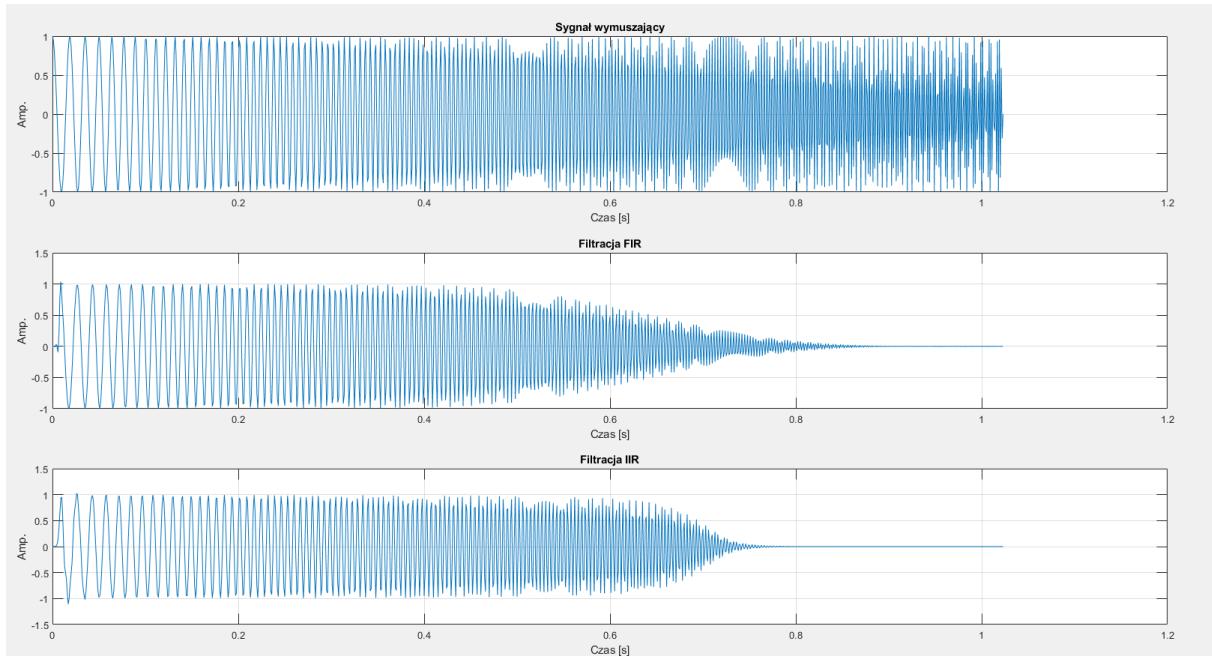


Rysunek 7.6. Spektrogram po filtracji filtrem IIR (Butterwortha).



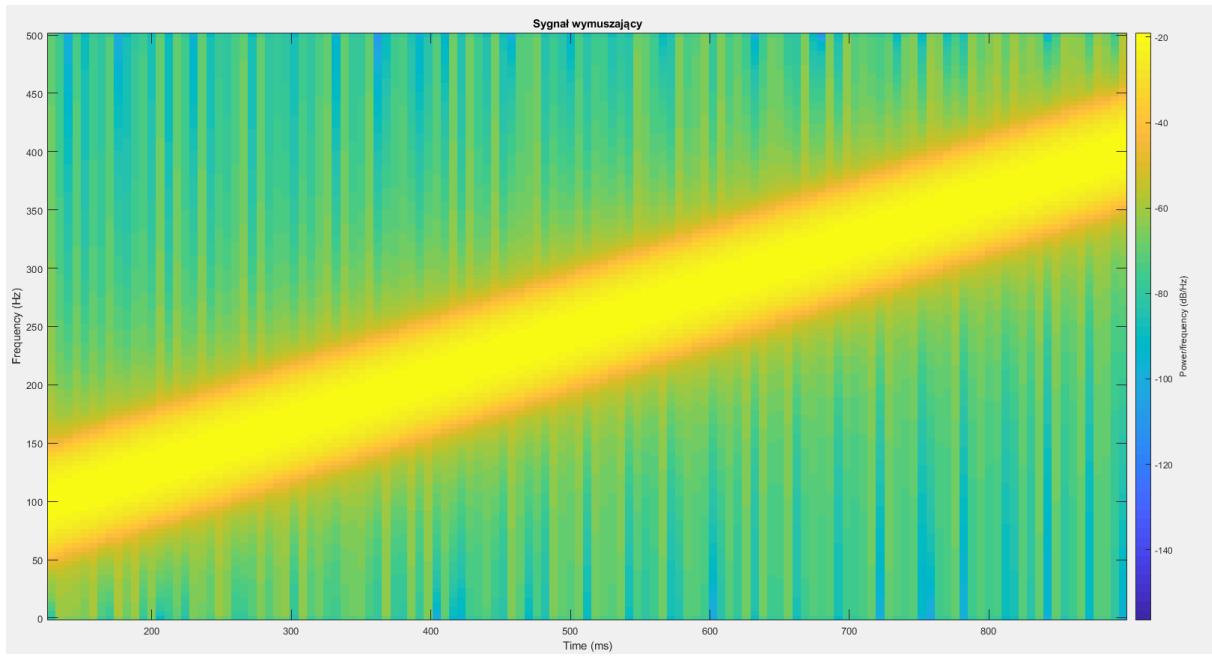
Rysunek 7.7. Spektrogram po filtracji filtrem FIR (fir1).

Jak można zauważyć dla sygnałów o stałej zawartości częstotliwościowej oba filtry poradziły sobie z usunięciem składowej powyżej częstotliwości odcięcia. Jednak filtr cyfrowy potrzebował dwukrotnie większy rząd. W celu sprawdzenia filtrów na sygnałach zmodulowanych dokonano analizy sygnału z modulacją. Sygnał został stworzony z użyciem metody chirp która zwraca sygnał o liniowej zmianie częstotliwości od zadanego dolnego progu do górnego w określonym czasie trwania.



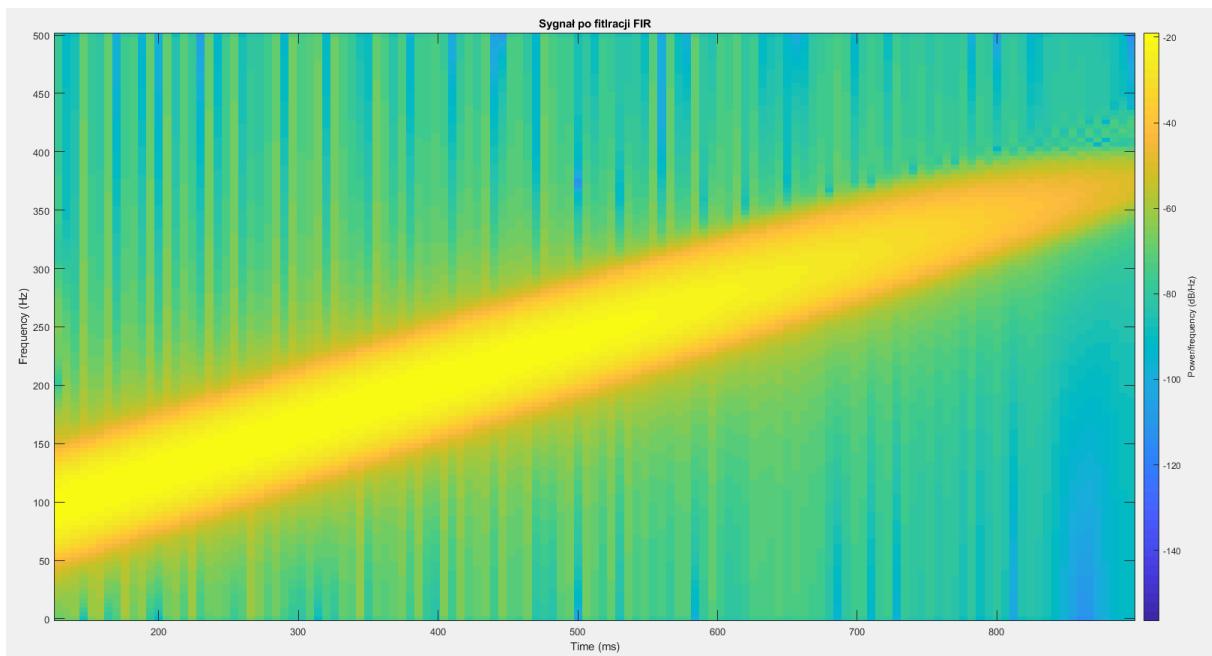
Rysunek 7.8. Przebiegi czasowe sygnału zmodulowanego i po filtracji odpowiednio filtrem nierekursywnym i rekursywnym.

Jak można zauważyć filtr rekursywny o wiele szybciej wytłumił amplitudę sygnału, gdy ten przekroczył częstotliwość odcięcia.

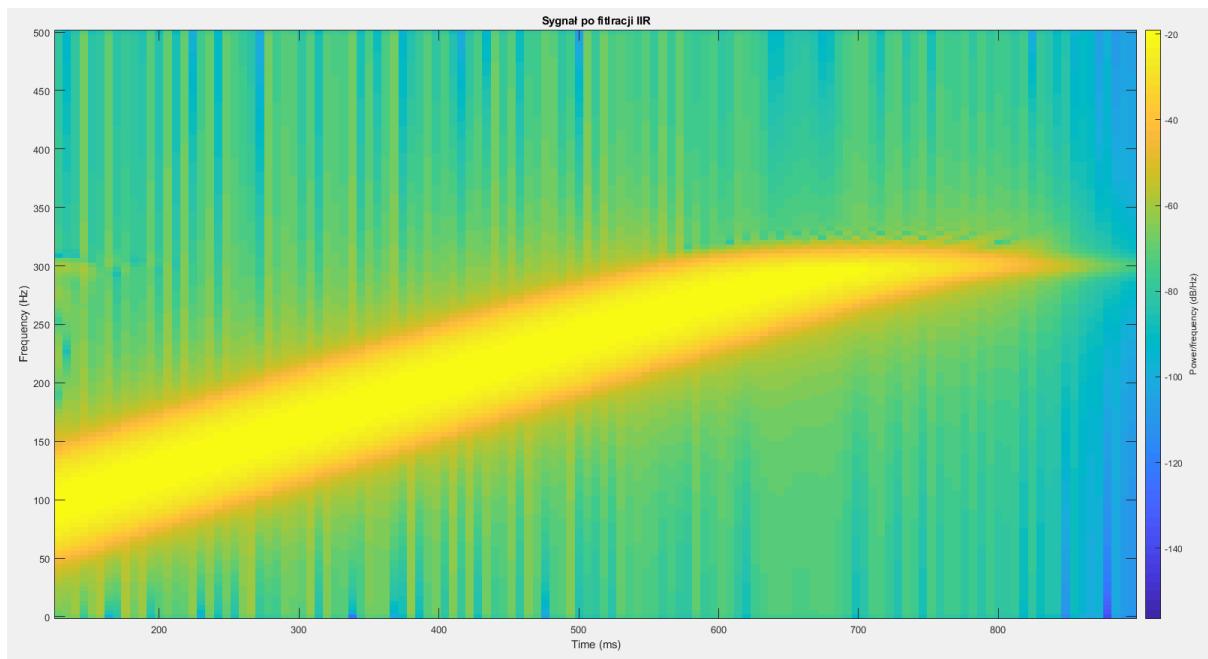


Rysunek 7.9. Spektrogram sygnału wymuszającego.

Widoczna jest linowa zmiana częstotliwości sygnału z czasem.



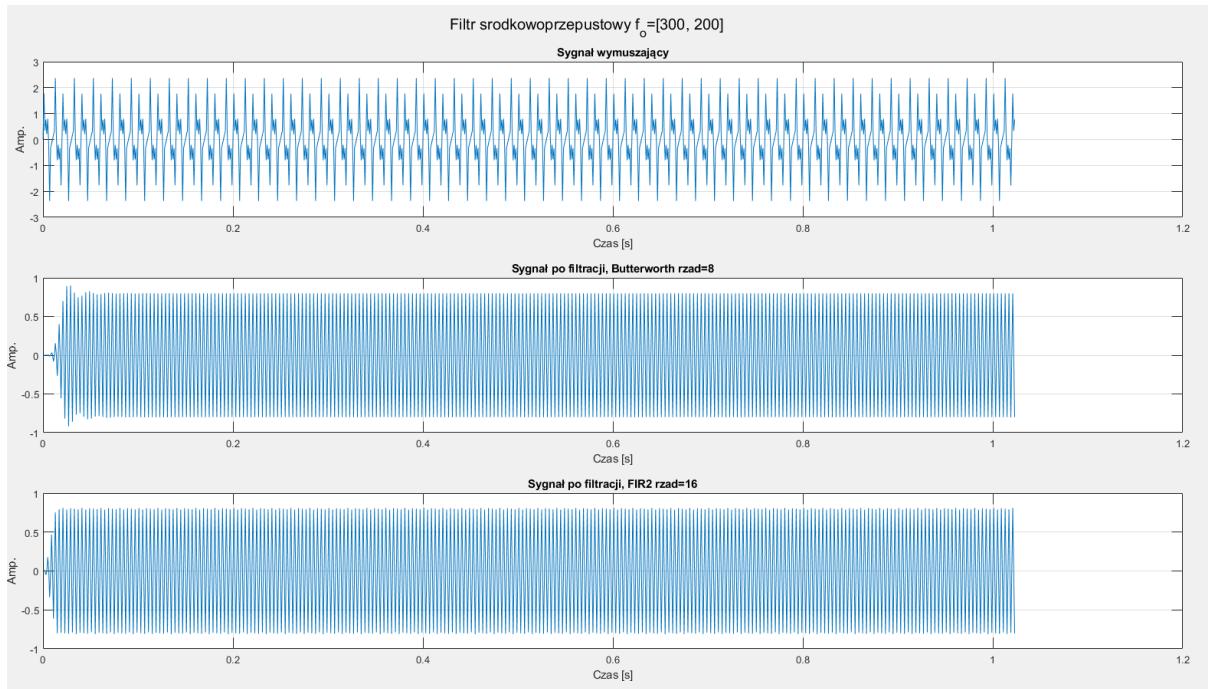
Rysunek 7.10. Spektrogram sygnału po filtracji filtrem nierekursywnym.



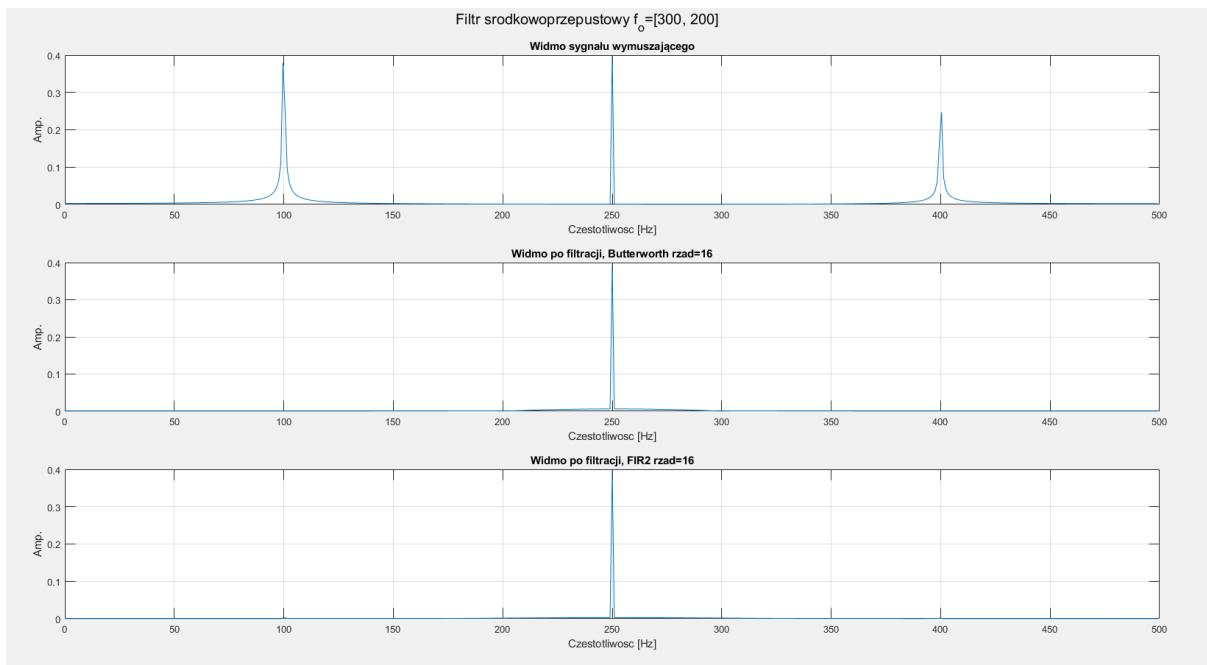
Rysunek 7.11. Spektrogram sygnału po filtracji filtrem rekursywnym.

Rysunki 8.10 i 8.11 pokazują jak szybko filtr wyciął z sygnału składowe przekraczające zadaną częstotliwość odcięcia. Widać, że filtr nierekursywny potrzebuje dłuższego czasu, aby wytłumić sygnał.

Taką samą analizę przeprowadzono z wykorzystaniem filtra średkowoprzepustowego o częstotliwościach odcięcia 200 i 300 Hz. W przypadku filtra rekursywnego zastosowany zastał filtr 8 rzędu a dla filtra nierekursywnego 16 rzędu.

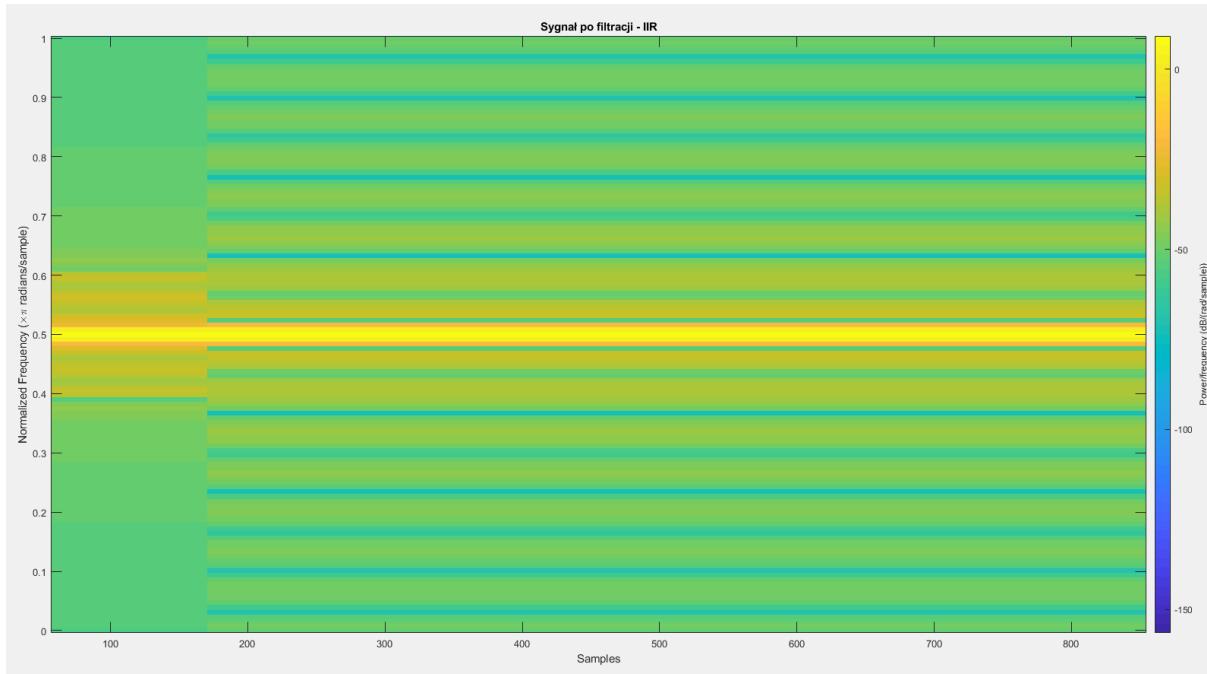


Rysunek 7.12. Przebiegi czasowe sygnałów: wymuszający, po filtracji filtrem rekursywnym, po filtracji filtrem nierekursywnym.

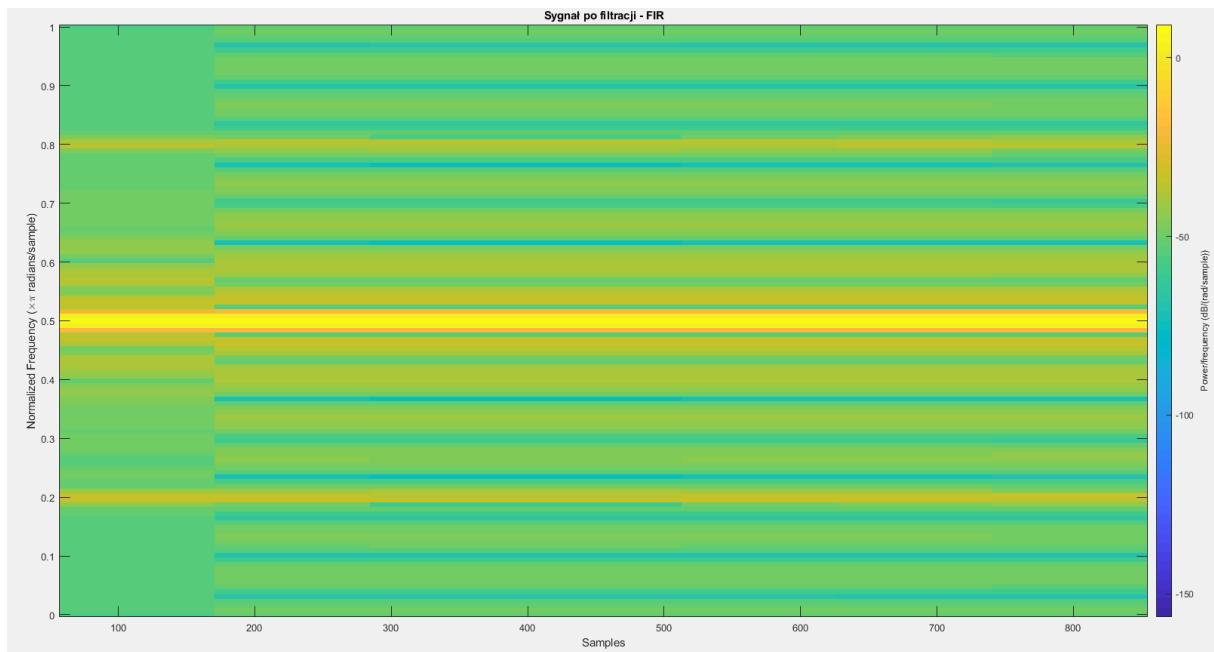


Rysunek 7.13. Widma sygnałów w kolejności jak na rysunku powyżej.

Spektrogram sygnału wymuszającego przedstawiony jest już na rysunku 8.5. Poniżej przedstawione zostaną spektrogramy po filtracji oraz filtracja sygnału zmodulowanego, którego spektrogram widoczny jest na rysunku 8.9.

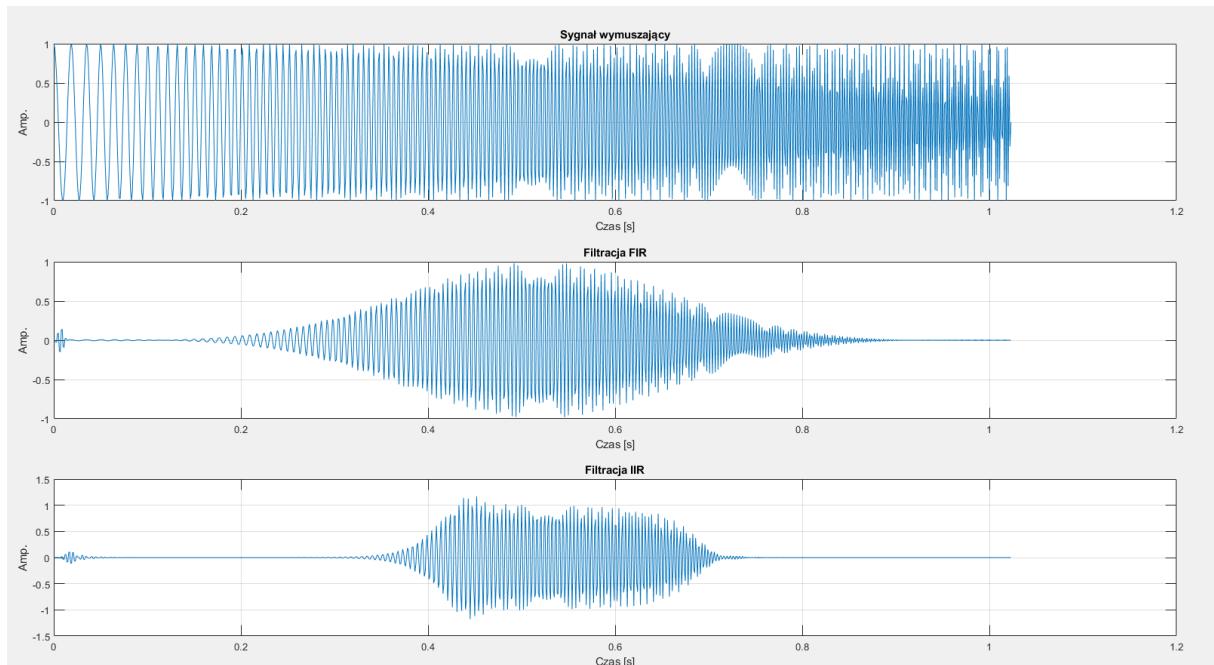


Rysunek 7.14. Spektrogram po filtracji filtrem rekursywnym.

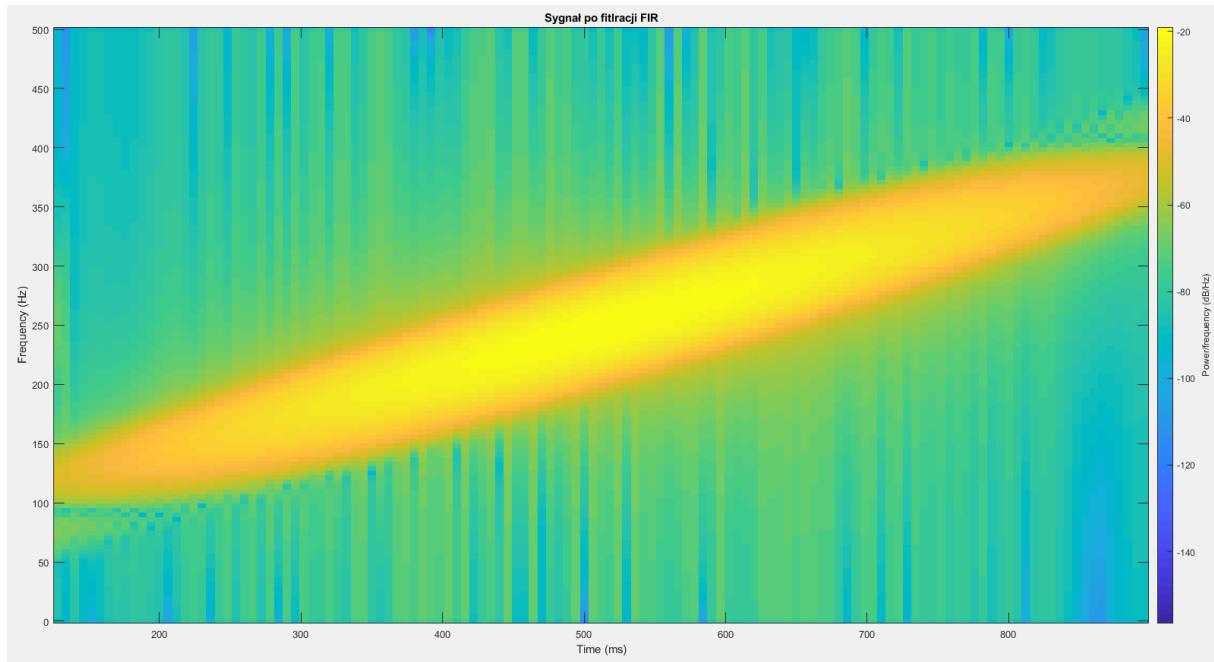


Rysunek 7.15. Spektrogram po filtracji filtrem nierekursywnym.

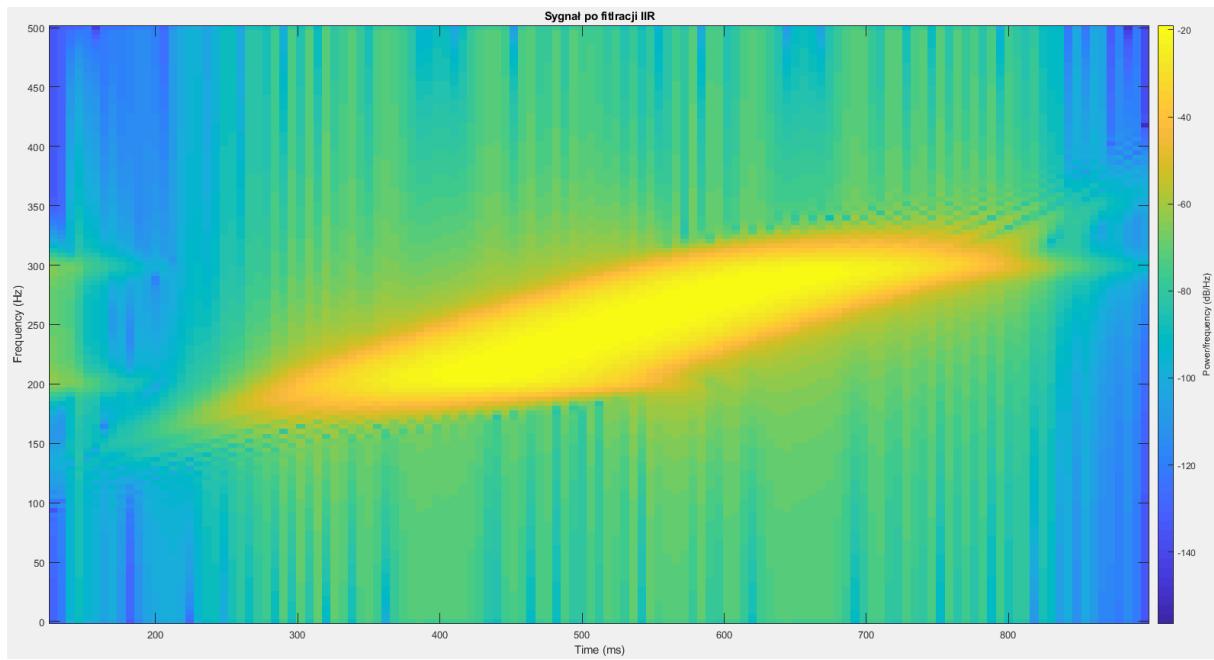
W przypadku filtra FIR widoczne są pozostałości składowych poza pasmem przepustowym. W dalszej części rozdziału widoczne są charakterystyki czasowo-częstotliwościowe sygnału zmodulowanego po filtracji. Ponownie filtr rekursywny wykazuje się słabszym i wolniejszym działaniem od filtra nierekursywnego, który posiada dwukrotnie mniejszy rząd. Sprzężenie zwrotne filtra rekursywnego znacznie polepsza jego działanie, jednak należy zwrócić uwagę na stabilność tych filtrów.



Rysunek 7.16. Przebiegi czasowe sygnału zmodulowanego i po filtracji śródkowo-przepustowej odpowiednio filtrem nierekursywnym i rekursywnym.



Rysunek 7.17. Spektrogram sygnału po filtracji średkowo-przepustowej filtrem nerekursywnym.



Rysunek 7.18. Spektrogram sygnału po filtracji średkowo-przepustowej filtrem rekursywnym.

8. Kody programów dodatkowych.

W tym rozdziale przedstawione są kody programów zrealizowanych według przykładów z książki prof. Zielińskiego „Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań”.

Tab. 8.1. Kod programów na podstawie rozdziału 1.

```
% Rozdzial 1 Zielinski
% 1. Wygeneruj N=1000 próbek sygnału sinusoidalnego
% x(t)=Asin(2?fxt) o amplitudzie A=5 i o częstotliwości fx=10 Hz,
% spróbkowanego z częstotliwością fp=1000 Hz. Narysuj ten sygnał.
%
% Mateusz Krupnik
disp('1. Wygeneruj N=1000 próbek sygnału sinusoidalnego x(t)=Asin(2fxt)')
disp(' o amplitudzie A=5 i o częstotliwości fx=10 Hz, spróbkowanego z')
disp(' częstotliwością fp=1000 Hz. Narysuj ten sygnał.')
N=1000; A=5; fx=10; fp=1000; % parametry sygnalu
dt = 1/fp; % okres próbkiowania
t = dt*(0:N-1); % wektor czasu
x = A*sin(2*pi*fx*t); % sygnał
plot(t, x); grid; title('Sygnał x(t)'); xlabel('Czas [s]');

%% Obliczenie parametrów sygnału
x_sred = mean(x), x_sred2 = sum(x)/N % wartości średnie
x_max = max(x), x_min = min(x), x_std1 = std(x) % max, min, odch. stand.
x_std2 = sqrt(sum((x-mean(x)).^2) / (N-1)) % odch. stand.
x_eng = dt*sum(x.^2) % energia sygnału
x_moc = 1/N * x_eng % moc sygnału
x_skut = sqrt(x_moc) % wartości skuteczne

%% Obliczenie korelacji R1, R2 i R3 sygnału
R1 = xcorr(x); % korelacja nie umormowana
R2 = xcorr(x, 'biased'); % unormowana przez długość /N
R3 = xcorr(x, 'unbiased'); % unormowana przez / (N-abs(k))
tR = [-fliplr(t) t(2:N)]; R = [R1; R2; R3];
% Generowanie wykresów
for i=1:3
    subplot(3, 1, i);
    plot(tR, R(i, :)); grid; title('Autokorelacja');
end

%% Wyznaczenie autokorelacji samemu
R_w = zeros(size(x));
for k=0:N-1
    R_w(k+1) = sum( x(1:N-k).*x(1+k:N) ) / (N-k);
end
R_w = [fliplr(R_w) R_w(2:N)];
% Generowanie wykresów
figure();
plot(tR, R_w); grid; title('Autokorelacja własna');

%% obliczenia współczynników szeregu Fouriera sygnału
X = fft(x); % szybka dyskretna transf Fouriera
df = 1/(N*dt); % częstotliwość podstawowa f0=df=1/T
f = df * (0:N-1);
% Generowanie wykresów
subplot(3,1,1); plot(f, real(X)); grid; title('Real(X)');
subplot(3,1,2); plot(f, imag(X)); grid; title('Imag(X)');
subplot(3,1,3); plot(f, abs(X)); grid; title('Abs(X)');
figure();
plot(f(1:N/2+1), abs(X(1:N/2+1))/(N/2)); grid; title('Po wyskalowaniu');
```

```

%% Zsyntezuj sygnał na podstawie współczynników szeregu i porównaj z
% orginalem
% Odwrotna transforamcja Fouriera
xs = ifft(X);
% Generowanie wykresów
figure();
plot(t, real(x-xs));
grid; title('Różnica pomiędzy sygnałem x(t) a xs(t)');

%% 2. Wygeneruj N=1000 próbek szumu o rozkładzie równomiernym
% i normalnym (gaussowskim). Wyznacz dla nich
% funkcję autokorelacji, autokowariancji, histogram,
% szereg Fouriera i periodogram.

disp('2. Wygeneruj N=1000 próbek szumu o rozkładzie równomiernym i')
disp(' normalnym (gaussowskim). Wyznacz dla nich funkcję autokorelacji,')
disp(' autokowariancji, histogram, szereg Fouriera i periodogram.')

% Sygnały losowe, rozkład równomierny i normalny
s1 = rand(1, N); s2=randn(1, N);
% Obliczenie autokorelacji
R1 = xcorr(s1, 'unbiased'); R2 = xcorr(s2, 'unbiased');
% Generowanie wykresów
figure();
subplot(2,1,1); plot(tR, R1); grid; title('Autokorelacja równomierny');
subplot(2,1,2); plot(tR, R2); grid; title('Autokorelacja normlany');

% Obliczenie autokowariancji
C1 = xcov(s1); C2 = xcov(s2);
% Generowanie wykresów
figure();
subplot(2,1,1); plot(tR, C1); grid; title('Auto kowariancja rownomierny');
subplot(2,1,2); plot(tR, C2); grid; title('Auto kowariancja normlany');

% Histogram rozkładów
Hs1 = hist(s1, 100); Hs2 = hist(s2, 100);
% Generowanie wykresów
figure();
subplot(2,1,1); plot(Hs1); grid; title('Histogram szumu rawnomiernego');
subplot(2,1,2); plot(Hs2); grid; title('Histogram szumu normlanego');

% Transformacja Fouriera
S1 = fft(s1); S2 = fft(s2);
% Generowanie wykresów
figure();
subplot(2,1,1); plot(f(1:N/2+1), abs(S1(1:N/2+1))/(N/2));
grid; xlabel('f [Hz]'); title('Widmo szumu rawnomiernego');
subplot(2,1,2); plot(f(1:N/2+1), abs(S2(1:N/2+1))/(N/2));
grid; xlabel('f [Hz]'); title('Widmo szumu normlanego');

% Krótkoczasowa transformata Fouriera
[Pss1, fss1] = periodogram(s1, [], [], fp);
[Pss2, fss2] = periodogram(s2, [], [], fp);
% Generowanie wykresów
figure();
subplot(2,1,1); plot(fss1, Pss1);
grid; title('Widmo usrednione szumu rawnomiernego'); xlabel('f [Hz]');
subplot(2,1,2); plot(fss2, Pss2);

```

```

grid; title('Widmo usrednione szumu normlanego'); xlabel('f [Hz]');

%% 3. Dodaj sygnały z punktu 1 i 2. Wyznacz i narysuj funkcję
% autokorelacji, autokowariancji i histogram szuma
% sumarycznego.

disp('Zadanie 3. Dodaj sygnały z punktu 1 i 2. Wyznacz i narysuj funkcję')
disp('autokorelacji, autokowariancji i histogram szuma sumarycznego.')
% Dodawanie sygnałów
x_new1 = x + s1; x_new2 = x + s2;

% Autokorelacja sygnałów
R1 = xcorr(x_new1, 'unbiased'); R2 = xcorr(x_new2, 'unbiased');
% Generowanie wykresów
figure();
subplot(2,1,1); plot(tR, R1);
grid; title('Autokorelacja z szumem rownomiernym');
subplot(2,1,2); plot(tR, R2);
grid; title('Autokorelacja z szumem normlonym');

% Autokowariancja sygnałów
C1 = xcov(x_new1); C2 = xcov(x_new2);
% Generowanie wykresów
figure();
subplot(2,1,1); plot(tR, C1);
grid; title('Autokowariancja z szumem rownomiernym');
subplot(2,1,2); plot(tR, C2);
grid; title('Autokowariancja z szumem normlonym');

% Histogramy rozkładu sygnałów
Hs1 = hist(x_new1, 100); Hs2 = hist(x_new2, 100);
% Generowanie wykresów
figure();
subplot(2,1,1); plot(Hs1);
grid; title('Histogram sumy z szumem rownomiernym');
subplot(2,1,2); plot(Hs2);
grid; title('Histogram sumy z szumem normlonym');

%% 4. Powtórz operacje z punktu 3 po dodaniu do sygnału 1+2 jeszcze
% jednej sinusoidy o częstotliwości 250 Hz.

disp('4. Powtórz operacje z punktu 3 po dodaniu do sygnału 1+2')
disp('jeszcze jednej sinusoidy o częstotliwości 250 Hz.')
% Generowanie sygnałów
x_new1 = x + s1 + A*sin(2*pi*250*t); x_new2 = x + s2 + A*sin(2*pi*250*t);

% Autokorelacja sygnałów
R1 = xcorr(x_new1, 'unbiased'); R2 = xcorr(x_new2, 'unbiased');
% Generowanie wykresów
figure();
subplot(2,1,1); plot(tR, R1);
grid; title('Autokorelacja z szumem rownomiernym');
subplot(2,1,2); plot(tR, R2);
grid; title('Autokorelacja z szumem normlonym');

% Autokowariancja sygnałów
C1 = xcov(x_new1); C2 = xcov(x_new2);
% Generowanie wykresów
figure();

```

```

subplot(2,1,1); plot(tR, C1);
grid; title('Auto kowariancja z szumem rownomiernym');
subplot(2,1,2); plot(tR, C2);
grid; title('Auto kowariancja z szumem normlany');

% Histogramy rozkładu sygnałów
Hs1 = hist(x_new1, 100); Hs2 = hist(x_new2, 100);
% Generowanie wykresów
figure();
subplot(2,1,1); plot(Hs1);
grid; title('Histogram sumy z szumem rownomiernym');
subplot(2,1,2); plot(Hs2);
grid; title('Histogram sumy z szumem normlany');

%% 5. Zmoduluj w amplitudzie sygnał sinusoidalny z punktu pierwszego.

disp('5. Zmoduluj w amplitudzie sygnał sinusoidalny z punktu pierwszego.')
% Generowanie wykresów i sygnałów
figure();
ampl = hamming(N); % Okno hamminga
y1 = ampl.*x; subplot(2,1,1);
plot(t, y1); grid; title('Sygnal z modulacja amplitudy')

ampl = exp(-10*t); % Okno tłumiące
y2 = ampl.*x; subplot(2,1,2);
plot(t, y2); grid; title('Sygnal z modulacja amplitudy')

%% 6. Wygeneruj sygnał sinusoidalny z liniową modulacją częstotliwości
% oraz z sinusoidalną modulacją częstotliwości.

disp('6. Wygeneruj sygnał sinus. z liniową modulacją częstotliwości')
disp(' oraz z sinusoidalną modulacją częstotliwości.')
% Generowanie wykresów i sygnałów
figure();
fx = 0; alfa = 50;
y3 = sin(2*pi*(fx*t + 0.5*alfa*t.^2));
subplot(2,1,1); plot(t, y3);
grid; title('Sygnal z modulacja czestotliwosci')

fx = 10; fm = 5; df = 25;
y4 = sin(2*pi*(fx*t + df * sin(2*pi*fm*t)/(2*pi*fm)));
subplot(2,1,2); plot(t, y4);
grid; title('Sygnal z modulacja czestotliwosci')

%% 7. Sklej dwa sygnały.
disp('7. Sklej dwa sygnały.')
y5 = [ y1 y4];
plot(y5); grid; title('Sygnal sklejony');

%% 8. „Spleć” ze sobą dwa sygnały, czyli dokonaj filtracji jednego z nich
% przez drugi.

disp('8. „Spleć” ze sobą dwa sygnały, czyli dokonaj')
disp(' filtracji jednego z nich przez drugi.')
% Generowanie sygnałów
T = 5; N = 1000; dt = T/N; t = dt*(0:N);
x = sin(2*pi*t.^2) + 0.5*sin(2*pi*t.^8);
h = sin(2*pi*2*t).*exp(-4*t);
y = conv(x, h);

```

```
% Generowanie wykresow
figure()
subplot(3,1,1); plot(t, x); grid; title('Sygnal wejsciowy');
subplot(3,1,2); plot(t, h); grid; title('Sygnal filtru');
subplot(3,1,3); plot(t, y(1:N+1)); grid; title('Sygnal wyjsciowy');

%% 9. Skwantuj sygnał x(t) z punktu 8
disp('9. Skwantuj sygnał x(t) z punktu 8')
% Generowanie wykresow
figure(); plot(t, x); hold on;
x_min = -1.5; x_max = 1.5; x_zakres = x_max-x_min; % Zakres min, max
Nb = 3; Nq = 2^Nb; %Nb - l. bitow, Nq - l. przedzialow kwantowania
dx = x_zakres/Nq; % szerokosc przedzialu
xq = dx*round(x/dx); %kwatnyzacja
plot(t, xq); title('Sygnal po i przed skwantowaniem')
```

Tab. 8.2. Kod programów na podstawie rozdziału 2.

```
% Zielinski 2.5
% Przykład transofrmacji orogonalnych sygnalow
% Mateusz Krupnik

% Transformaty ortogonalne sygnałów
% 1) kształt dyskretnych baz: Fouriera, kosinusowej,
% sinusowej, Hadamarda, Walsha
% 2) dopasowanie bazy, przykładowa dekompozycja dwóch sygnałów
clear all; close all; clc; subplot(111);
N=16; % wybór liczby funkcji bazowych (wymiar przestrzeni wektorowej)
n=0:N-1; % indeksy wszystkich próbek poszczególnych funkcji bazowych
NN=2*N; % zmienna pomocnicza

% Kształt funkcji bazowych dla transformacji kosinusowej i sinusowej
% n-ta próbka k-tej funkcji bazowej
f = 1/sqrt(N); %normalizacja
c = [sqrt(1/N) sqrt(2/N)*ones(1, N-1)];
s = sqrt(2/(N+1));

for k=0:N-1
    % zamiast kolejnej petli for n:N-1 jest wektor razy skalar wpisny jak
    % wiersz a kolejna baza jako kolejny wiersz
    baza_fouriera(k+1, n+1) = f * exp(2i*pi*k*n/N);
    baza_cos(k+1, n+1) = c(k+1) * cos(k*pi*(n+1/2)/N);
    baza_sin(k+1, n+1) = s * sin(pi*(k+1)*(n+1) / (N+1));
end

% Kształt funkcji bazowych dla transformacji Hadamarda
% n-ta próbka k-tej funkcji bazowej
m=log2(N); c = sqrt(1/N);
for k=0:N-1
    kk = k;
    for i=0:m-1
        ki(i+1) = rem(kk, 2); kk=floor(kk/2);
    end
    for n=0:N-1
        nn = n;
        for i=0:m-1
            ni(i+1) = rem(nn, 2); nn = floor(nn/2);
        end
    end
end
```

```

        end
    baza_HD(k+1,n+1) = c * (-1)^sum(ki .* ni);
end

% Kształt funkcji bazowych dla transformacji Haara
% n-ta próbka k-tej funkcji bazowej
c = sqrt(1/N); baza_HR(1,1:N) = c*ones(1, N);
for k = 1:N-1
    p = 0;
    while (k+1 > 2^p)
        p = p + 1;
    end
    p=p-1;
    q=k-2^p+1;
    for n=0:N-1
        x = n/N;
        if ( ((q-1)/2^p <= x) & (x < (q-1/2)/2^p) )
            baza_HR(k+1, n+1) = c*2^(p/2);
        elseif ( ((q-1/2)/2^p <= x) & (x < q/2^p) )
            baza_HR(k+1, n+1) = -c*2^(p/2);
        else baza_HR(k+1, n+1) = 0;
        end
    end
end
n=0:N-1;
figure()
subplot(2,1,1);
plot(n, imag(baza_fouriera(:,:,1))); grid; title('IMAG fourier');
subplot(2,1,2);
plot(n, real(baza_fouriera(:,:,1))); grid; title('REAL fourier');

figure()
subplot(2,1,1);
plot(n, real(baza_sin(:,:,1))); grid; title('baza sinusowa');
subplot(2,1,2);
plot(n, real(baza_cos(:,:,1))); grid; title('baza cosinusowa');
figure()
subplot(2,1,1);
plot(n, real(baza_HR(:,:,1)), '--'); grid; title('baza HR');
subplot(2,1,2);
plot(n, real(baza_HD(:,:,1)), '-.'); grid; title('baza HD');
% Sprawdzenie ortonormalności wybranych funkcji bazowych
for k=1:N % zbudowanie macierzy transformacji
    Tf(k,1:N) = baza_fouriera(k,1:N); % transformacja Fouriera
    Tc(k,1:N) = baza_cos(k,1:N); % transformacja kosinusowa
    Ts(k,1:N) = baza_sin(k,1:N); % transformacja sinusowa
    THD(k,1:N) = baza_HD(k,1:N); % transformacja Hadamarda
    THR(k,1:N) = baza_HR(k,1:N); % transformacja Haara
end
T = Tc; % wybierz transformację
I = T * T'; % sprawdź, czy iloczyn jest macierzą diagonalną jednostkową

% Przykład analizy (dekompozycji) i syntezy sygnału
% Generacja sygnałów testowych
kk = 2; % testowy „indeks” częstotliwości, np. 1.35, 2, 2.5, 3
fi = 0; % przesunięcie fazowe 0, pi/8, pi/4, pi/2
n = 0 : N-1;
x1 = cos( (2*pi/N)*kk*n + fi ); % cz. rzeczywista bazy fourierowskiej
x2 = cos( pi*kk*(2*n+1)/NN + fi ); % wektor bazy kosinusowej

```

```

x3 = sin( pi*(kk+1)*(n+1)/(N+1) + fi ); % wektor bazy sinusowej
x4 = cos( (2*pi/N)*2*n + fi ) + cos( (2*pi/N)*4*n + fi );
x5 = [ ones(1,N/2) zeros(1,N/2) ];
x6 = [ -ones(1,N/4) ones(1,N/2) -ones(1,N/4) ];
x = x4; % wybór konkretnego sygnału do dekompozycji
T = THD; % wybór transformacji: Tf, Tc, Ts, THD, THR

a = T * x'; % analiza w zapisie macierzowym
y = T' * a; % synteza w zapisie macierzowym
y = y'; % zamień wektor pionowy na poziomy
figure()
stem(n,x,'filled','-k');
axis tight; title('sygnał analizowany x(l)');
xlabel('numer próbki');
figure()
stem(n,real(a),'filled','-k');
axis tight; title('wsp dekompozycji alfa(k)');
xlabel('numer próbki');
figure()
stem(n,y,'filled','-k');
axis tight; title('sygnał zsyntezowany x(l)');
xlabel('numer próbki');
figure()
stem(n,y-x,'filled','-k');
axis tight; title('błąd syntezy 1: y(l)-x(l)');
xlabel('numer próbki');

% Analiza i synteza w zapisie niemacierzowym
y=zeros(1,N); %
for k = 0 : N-1 % ANALIZA: oblicz współczynniki
    a(k+1) = sum( x .* conj(T(k+1,1:N)) ); %
end %
for k = 0 : N-1 % SYNTEZA: odtwórz sygnał
    y = y + a(k+1) * T(k+1,1:N); %
end %
figure()
stem(n,y-x,'filled','-k');
axis tight; title('błąd syntezy 2: y(l)-x(l)');
xlabel('numer próbki');

```

Tab. 8.3. Kod programów na podstawie rozdziału 3.

```

% Rozdzia 3 Zielinski - przykady
%                               Mateusz Krupnik
clear all; close all; clc;

% Parametry programu
T = 1; % okres [s]
N = 1000; % liczba probek
dt = T/N; f0 = 1/T; % podstawa czasu [s], czest [Hz]
t = 0:dt:(N-1)*dt; % wektor czasu
A = 1; NF=60; % amplituda i liczba wspolczynnikow sz F
f=0:f0:(NF-1)*f0;

% Generacja sygnalow prostokątnych, trojkątnych i sinudoidalnych

```

```

x= [0 A*ones(1, N/2-1) 0 -A*ones(1, N/2-1)];
x(2, :)=[A*ones(1,N/4) 0 -A*ones(1,N/2-1) 0 A*ones(1,N/4-1)];
x(3, :)=[A*ones(1,N/8) 0 -A*ones(1,5*N/8-1) 0 A*ones(1,2*N/8-1)];
x(4, :)=A/T*t;
x(5, :)=[2*A/T*t(1:N/2+1) 2*A/T*t(N/2:-1:2)];
x(6, :)=sin(2*pi*t/T);

% Wykresy sygnałów
figure("Name", "Sygnaly");
sgtitle("Sygnaly")
for i=1:size(x, 1)
    subplot(size(x, 1)/2, floor(size(x, 1)/2), i);
    plot(t, x(i,:)); title(['Sygnal ' num2str(i)]);
    grid; xlabel('Czas [s]'); ylabel('Ampl.');
end
hold on;

%% Wyznaczanie współczynników rozwinięcia sygnału w szereg Fouriera
figure("Name", "Współczynniki szeregu Fouriera");
sgtitle("Współczynniki szeregu Fouriera");
% preallokacja pamięci
a = zeros(size(x, 1), NF); b=a; c0 = zeros(1, size(x, 1));
for i=1:size(x, 1)
    for k=0:NF-1
        ck=cos(2*pi*k*f0*t); sk=sin(2*pi*k*f0*t);
        % cosinus oraz sinuns o nr k
        a(i, k+1)=sum(x(i,:).*ck)/N; % współczynnik a
        b(i, k+1)=sum(x(i,:).*sk)/N; % współczynnik b
    end
    a(i, 1) = a(i, 1)/2;
    %Generowanie wykresów dla sygnałów
    subplot(size(x, 1), 2, 2*i-1);
    stem(f,a(i,:),'filled'); xlabel('[Hz]');
    title(['Syg. ' num2str(i) ': COS']);
    subplot(size(x, 1), 2, 2*i);
    stem(f,b(i,:),'filled'); xlabel('[Hz]');
    title(['Syg. ' num2str(i) ': SIN']);
end

%% Porównanie z FFT
figure("Name", "Blad współczynników względem FFT")
sgtitle("Blad współczynników względem FFT")
for i=1:size(x,1)
    X = fft(x(i,:), N)/N; % obliczenie współczynników
    X = conj(X); % sprzężenie
    % Generowanie błędów dla współczynników
    subplot(size(x, 1), 2, 2*i-1);
    plot(f, a(i,:) - real(X(1:NF)));
    xlabel('[Hz]'); title(['Syg. ' num2str(i) ': DFT-SIN']);
    subplot(size(x, 1), 2, 2*i);
    plot(f, b(i,:) - imag(X(1:NF)));
    xlabel('[Hz]'); title(['Syg. ' num2str(i) ':DFT-COS']);
end

%% Synteza sygnałów z współczynników
figure("Name", "Zsyntezowane sygnały z rozwinięcia w szereg")
sgtitle("Zsyntezowane sygnały z rozwinięcia w szereg Fouriera")
for i=1:size(x, 1)
    y=zeros(size(t)); y_temp=y; % preallokacja
    for k=0:NF-1

```

```
% generowanie sygnałów dla kolejnych współczynników
y_temp = 2*a(i, k+1)*cos(2*pi*k*f0*t) + ...
         2*b(i, k+1)*sin(2*pi*k*f0*t);
y = y + y_temp;
subplot(size(x, 1), 2, 2*i-1); plot(t, y_temp); hold on;
end
% Wykres syntezy
subplot(size(x, 1), 2, 2*i-1);
plot(t, y, '--'); title(['Syg. ' num2str(i) ' składowe']);
subplot(size(x, 1), 2, 2*i);
plot(t, x(i, :), t, y, '--');
xlabel('Czas [s]'); title(['Syg. ' num2str(i)]);
end
```

Tab. 8.4. Kod programów na podstawie rozdziału 4.

```
% Rozdział 4 Zielinski
% Mateusz Krupnik
clear all; close all;
% Generowanie przebiegu
fx = 1; % częstotliwość sygnału [Hz]
fps = 100; % stara częstotliwość próbkowania [Hz]
N = 200; % liczba próbek sygnału spróbk. z częstotliwością fps (stara)
K = 10; % ile razy zmniejszyć częstotliwość próbkowania

% Generacja sygnału
dts = 1/fps; % stary okres próbkowania
ts = 0:dts:(N-1)*dts;
xs = sin(2*pi*fx*ts); % sprobkowany
% Generacja wykresu
subplot(3,1,1)
plot(ts, xs, 'r', ts, xs, 'o');
grid; title('Sygnał sprobkowany stara czest. probk.');

fpn = fps/K;
xn = xs(1:K:length(xs)); M = length(xn); % sprobk. nowa częstotliwość
dtn = K*dts; tn = 0:dtn:(M-1)*dtn;
% Generacja wykresu
subplot(3,1,2)
plot(tn, xn, 'b', tn, xn, 'o');
grid; title('Sygnał sprobkowany nowa czest. probk.');?>
hold on;
subplot(3,1,3)
stem(xn, 'b');

%% Rekonstrukcja sygnału
% Funkcja aproks.
t = -(N-1)*dts : dts : (N-1)*dts; % czas trwania f. aproks
f = 1/(2*dtn); % częstotliwość zer w f. aproks
fun_aproks = sin(2*pi*f*t)./(2*pi*f*t); %funkcja sinc=sinx/x
fun_aproks(N) = 1; % w wartości 0 mamy 0/0
tz = [-fliplr(tn) tn(2:M)];
z = [zeros(1, M-1) 1 zeros(1, M-1)] % delta diraca

% Wykres
figure()
plot(t, fun_aproks, 'b', tz, z, 'o'); grid; title('Sinc - f. aproks.');
```

```

%% Aproksymacja
y = zeros(1, N); % preallokacja
ty = 0:dts:(N-1)*dts; % wektor czasu
figure()
for k=1:M
    % funkcja aproksymujaca
    fun_aproks1 = fun_aproks( N-(k-1)*K : (2*N-1)-(k-1)*K );
    y1 = xn(1, k)*fun_aproks1; % probka*f.aprosk
    y = y + y1; % sumowanie
    % wykres
    subplot(311); plot(ty, fun_aproks1);
    grid; hold on; title('Kolejna funkcja aproksymujaca');
    subplot(312); plot(ty,y1);
    grid; title('Kolejny składnik sumy'); hold on;
end
% Wykres calosci
subplot(313); plot(ty,y); grid; title('Suma składowych');

%% Wykres porównawczy
figure()
subplot(211); plot(ty, y, 'b');
grid; title('Sygнал odtworzony');
subplot(212); plot(ty, xs(1:N)-y(1:N), 'b');
grid; title('Różnica między syg. org. a odtw.');

```

Tab. 8.5. Kod programów na podstawie rozdziału 5.

```

% Rodzial 5 Zielinski
% Mateusz Krupnik

% Projektowanie filtrów metoda zer i biegunow
clc; clear all; close all;
% Przykład 1: projekt filtra pasmowoprzepustowego
% o wpass1 = 9.5 rd, wpass2 = 10.5 rd

z1 = 5; z2 = 15; % ZERA na osi urojonej
z = j*[ -z2, -z1, z1, z2 ];
odl = 0.5; p1 = 9.5; p2 = 10.5; % BIEGUNY w pobliży osi urojonej
p = [ -odl-j*p2, -odl-j*p1, -odl+j*p1, -odl+j*p2 ];
WMAX=20; TMAX=20; % max pulsacja, max czas obserwacji
show_results(z, p, WMAX, TMAX, "przyklad 1.");

%% Przykład 2: znajdowanie zer i biegunów zadanej transmitancji
b=[ 0.66667 0 1 ]; % współczynniki licznika transmitancji
a=[ 4.0001 5.0081 3.1650 1 ]; % współczynniki mianownika transmitancji
[z,p,wzm] = tf2zp(b,a); % wspóln. wielomianów -> zera wielomianów
z = z'; p = p'; % wektor pionowy -> wektor poziomy
WMAX=5; TMAX=25; % max pulsacja, max czas obserwacji
show_results(z, p, WMAX, TMAX, "przyklad 2.");

%% Przykład 3: projekt filtra górnoprzepustowego
z1 = 0; % ZERA na osi urojonej
z2 = 0+j*1; z3 = 0-j*1;
z4 = 0+j*2; z5 = 0-j*2;
z6 = 0+j*3; z7 = 0-j*3;
z = [ z1 z2 z3 z4 z5 z6 z7 ];
p1 = -1; % BIEGUNY w pobliży osi urojonej

```

```

p2 = -1+j*1; p3 = -1-j*1;
p4 = -1+j*2; p5 = -1-j*2;
p6 = -1+j*3; p7 = -1-j*3;
p = [ p1 p2 p3 p4 p5 p6 p7 ];
WMAX=20; TMAX=5; % max pulsacja, max czas obserwacji
show_results(z, p, WMAX, TMAX, "przyklad 3.");

%%%%% FUNKCJA PREZENTACJI %%%%%%
function show_results(z, p, WMAX, TMAX, title_text)
figure();
plot(real(z), imag(z), 'or', real(p), imag(p), 'xb'); grid;
title(['Zera (o) i bieguny (x):' title_text]);
xlabel('Real'); ylabel('Imag [rd/s]');
w = 0 : 0.01 : WMAX; % wybrane pulsacje widma
[b,a] = zp2tf(z',p',1); % zera, bieguny -> współczynniki wielomianów
H = freqs(b,a,w); % wyznaczenie widma transmitancji dla zadanego w
Hm = abs(H); HmdB = 20*log10(Hm); % moduł transmitancji
Hf = angle(H); Hfu = unwrap(Hf); % faza transmitancji
figure(); sgtitle(['Charakterystyki fazowe dla: ' title_text]);
subplot(221);
plot(w,Hm,'k'); grid;
title('Ch-ka amplitudowa'); xlabel('Częstość w[rd/s]');
subplot(222);
plot(w,HmdB,'k'); grid;
title('Ch-ka amplitudowa w dB'); xlabel('Częstość w[rd/s]');
subplot(223);
plot(w,Hf,'k'); grid;
title('Ch-ka fazowa'); xlabel('Częstość w[rd/s]'); ylabel('[rd]');
subplot(224);
plot(w,Hfu,'k'); grid;
title('Ch-ka fazowa unwrap');
xlabel('Częstość w[rd/s]'); ylabel('[rd]');

% Odpowiedź impulsowa
h = impulse(b,a,TMAX); % funkcja z przybornika CONTROL
dt = TMAX/(length(h)-1); th = 0 : dt : TMAX;
figure(); sgtitle(['Charakterystyki czasowe dla: ' title_text]);
subplot(211);
plot(th,h,'k');
grid; title('Odpowiedź impulsowa'); xlabel('Czas t[s]');
% Odpowiedź na skok jednostkowy
u = step(b,a,TMAX); % funkcja z przybornika CONTROL
dt = TMAX/(length(u)-1); tu = 0 : dt : TMAX;
subplot(212);
plot(tu,u,'k'); grid; title('Odpowiedź skokowa'); xlabel('Czas t[s]');
end

```

Tab. 8.6. Kod programów na podstawie rozdziału 6.

```

% Rozdział 6 Zielinski
% Mateusz Krupnik

clc; close all; clear all;
% Analogowe filtry Butterwortha i Czebyszewa
apass = 1; % nielinowość pasma przepustowego w dB

```

```

astop = 50; % tłumienie w paśmie zapowym

%% Filtry Butterwortha HP, BP, BS - filtry oparte na okregu w lewej
polpl.
typ = "Butterworth";
for i=1:4
    % filtr dolnoprzepustowy LP
    if i==1
        filtr = 'LowPass';
        fpass = 1000; fstop = 4000; % Hz
        ws = fstop/fpass; % częstotliwość znormalizowana. s=s'/w0, w0=2*pi*fpass

    % filtr gornoprzepustowy HP
    elseif i==2
        filtr = 'HighPass';
        fstop = 2000; % częstotliwość przepustowej odpowiadającej astopy
        fpass = 3000; % częstotliwość zaporowej odpowiadającej apass
        ws = fpass/fstop; % transformacja częstotliwości: s=w0/s', w0=2*pi*fpass

    % filtr srodkowoprzepustowy BP
    elseif i==3
        filtr = 'BandPass';
        fs1 = 1500; % dolna częstotliwość stopa
        fp1 = 2000; % dolna częstotliwość pasa
        fp2 = 3000; % górna częstotliwość pasa
        fs2 = 3500; % górna częstotliwość stopa
        % transformacja częstotliwości
        ws1t = (fs1^2 - fp1*fp2) / (fs1*(fp2-fp1));
        ws2t = (fs2^2 - fp1*fp2) / (fs2*(fp2-fp1));
        ws = min(abs(ws1t), abs(ws2t));

    % filtr srodkowozaporowy BS
    else
        filtr = 'BandStop';
        fp1 = 1500; % dolna częstotliwość filtra pasmowego
        fs1 = 2000; % dolna częstotliwość filtra pasmowego
        fs2 = 3000; % górna częstotliwość filtra pasmowego
        fp2 = 3500; % górna częstotliwość filtra pasmowego
        % transformacja częstotliwości
        ws1t = (fs1*(fp2-fp1)) / (fs1^2 - fp1*fp2);
        ws2t = (fs2*(fp2-fp1)) / (fs2^2 - fp1*fp2);
        ws = min(abs(ws1t), abs(ws2t));
    end

    % Przeliczenie na wartość bezwzględna
    wzm_p = 10^(-apass/20);
    wzm_s = 10^(-astop/20);

    if ( (i==1) || (i==2) )
        vp = 2*pi*fpass;
        vs = 2*pi*fstop;
        f_ps = [fpass, fstop];
        wzm_ps = [wzm_p, wzm_s];
        wzmdB_ps = [-apass, -astop];
    else
        vp = 2*pi*[ fp1 fp2 ];
        vs = 2*pi*[ fs1 fs2 ];
        vc = 2*pi*sqrt(fp1*fp2); % pulsacja środkowa
        % szerokość filtra wokół vc
        f_ps = [fp1,fp2,fs1,fs2];
    end

```

```

dv = 2*pi*(fp2-fp1);
wzm_ps = [wzm_p, wzm_p, wzm_s, wzm_s];
wzmdB_ps = [-apass, -apass, -astop, -astop];
end

disp(['num2str(i) ' - FILTR: ' filtr ' - ' typ]);

% Obliczenie parametrów N i w0 - funkcja buttord
wp = 1;
N = ceil(log10( (10^(astop/10)-1) /(10^(apass/10)-1) ) /
(2*log10(ws/wp)) );
w0 = ws / (10^(astop/10)-1)^(1/(2*N));

% Obliczenie biegunów trans. prototypu - funkcja buttap i zp2tf
dfi0 = (2*pi)/(2*N); % kat „kawałka tortu”
fi = pi/2 + dfi0/2 + (0 : N-1)*dfi0; % katy biegunow
p = w0*exp(j*fi); % bieguny
z = []; % zera
wzm = real( prod(-p) ); % wzmacnienie
a = poly(p); % bieguny --> wsp wielomianu mianownika A(z)
b = wzm; % wielomian licznika B(z)
%
z, p, b, a
figure(4*i-3)
plot( real(p), imag(p), 'x' ); grid;
title(['Położenie biegunów dla filtra:' filtr ' - ' typ]);
xlabel('real'); ylabel('imag');

% Porównanie z Matlabem
[NN,ww0] = buttord( vp, vs, apass, astop, 's' );
blad_N = N-NN; disp(['Blad rzedu:' num2str(blad_N)]);

% Oblicz charakterystykę częstotliwościową H(w)=B(w)/A(w)
% zakres pulsacji unorm.; pulsacja granicy pasma przepustowego = 1
w = 0 : 0.005 : 2;
H = freqs(b,a,w); % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);

figure(4*i-2); subplot(211);
plot(w,abs(H));
grid; title(['Moduł prototypu LowPass dla' filtr ' - ' typ]);
xlabel('Pulsacja [rad/sek]');

subplot(212); plot(w,20*log10(abs(H))); grid;
title(['Moduł prototypu LowPass w dB dla' filtr ' - ' typ]);
xlabel('Pulsacja [rad/sek]'); ylabel('dB');
% Transformata częstotliwości filtra analogowego: prototyp unormowany
% --> wynikowy filtr

% LowPass to LowPass: s=s/w0
if (i==1) [z,p,wzm] = lp2lpTZ(z,p,wzm,wp); end
% LowPass to HighPass: s=w0/s
if (i==2) [z,p,wzm] = lp2hpTZ(z,p,wzm,wp); end
% LowPass to BandPass: s=(s^2+wc^2)/(dw*s)
if (i==3) [z,p,wzm] = lp2bpTZ(z,p,wzm,vc,dv); end
% LowPass to BandStop: s=(dw*s)/(s^2+wc^2)
if (i==4) [z,p,wzm] = lp2bstZ(z,p,wzm,vc,dv); end

b=wzm*poly(z); a=poly(p);
% Pokaż zera i bieguny po transformacji częstotliwości
figure(4*i-1)

```

```

plot( real(z), imag(z), 'o', real(p), imag(p), 'x' ); grid;
title(['Położenie biegunów dla filtra ' filtr ' - ' typ]);
xlabel('real'); ylabel('imag');

%
% p, z
% a, b
printsys(b,a,'s');
% Końcowa charakterystyka częstotliwościowa
NF = 1000; % ile punktów
fmin = 0; % dolna częstotliwość
fmax = 5000; % górna częstotliwość
f = fmin : (fmax-fmin)/(NF-1) : fmax; % wszystkie częstotliwości
w = 2*pi*f; % wszystkie pulsacje
H = freqs(b,a,w); % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);

figure(4*i); subplot(211);
plot( f,abs(H), f_ps, wzm_ps, 'ro' );
grid; title(['Moduł dla filtra: ' filtr ' - ' typ]);
xlabel('Częstotliwość [Hz]');

subplot(212);
plot(f,20*log10(abs(H)), f_ps, wzmdB_ps, 'ro' );
axis([fmin,fmax,-100,20]);
grid; title(['Moduł w dB filtra ' filtr ' - ' typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f,unwrap(angle(H))); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel('rad');

end

%% Filtry na podstawie prototypu Czebyszewa I typu
typ = "Czebyszewa I";
for i=1:4
    % filtr dolnoprzepustowy LP
    if i==1
        filtr = 'LowPass';
        fpass = 1000; fstop = 4000; % Hz
        ws = fstop/fpass; % częstotliwość znormalizowana. s=s'/w0, w0=2*pi*fpass

    % filtr gornoprzepustowy HP
    elseif i==2
        filtr = 'HighPass';
        fstop = 2000; % częst. pasma przepustowego odpowiadająca astop
        fpass = 3000; % częst. pasma zaporowego odpowiadająca apass
        ws = fpass/fstop; % transformacja częst. i: s=w0/s', w0=2*pi*fpass

    % filtr srodkowoprzepustowy BP
    elseif i==3
        filtr = 'BandPass';
        fs1 = 1500; % dolna częstotliwość stop
        fp1 = 2000; % dolna częstotliwość pass
        fp2 = 3000; % górna częstotliwość pass
        fs2 = 3500; % górna częstotliwość stop
        % transformacja częstotliwości
        ws1t = (fs1^2 - fp1*fp2) / (fs1*(fp2-fp1));
        ws2t = (fs2^2 - fp1*fp2) / (fs2*(fp2-fp1));
        ws = min(abs(ws1t), abs(ws2t));

    % filtr srodkowozaporowy BS
    else
        filtr = 'BandStop';
        fp1 = 1500; % dolna częstotliwość filtra pasmowego
        fs1 = 2000; % dolna częstotliwość filtra pasmowego
    end
end

```

```

fs2 = 3000; % górnna częstotliwość filtra pasmowego
fp2 = 3500; % górnna częstotliwość filtra pasmowego
% transformacja częstotliwości
ws1t = (fs1*(fp2-fp1)) / (fs1^2 - fp1*fp2);
ws2t = (fs2*(fp2-fp1)) / (fs2^2 - fp1*fp2);
ws = min(abs(ws1t), abs(ws2t));
end

% Przeliczenie na wartość bezwzględna
wzm_p = 10^(-apass/20);
wzm_s = 10^(-astop/20);

if ( (i==1) || (i==2))
    vp = 2*pi*fpass;
    vs = 2*pi*fstop;
    f_ps = [fpass, fstop];
    wzm_ps = [wzm_p, wzm_s];
    wzmdB_ps = [-apass, -astop];
else
    vp = 2*pi*[ fp1 fp2 ];
    vs = 2*pi*[ fs1 fs2 ];
    vc = 2*pi*sqrt(fp1*fp2);      % pulsacja środka
    % szerokość filtra wokół vc
    f_ps = [fp1,fp2,fs1,fs2];
    dv = 2*pi*(fp2-fp1);
    wzm_ps = [wzm_p, wzm_p, wzm_s, wzm_s];
    wzmdB_ps = [-apass, -apass, -astop, -astop];
end

disp([num2str(i) ' - FILTR: ' filtr ' - ' typ]);

% Obliczenie parametrów N i w0
wp = 1;
Nreal = acosh(sqrt((10^(astop/10)-1) / ...
    (10^(apass/10)-1))) / acosh(ws/wp);
N = ceil(Nreal);
epsi = sqrt(10^(apass/10)-1);
D = asinh(1/epsi)/N; R1 = sinh(D); R2 = cosh(D);

% Obliczenie biegunów trans. prototypu - funkcja buttap i zp2tf
dfi0 = (2*pi)/(2*N);                      % kąt „kawałka tortu”
fi = pi/2 + dfi0/2 + (0 : N-1)*dfi0;       % kąty biegunów
p1 = R1 * exp(1i*fi);                      % bieguny R1
p2 = R2 * exp(1i*fi);                      % bieguny R2
p = real(p1) +1i*imag(p2);                 % Polaczzone bieguny
z = [];                                     % zera
wzm = prod(-p);                            % wzmacnianie
a = poly(p);                                % bieguny --> wsp wielomianu mianownika A(z)
b = wzm;                                     % wielomian licznika B(z)
if (rem(N,2)==0) b = b*10^(-apass/20); end

figure(4*i-3+16)
plot( real(p), imag(p), 'x' ); grid;
title(['Położenie biegunów dla filtra:' filtr ' - ' typ]);
xlabel('real'); ylabel('imag');

% Porównanie z Matlabem
[NN,ww0] = cheblord( vp, vs, apass, astop, 's' );
blad_N = N-NN; disp(['Blad rzedu:' num2str(blad_N)]);

```

```

% Oblicz charakterystykę częstotliwościową H(w)=B(w) / A(w)
% zakres pulsacji unormowanej; pulsacja granicy pasma przepustowego =
1
w = 0 : 0.005 : 2;
H = freqs(b,a,w);    % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);

figure(4*i-2+16); subplot(211);
plot(w,abs(H)); grid;
title(['Moduł prototypu LowPass dla' filtr ' - ' typ]);
xlabel('Pulsacja [rad/sek]');

subplot(212); plot(w,20*log10(abs(H))); grid;
title(['Moduł prototypu LowPass w dB dla' filtr ' - ' typ]);
xlabel('Pulsacja [rad/sek]'); ylabel('dB');
% Transformata częstotliwości filtra analogowego: prototyp unormowany
% --> wynikowy filtr

% LowPass to LowPass: s=s/w0
if (i==1) [z,p,wzm] = lp2lpTZ(z,p,wzm,vp); end
% LowPass to HighPass: s=w0/s
if (i==2) [z,p,wzm] = lp2hpTZ(z,p,wzm,vp); end
% LowPass to BandPass: s=(s^2+wc^2)/(dw*s)
if (i==3) [z,p,wzm] = lp2bpTZ(z,p,wzm,vc,dv); end
% LowPass to BandStop: s=(dw*s)/(s^2+wc^2)
if (i==4) [z,p,wzm] = lp2bstZ(z,p,wzm,vc,dv); end

b=wzm*poly(z); a=poly(p);
% Pokaż zera i biegundy po transformacji częstotliwości
figure(4*i-1+16)
plot( real(z), imag(z), 'o', real(p),imag(p), 'x' ); grid;
title(['Położenie biegunów dla filtra' filtr ' - ' typ]);
xlabel('real'); ylabel('imag');

%
% p, z
% a, b
printsys(b,a,'s');
% Końcowa charakterystyka częstotliwościowa
NF = 1000; % ile punktów
fmin = 0; % dolna częstotliwość
fmax = 5000; % górna częstotliwość
f = fmin : (fmax-fmin)/(NF-1) : fmax; % wszystkie częstotliwości
w = 2*pi*f; % wszystkie pulsacje
H = freqs(b,a,w); % alternatywa: H = polyval( b,j*w )./polyval(a,j*w);

figure(4*i+16); subplot(211);
plot(f, abs(H), f_ps, wzm_ps, 'ro');
grid; title(['Moduł dla filtra:' filtr ' - ' typ]);
xlabel('Częstotliwość [Hz]');

subplot(212);
plot(f,20*log10(abs(H)), f_ps, wzmdB_ps, 'ro');
axis([fmin,fmax,-100,20]);
grid; title(['Moduł w dB filtra' filtr ' - ' typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f,unwrap(angle(H))); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel('[rad]');
end

%% Filtry na podstawie prototypu Czebyszewa II typu
typ = "Czebyszewa II";

```

```

for i=1:4
    % filtr dolnoprzepustowy LP
    if i==1
        filtr = 'LowPass';
        fpass = 1000; fstop = 4000; % Hz
        ws = fstop/fpass; % częstotliwość znormalizowana. s=s'/w0, w0=2*pi*fpass

    % filtr gornoprzepustowy HP
    elseif i==2
        filtr = 'HighPass';
        fstop = 2000; % częstotliwość przepustowej odpowiadającej astop
        fpass = 3000; % częstotliwość zaporowej odpowiadającej apass
        ws = fpass/fstop; % transformacja częstotliwości: s=w0/s', w0=2*pi*fpass

    % filtr srodkowoprzepustowy BP
    elseif i==3
        filtr = 'BandPass';
        fs1 = 1500; % dolna częstotliwość stop
        fp1 = 2000; % dolna częstotliwość pass
        fp2 = 3000; % górna częstotliwość pass
        fs2 = 3500; % górna częstotliwość stop
        % transformacja częstotliwości
        ws1t = (fs1^2 - fp1*fp2) / (fs1*(fp2-fp1));
        ws2t = (fs2^2 - fp1*fp2) / (fs2*(fp2-fp1));
        ws = min(abs(ws1t), abs(ws2t));

    % filtr srodkowozaporowy BS
    else
        filtr = 'BandStop';
        fp1 = 1500; % dolna częstotliwość filtra pasmowego
        fs1 = 2000; % dolna częstotliwość filtra pasmowego
        fs2 = 3000; % górna częstotliwość filtra pasmowego
        fp2 = 3500; % górna częstotliwość filtra pasmowego
        % transformacja częstotliwości
        ws1t = (fs1*(fp2-fp1)) / (fs1^2 - fp1*fp2);
        ws2t = (fs2*(fp2-fp1)) / (fs2^2 - fp1*fp2);
        ws = min(abs(ws1t), abs(ws2t));
    end

    % Przeliczenie na wartość bezwzględna
    wzm_p = 10^(-apass/20);
    wzm_s = 10^(-astop/20);

    if ( (i==1) || (i==2) )
        vp = 2*pi*fpass;
        vs = 2*pi*fstop;
        f_ps = [fpass, fstop];
        wzm_ps = [wzm_p, wzm_s];
        wzmdB_ps = [-apass, -astop];
    else
        vp = 2*pi*[ fp1 fp2 ];
        vs = 2*pi*[ fs1 fs2 ];
        vc = 2*pi*sqrt(fp1*fp2); % pulsacja środkowa
        % szerokość filtra wokół vc
        f_ps = [fp1,fp2,fs1,fs2];
        dv = 2*pi*(fp2-fp1);
        wzm_ps = [wzm_p, wzm_p, wzm_s, wzm_s];
        wzmdB_ps = [-apass, -apass, -astop, -astop];
    end

```

```

disp([num2str(i) ' - FILTR: ' filtr ' - ' typ]);
```

% Obliczenie parametrów N i w0

```

wp = 1;
Nreal = acosh(sqrt((10^(astop/10)-1) / ...
(10^(apass/10)-1))) / acosh(ws/wp);
N = ceil(Nreal);
epsi = sqrt(1 / (10^(apass/10)-1));
D = asinh(1/epsi)/N; R1 = sinh(D); R2 = cosh(D);
```

% Obliczenie biegunów trans. prototypu - funkcja buttap i zp2tf

```

dfi0 = (2*pi)/(2*N); % kąt „kawałka tortu”
fi = pi/2 + dfi0/2 + (0 : N-1)*dfi0; % kąty biegunów
p1 = R1 * exp(j*fi); % bieguny R1
p2 = R2 * exp(j*fi); % bieguny R2
p = real(p1) +1i*imag(p2); % Polaczzone bieguny
z = 1i*sin(fi) % zera
wzm = prod(-z) / prod(-p); % wzmacnienie
z = 1./z; p = 1./p;
a = poly(p); % bieguny --> wsp wielomianu mianownika A(z)
b = wzm*poly(z); % wielomian licznika B(z)
```

figure(4*i-3+32)
plot(real(p), imag(p), 'x'); grid;
title(['Położenie biegunów dla filtra:' filtr ' - ' typ]);
xlabel('real'); ylabel('imag');

% Porównanie z Matlabem

```

[NN,ww0] = cheb2ord( vp, vs, apass, astop, 's' );
blad_N = N-NN; disp(['Blad rzedu: ' num2str(blad_N)]);
```

% Oblicz charakterystykę częstotliwościową H(w)=B(w)/A(w)

% zakres pulsacji unormowanej; pulsacja granicy pasma przepustowego =

```

1
w = 0 : 0.005 : 2;
H = freqs(b,a,w); % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);
```

figure(4*i-2+32); subplot(211);
plot(w,abs(H)); grid;
title(['Moduł prototypu LowPass dla' filtr ' - ' typ]);
xlabel('Pulsacja [rad/sek]');

```

subplot(212); plot(w,20*log10(abs(H))); grid;
title(['Moduł prototypu LowPass w dB dla' filtr ' - ' typ]);
xlabel('Pulsacja [rad/sek]'); ylabel('dB');
% Transformata częstotliwości filtra analogowego: prototyp unormowany
% --> wynikowy filtr
% LowPass to LowPass: s=s/w0
if (i==1) [z,p,wzm] = lp2lpTZ(z,p,wzm,wp); end
% LowPass to HighPass: s=w0/s
if (i==2) [z,p,wzm] = lp2hpTZ(z,p,wzm,wp); end
% LowPass to BandPass: s=(s^2+wc^2) / (dw*s)
if (i==3) [z,p,wzm] = lp2bpTZ(z,p,wzm,vc,dv); end
% LowPass to BandStop: s=(dw*s) / (s^2+wc^2)
if (i==4) [z,p,wzm] = lp2bsTZ(z,p,wzm,vc,dv); end
```

```

b=wzm*poly(z); a=poly(p);
% Pokaż zera i bieguny po transformacji częstotliwości
figure(4*i-1+32)
```

```

plot( real(z), imag(z), 'o', real(p), imag(p), 'x' ); grid;
title(['Położenie biegunów dla filtra ' filtr ' - ' typ]);
xlabel('real'); ylabel('imag');

%
% p, z
% a, b
printsys(b,a,'s');
% Końcowa charakterystyka częstotliwościowa
NF = 1000; % ile punktów
fmin = 0; % dolna częstotliwość
fmax = 5000; % górna częstotliwość
f = fmin : (fmax-fmin)/(NF-1) : fmax; % wszystkie częstotliwości
w = 2*pi*f; % wszystkie pulsacje
H = freqs(b,a,w); % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);

figure(4*i+32);
subplot(211);
plot(f, abs(H), f_ps, wzm_ps, 'ro');
grid; title(['Moduł dla filtra: ' filtr ' - ' typ]);
xlabel('Częstotliwość [Hz]');
subplot(212);
plot(f,20*log10(abs(H)), f_ps, wzmdB_ps, 'ro');
axis([fmin,fmax,-100,20]);
grid; title(['Moduł w dB filtra ' filtr ' - ' typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f,unwrap(angle(H))); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel('[rad]');
end

%% Zaprojektowanie układu elektronicznego dolnoprzepustowego filtra
% Butterwortha
% Dane projektowe
fpass = 8000; % często. pasma przepustowego odpowiadająca apass
fstop = 22050; % często. pasma zaporowego odpowiadająca astop
apass = 2; % nieliniowość pasma przepustowego w dB („zwis”)
astop = 40; % tłumienie w paśmie zaporowym

wzm_p = 10^(-apass/20); % tłumienie pass -> wzmacnianie pass
wzm_s = 10^(-astop/20); % tłumienie stop -> wzmacnianie stop
ws = fstop/fpass; % transformacja częstotliwości: s=s'/w0, w0=2*pi*fpass
vp = 2*pi*fpass; vs = 2*pi*fstop;
f_ps = [fpass, fstop];
wzm_ps = [wzm_p, wzm_s];
wzmdB_ps = [-apass, -astop];

wp = 1;
Nreal = log10( (10^(astop/10)-1)/(10^(apass/10)-1) ) / (2*log10(ws/wp));
N = ceil(Nreal);
w0 = ws / (10^(astop/10)-1)^(1/(2*N));

dfi0 = (2*pi)/(2*N); % kat „kawałka tortu”
fi = pi/2 + dfi0/2 + (0 : N-1)*dfi0; % kąty biegunów
p = w0*exp(ji*fi); % bieguny
z = []; % zera
wzm = real(prod(-p)); % wzmacnianie

figure(49)
plot( real(p), imag(p), 'x' ); grid;
title(['Położenie biegunów dla filtra dolnoprzepustowego']);
xlabel('real'); ylabel('imag');

```

```

b = wzm; % wielomian licznika B(z)
a = poly(p); % bieguny --> wsp wielomianu mianownika A(z)
printsys(b,a,'s');

% Porównanie z Matlabem
[NN,ww0] = buttord( vp, vs, apass, astop, 's' );
blad_N = N-NN; disp(['Błąd rzedu: ' num2str(blad_N)]);

% Oblicz charakterystykę częstotliwościową H(w)=B(w)/A(w)
% zakres pulsacji unormowanej; pulsacja granicy pasma przepustowego = 1
w = 0 : 0.005 : 2;
H = freqs(b,a,w); % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);

figure(50); subplot(211);
plot(w,abs(H)); grid; title(['Moduł prototypu LowPass']);
xlabel('Pulsacja [rad/sek]');

subplot(212); plot(w,20*log10(abs(H))); grid;
title(['Moduł prototypu LowPass w dB']);
xlabel('Pulsacja [rad/sek]'); ylabel('dB');

% Transformata częstotliwości filtra analogowego: prototyp unormowany -->
% wynikowy filtr
[z,p,wzm] = lp2lpTZ(z,p,wzm,vp); % LowPass to LowPass: s=s/w0

b=wzm*poly(z); a=poly(p);
% Pokaż zera i bieguny po transformacji częstotliwości
figure(51)
plot( real(z), imag(z), 'o', real(p),imag(p),'x' ); grid;
title(['Położenie biegunów dla filtra LowPass']);
xlabel('real'); ylabel('imag');
printsys(b,a,'s');

% Końcowa charakterystyka częstotliwościowa
NF = 1000; % ile punktów
fmin = 0; % dolna częstotliwość
fmax = 50000; % górna częstotliwość
f = fmin : (fmax-fmin)/(NF-1) : fmax; % wszystkie częstotliwości
w = 2*pi*f; % wszystkie pulsacje
H = freqs(b,a,w); % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);

figure(52); subplot(211);
plot( f,abs(H), f_ps, wzm_ps, 'ro' );
grid; title(['Moduł dla filtra lowpass']);
xlabel('Częstotliwość [Hz]');

subplot(212);
plot(f,20*log10(abs(H)), f_ps, wzmdB_ps, 'ro' );
axis([fmin,fmax,-100,20]);
grid; title(['Moduł w dB filtra lowpass']);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f,unwrap(angle(H))); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel('[rad]');

% Oblicz elementy układu ze wzmacniaczami operacyjnymi
p1 = [ p(1) conj(p(1)) ];
p2 = [ p(2) conj(p(2)) ];
p3 = p(4);
aw1 = poly(p1), aw2 = poly(p2), aw3 = poly(p3);
C = 10^(-9); RA=10^4; Rwy = 10^4;

```

```

disp('== Układ 1 ==')
a = aw1;
a2=a(1); a1=a(2); a0=a(3);
R = 1/(C*sqrt(a0))
RB = (2-a1/sqrt(a0)) * RA
K1 = 1+RB/RA

disp('== Układ 2 ==')
a = aw2;
a2=a(1); a1=a(2); a0=a(3);
R = 1/(C*sqrt(a0))
RB = (2-a1/sqrt(a0)) * RA
K2 = 1+RB/RA

disp('== Układ 3 ==')
a = aw3;
a1=a(1); a0=a(2);
R=1/(C*a0)
K3=1

disp('== Obciążenie ==')
K=K1*K2*K3
G=1
Rx = (K/G)*Rwy
Ry = (G/K)/(1-G/K)*Rx

```

%%%%%%%%%%%%% DEFINICJE FUNKCJI ZAGNIEZDZONYCH %%%%%%

```

% Funkcja transformująca filtr LP znormalizowany na wymagany LP
function [zz,pp,wzm] = lp2lpTZ(z,p,wzm,w0)
    % LowPass to LowPass TZ
    zz = []; pp = [];
    for k=1:length(z)
        zz = [ zz z(k)*w0 ];
        wzm = wzm/w0;
    end
    for k=1:length(p)
        pp = [ pp p(k)*w0 ];
        wzm = wzm*w0;
    end
end

% Funkcja transformująca filtr LP znormalizowany na wymagany HP
function [zz,pp,wzm] = lp2hpTZ(z,p,wzm,w0)
    % LowPass to HighPass TZ
    zz = []; pp = [];
    for k=1:length(z)
        zz = [ zz w0/z(k) ];
        wzm = wzm*(-z(k));
    end
    for k=1:length(p)
        pp = [ pp w0/p(k) ];
        wzm = wzm/(-p(k));
    end
    for k=1:(length(p)-length(z))
        zz = [ zz 0 ];
    end

```

```

    end
end

% Funkcja transformująca filtr LP znromalizowany na wymagany BP
function [zz,pp,wzm] = lp2bpTZ(z,p,wzm,w0,dw)
    % LowPass to BandPass TZ
    pp = []; zz = [];
    for k=1:length(z)
        zz = [ zz roots([ 1 -z(k)*dw w0^2])' ];
    wzm = wzm/dw;
    end
    for k=1:length(p)
        pp = [ pp roots([ 1 -p(k)*dw w0^2])' ];
        wzm = wzm*dw;
    end
    for k=1:(length(p)-length(z))
        zz = [ zz 0 ];
    end
end

% Funkcja transformująca filtr LP znromalizowany na wymagany BS
function [zz,pp,wzm] = lp2bsTZ(z,p,wzm,w0,dw)
    % LowPass to BandStop TZ
    zz = []; pp = [];
    for k=1:length(z)
        zz = [ zz roots([ 1 -dw/z(k) w0^2])' ];
        wzm = wzm*(-z(k));
    end
    for k=1:length(p)
        pp = [ pp roots([ 1 -dw/p(k) w0^2])' ];
        wzm = wzm/(-p(k));
    end
    for k=1:(length(p)-length(z))
        zz = [ zz roots([ 1 0 w0^2])' ];
    end
end

```

Tab. 8.7. Kod programów na podstawie rozdziału 9.

```

% Rozdział 9 Zielinski
%                               Mateusz Krupnik

% Implementacj algorytmu RADIX-2
clear all; close all; clc;
% Sygnały
syg_test = 2           % 1 - 8 punktowy testowy, 2 - zdefinowany
% Sygnał 1
if syg_test == 1
    N=8;                % liczba próbek sygnału
    x=0:N-1;              % przykładowe wartości próbek
    f = 1:N;               % próbki
else
    N = 512;
    fpr = 1000; dt=1/fpr;
    t = dt*(0:N-1);

```

```

x = sin(2*pi*150*t)+sin(2*pi*15*t);
f = fpr*(0:N-1)/N;
end
xc = x; % kopia sygnału x

% obliczenie widma metodą zaimplementowaną w Matlabie
wid_fft = fft(xc);

% Opcja 1 - odwracanie w implementacji 1 i FFT poglądowe
syg = bitReverse1(x); % decymacja próbek
wid_1 = fft_1(syg); % Odpisanie FFT
% Obliczenie błędów i wykresy
blad_real = abs(real(wid_1-wid_fft));
blad_imag = abs(imag(wid_1-wid_fft));
figure(1); subplot(311);
plot(f, abs(wid_1), f, abs(wid_fft), 'r');
title('Widmo wynikowe i oryginalne');
legend('FFT Własne', 'FFT Matlab');
subplot(312); plot(blad_real); title('Błąd części rzeczywistej');
subplot(313); plot(blad_imag); title('Błąd części urojonej');

%% Opcja 2 - odwracanie w implementacji 2 i FFT poglądowe
syg = bitReverse2(x); % decymacja próbek
wid_2 = fft_1(syg); % Odpisanie FFT
% Obliczenie błędów i wykresy
blad_real = abs(real(wid_2-wid_fft));
blad_imag = abs(imag(wid_2-wid_fft));
figure(2); subplot(311);
plot(f, abs(wid_1), f, abs(wid_fft), 'r');
title('Widmo wynikowe i oryginalne');
legend('FFT Własne', 'FFT Matlab');
subplot(312); plot(blad_real); title('Błąd części rzeczywistej');
subplot(313); plot(blad_imag); title('Błąd części urojonej');

%% Opcja 3 - odwracanie w implementacji 1 i FFT szybkie
syg = bitReverse1(x); % decymacja próbek
wid_3 = fft_2(syg); % Odpisanie FFT
% Obliczenie błędów i wykresy
blad_real = abs(real(wid_3-wid_fft));
blad_imag = abs(imag(wid_3-wid_fft));
figure(3); subplot(311);
plot(f, abs(wid_1), f, abs(wid_fft), 'r');
title('Widmo wynikowe i oryginalne');
legend('FFT Własne', 'FFT Matlab');
subplot(312); plot(blad_real); title('Błąd części rzeczywistej');
subplot(313); plot(blad_imag); title('Błąd części urojonej');

%% Opcja 4 - odwracanie w implementacji 2 i FFT szybkie
syg = bitReverse2(x); % decymacja próbek
wid_4 = fft_2(syg); % Odpisanie FFT
% Obliczenie błędów i wykresy
blad_real = abs(real(wid_4-wid_fft));
blad_imag = abs(imag(wid_4-wid_fft));
figure(4); subplot(311);
plot(f, abs(wid_1), f, abs(wid_fft), 'r');

```

```

title('Widmo wynikowe i orginalne');
legend('FFT Własne', 'FFT Matlab');
subplot(312); plot(blad_real); title('Błąd części rzeczywistej');
subplot(313); plot(blad_imag); title('Błąd części urojonej');

%% Implementacje funkcji przestawiania próbek
function syg1=bitReverse1(syg_wej1)
N1 = length(syg_wej1); % liczba próbek
MSB=log2(N1); % liczba bitów numerów próbek
for n=0:N1-1 % kolejne próbki
    ncopy=n; % stary numer próbki (kopią)
    nr=0; % nowy numer próbki (inicjalizacja)
    for m=1:MSB % po wszystkich bitach
        if (rem(n,2)==0) % czy jedynka na LSB
            n=n/2; % jeśli nie, przesuń w prawo
        else
            nr=nr+2^(MSB-m); % dodaj 2^(MSB-m)
            n=(n-1)/2; % odejmij jedynkę, przesuń w prawo
        end
    end
    y(nr+1)=syg_wej1(ncopy+1); % skopiuj we właściwe miejsce
end
syg1 = y; % podstaw wynik pod x
end

% Metoda przestawiana w miejscu
function syg2=bitReverse2(syg_wej2)
N2 = length(syg_wej2); % liczba próbek
a=1;
for b=1:N2-1 % kolejne próbki
    if (b<a) % porównanie czy indeks jest większy
        T=syg_wej2(a);
        syg_wej2(a)=syg_wej2(b);
        syg_wej2(b)=T; % zamiana miejscami
    end
    c=N2/2; % indeks połowy próbek
    while (c<a)
        a=a-c; c=c/2; % decymacja
    end
    a=a+c;
end
syg2=syg_wej2;
end

% Obliczanie FFT - wersja poglądowa
% Wymaga podania sygnału po decymacji funkcjami powyżej
function x=fft_1(x)
Nsyg = length(x); % liczba próbek
for e = 1 : log2(Nsyg) % KOLEJNE ETAPY
    SM = 2^(e-1); % szerokość motylka
    LB = Nsyg/(2^e); % liczba bloków
    LMB = 2^(e-1); % liczba motylków w bloku
    OMB = 2^e; % odległość między blokami
    W = exp(-1i*2*pi/2^e); % podstawa bazy Fouriera
    for b = 1 : LB % KOLEJNE BLOKI
        for m = 1 : LMB % KOLEJNE MOTYLKI
            g = (b-1)*OMB + m; % indeks górnej próbki motylka
            d = g + SM; % indeks dolnej próbki motylka
            xg = x(g); % skopiowanie górnej próbki
            x(d)=xg;
        end
    end
end

```

```

        xd = x(d) * W^(m-1); % korekta dolnej próbki
        % nowa góra próbk: góra plus dolna po korekcji
        x(g) = xg + xd;
        % nowa dolna próbk: góra minus dolna po korekcji
        x(d) = xg - xd;
    end % koniec pętli motylków
end % koniec pętli bloków
end % koniec pętli etapów
end

% Obliczanie FFT - wersja szybsza
% Wymaga podania sygnału po decymacji funkcjami powyżej
function x=fft_2(x)
    Ns = length(x); % liczba próbek
    for e = 1 : log2(Ns) % KOLEJNE ETAPY
        L = 2^e; % długość bloków DFT, przesunięcie bloków
        M = 2^(e-1); % lba motylków w bloku, szerokość każdego motylka
        Wi = 1; % startowa wartość wsp. bazy w etapie
        W = cos(2*pi/L)-1i*sin(2*pi/L); % mnożnik bazy Fouriera
        for m = 1 : M % KOLEJNE MOTYLKI
            for g = m : L : Ns % W KOLEJNYCH BLOKACH
                % g „górnego”, d „dolny” indeks próbki motylka
                d = g+M;
                T2 = x(d)*Wi; % „serce” FFT
                x(d) = x(g)-T2; % nowa dolna próbk: góra minus „serce”
                x(g) = x(g)+T2; % nowa góra próbk: góra plus „serce”
            end % koniec pętli bloków
            Wi=Wi*W; % kolejna wartość bazy Fouriera
        end % koniec pętli motylków
    end
end

```

Tab. 8.8. Kod programów na podstawie rozdziału 10.

```

% Rodzial 10 Zielinski
% Mateusz Krupnik

%% Filtracja cyfrowa z wykorzystaniem buforów
%y = filter(b, a ,x); % Funkcja wbudowana Matlaba
% filterBK i filterBP - definicje poniżej

%% Projektowanie rekursywnych filtrów cyfrowych metodą zer i biegunów
clear all; clc; close all;
% Parametry do rysowania okregu
NP = 1000; fi = 2 * pi * (0:1:NP-1)/NP; s = sin(fi); c = cos(fi);
fpr = 1000; % częstotliwość próbki
% syngaly testowe
Nx = 1024; n=0:Nx-1; dt=1/fpr; t=dt*n;
f1 = 10; f2 = 50; f3 = 250;
x = sin(2*pi*f1*t) + sin(2*pi*f2*t) + sin(2*pi*f3*t);
xd = zeros(1, Nx); xd(1) = 1;
for i=1:2
    if (i == 1) % FILTR LP
        filtr = "dolnoprzepustowy";
        fz = [ 50 ]; % Częstotliwość zer
        fp = [ 10 ]; % Częstotliwość biegunów
        Rz = [ 1 ]; % współczynniki (promienie) zer
        Rp = [ 0.98 ]; % współczynniki (promienie) biegnów (!= 1)
    end

```

```

fmax = 100; df = 0.1; % widmo fouriera
else
    filtr = "srodkoprzepustowy";
    fz = [ 50 100 150 350 400 450 ]; % Czestotliwość zer
    fp = [ 200 250 300 ]; % Czestotliwość biegunów
    Rz = [ 1 ]; % współczynniki (promienie) zer
    Rp = [ 0.96 ]; % współczynniki (promienie) biegnów (!= 1)
    fmax = 500; df = 0.1; % widmo fouriera
end

% Obliczenie zer i biegunów transmitemancji H(z)
fi_z = 2*pi*(fz/fpr); % katy zer
fi_p = 2*pi*(fp/fpr); % katy biegunow
z = Rz .* exp(1i*fi_z); % zera
p = Rp .* exp(1i*fi_p); % bieguny
z = [ z conj(z)]; p = [ p conj(p)]; % dodanie par sprzężonych

% Położenie zer i biegnów
figure(6*i - 5);
plot(s, c, '-k', real(z), imag(z), 'or', real(p), imag(p), 'xb');
title(["Zera i bieguny filtra: " + filtr]); legend('Zera','Bieguny');
grid on;

% Obliczenia współczynników a i b z zer i biegnów
wzm = 1; [b, a] = zp2tf(z', p', wzm);

% Charakterystyka częstotliwościowa H(f)
% częstotliwość, częstość, częstość unormowana
f = 0 : df : fmax; w = 2*pi*f; wn = 2*pi*f/fpr;
H = freqz(b, a, wn);
Habs = abs(H); Hdb = 20*log10(Habs); Hfa = unwrap(angle(H));
figure(6*i-4);
sgtitle(["Analiza częstot. dla filtra: " + filtr]);
subplot(311); plot(f, Habs); grid on; title('|H(f)|');
xlabel('Częstotliwość f [Hz]'); ylabel('Ampl.');
subplot(312); plot(f, Hdb); grid on; title('|H(f)| [dB]');
xlabel('Częstotliwość f [Hz]'); ylabel('Ampl. [dB]');
subplot(312); plot(f, Hfa); grid on; title('angle(H(f))');
xlabel('Częstotliwość f [Hz]'); ylabel('rad.');

% Filtracja sygnału x
y1 = filter(b, a, x);
y2 = filterBP(b, a, x);
y3 = filterBK(b, a, x);

figure(6*i-3)
sgtitle(["Filtracja sygnału sinusoidalnego dla filtra: " + filtr]);
subplot(211); plot(t, x);
title('Sygnał wejściowy x'); xlabel('Czas [s]');
ylabel('Ampl.); grid on;
subplot(212); plot(t, y1, t, y2, t, y3);
title('Sygnał wyjściowy y'); xlabel('Czas [s]');
ylabel('Ampl.); grid on;
legend('Matlab filter', 'filterBP', 'filterBK');

% Filtracja sygnału xd
y1d = filter(b, a, xd);
y2d = filterBP(b, a, xd);
y3d = filterBK(b, a, xd);

```

```

figure(6*i-2)
sgtitle(["Filtracja sygnału jednostkowego dla filtra: " + filtr]);
subplot(211); plot(t, xd);
title('Sygnał wejściowy xd'); xlabel('Czas [s]');
ylabel('Ampl.');
```

```

subplot(212); plot(t, y1d, t, y2d, t, y3d);
title('Sygnał wyjściowy y'); xlabel('Czas [s]');
ylabel('Ampl.');
```

```

grid on;
legend('Matlab filter', 'filterBP', 'filterBK');
```



```
% Sygnaly w dziedzinie częstotliwości
n=Nx/2+1:Nx; X = freqz(x(n),1,wn)/(Nx/4);
Y = freqz(y1(n),1,wn)/(Nx/4);
Y(2, :) = freqz(y2(n),1,wn)/(Nx/4);
Y(3, :) = freqz(y3(n),1,wn)/(Nx/4);
X = abs(X); Y = abs(Y);
```



```
figure(6*i-1);
sgtitle(["Filtracja sygnału sinusoidalnego dla filtra: " + filtr]);
subplot(211); plot(f,X);
title('Wejście X(f)'); ylabel('Ampl.');
```

```

grid on;
subplot(212); plot(f, Y(1, :), f, Y(2, :), f, Y(3, :));
title('Wyjście Y(f)'); ylabel('Ampl.');
```

```

grid on;
xlabel('f [Hz]');
```

```

legend('Matlab filter', 'filterBP', 'filterBK');
```



```
% Sygnał impulsowy
```



```
Xd = freqz(xd(n),1,wn)/(Nx/4);
Yd = freqz(y1d(n),1,wn)/(Nx/4);
Yd(2, :) = freqz(y2d(n),1,wn)/(Nx/4);
Yd(3, :) = freqz(y3d(n),1,wn)/(Nx/4);
Xd = abs(Xd); Yd = abs(Yd);
figure(6*i);
sgtitle(["Filtracja sygnału impulsowego dla filtra: " + filtr]);
subplot(211); plot(f, Xd);
title('Wejście X(f)'); ylabel('Ampl.');
```

```

grid on;
subplot(212); plot(f, Yd(1, :), f, Yd(2, :), f, Yd(3, :));
title('Wyjście Y(f)'); ylabel('Ampl.');
```

```

grid on;
xlabel('f [Hz]');
```

```

legend('Matlab filter', 'filterBP', 'filterBK');
```

```

end
```



```
%%%%% DEFINICJE FUNKCJI %%%%%%
```



```
% Filtracji z buforami przesuwnymi
function y = filterBP(b, a, x)
Nx = length(x); % Długość sygnału x
M = length(b); N = length(a); % Długość buforów
a = a(2:N); N = N - 1; % Usunięcie a_1 = 1
bx = zeros(1, M); by = zeros(1, N); % Prealokacja buforów
y = [];
for n=1:Nx
    bx = [x(n) bx(1:M-1)];
    y(n) = sum(bx .* b) - sum(by .* a);
    by = [y(n) by(1:N-1)];
end
end
```

```
% Filtracja z buforami kolowymi
function y = filterBK(b, a, x)
    Nx = length(x); % Długość sygnału x
    M = length(b); N = length(a); % Długość buforów
    a = a(2:N); N = N - 1; % Usunięcie a_1 = 1
    bx = zeros(1, M); by = zeros(1, N); % Prealokacja buforów
    y = [];
    ix = 1; iy = 1; % wskaźniki od 1 miejsca tablic
    for n = 1 : Nx
        bx(ix) = x(n); % Bufor wejścia
        sum = 0; ib = 1; ia = 1; % wskazniki

        for k = 1 : M - 1 % Sumowanie probek wejścia
            sum = sum + bx(ix)*b(ib);
            ix = ix - 1;
            if (ix == 0) ix = M; end
            ib = ib + 1;
        end
        sum = sum + bx(ix)*b(ib); % ostatni składnik sumy

        for k = 1 : N - 1
            sum = sum - by(iy)*a(ia);
            iy = iy - 1;
            if (iy == 0) iy = N; end
            ia = ia + 1;
        end
        sum = sum - by(iy)*a(ia);

        y(n) = sum; % wynik filtracji
        by(iy) = sum; % dodanie wyjścia do bufora wyjść
    end
end
```

Tab. 8.9. Kod programów na podstawie rozdziału 11.

```
% Rozdział 11 Zielinski
% Mateusz Krupnik

% Filtry cyfrowe na podstawie filtrów analogowych z rozdziału 6.
clc; close all; clear all;
% Analogowe filtry Butterwortha i Czebyszewa
apass = 3; % nielinowość pasma przepustowego w dB
astop = 60; % tłumienie w paśmie zaprowym
fpr = 2000; % częstotliwość probkowania
fmx = 1000; % maks. składowa częstotliwości fpr/2

%% Filtry Butterwortha Hp, BP i BS - filtry oparte na okręgu w lewej
% polpł.
typ = "Butterworth ";
for i=1:4
    % filtr dolnoprzepustowy LP
    if i==1
        filtr = "LowPass ";
        fpass = 200; fstop = 300; % Hz
        fpass = fc2fa(fpass, fpr);
        fstop = fc2fa(fstop, fpr);
```

```

ws = fstop/fpass;    % częstotliwość znormalizowana. s=s'/w0, w0=2*pi*fpass

% filtr gornoprzepustowy HP
elseif i==2
    filtr = "HighPass ";
    fstop = 700; % częstotliwość przepustowa odpowiadająca astop
    fpass = 800; % częstotliwość zaporowa odpowiadająca apass
    fpass = fc2fa(fpass, fpr);
    fstop = fc2fa(fstop, fpr);
    ws = fpass/fstop; % transformacja częstotliwości: s=w0/s', w0=2*pi*fpass

% filtr srodkowoprzepustowy BP
elseif i==3
    filtr = "BandPass ";
    fs1 = 300; % dolna częstotliwość stop
    fp1 = 400; % dolna częstotliwość pass
    fp2 = 600; % górna częstotliwość pass
    fs2 = 700; % górna częstotliwość stop
    % transformacja częstotliwości
    fs1 = fc2fa(fs1, fpr);
    fp1 = fc2fa(fp1, fpr);
    fs2 = fc2fa(fs2, fpr);
    fp2 = fc2fa(fp2, fpr);
    ws1t = (fs1^2 - fp1*fp2) / (fs1*(fp2-fp1));
    ws2t = (fs2^2 - fp1*fp2) / (fs2*(fp2-fp1));
    ws = min(abs(ws1t), abs(ws2t));

% filtr srodkowozaporowy BS
else
    filtr = "BandStop ";
    fp1 = 200; % dolna częstotliwość filtra pasmowego
    fs1 = 300; % dolna częstotliwość filtra pasmowego
    fs2 = 700; % górna częstotliwość filtra pasmowego
    fp2 = 800; % górna częstotliwość filtra pasmowego

    fs1 = fc2fa(fs1, fpr);
    fp1 = fc2fa(fp1, fpr);
    fs2 = fc2fa(fs2, fpr);
    fp2 = fc2fa(fp2, fpr);
    % transformacja częstotliwości
    ws1t = (fs1*(fp2-fp1)) / (fs1^2 - fp1*fp2);
    ws2t = (fs2*(fp2-fp1)) / (fs2^2 - fp1*fp2);
    ws = min(abs(ws1t), abs(ws2t));
end

% Przeliczenie na wartość bezwzględna
wzm_p = 10^(-apass/20);
wzm_s = 10^(-astop/20);

if ( (i==1) || (i==2) )
    vp = 2*pi*fpass;
    vs = 2*pi*fstop;
    f_ps = [fpass, fstop];
    wzm_ps = [wzm_p, wzm_s];
    wzmdB_ps = [-apass, -astop];
else
    vp = 2*pi*[ fp1 fp2 ];
    vs = 2*pi*[ fs1 fs2 ];
    vc = 2*pi*sqrt(fp1*fp2);      % pulsacja środkowa
    % szerokość filtra wokół vc

```

```

f_ps = [fp1,fp2,fs1,fs2];
dv = 2*pi*(fp2-fp1);
wzm_ps = [wzm_p, wzm_p, wzm_s, wzm_s];
wzmdB_ps = [-apass, -apass, -astop, -astop];
end

disp(['num2str(i) ' - FILTR CYFROWY: ' filtr ' - ' typ]);;

% Obliczenie parametrów N i w0 - funkcja buttord
wp = 1;
N = ceil(log10( (10^(astop/10)-1) / ...
    (10^(apass/10)-1) ) / (2*log10(ws/wp)) );
w0 = ws / (10^(astop/10)-1)^(1/(2*N));

% Obliczenie biegunów trans. prototypu - funkcja buttap i zp2tf
dfi0 = (2*pi)/(2*N); % kat „kawałka tortu”
fi = pi/2 + dfi0/2 + (0 : N-1)*dfi0; % kąty biegunów
p = w0*exp(j*fi); % bieguny
z = [];% zera
wzm = real( prod(-p) ); % wzmacnienie
a = poly(p); % bieguny --> wsp wielomianu mianownika A(z)
b = wzm; % wielomian licznika B(z)
%
z, p, b, a
figure(7*i-6)
plot( real(p), imag(p), 'x' ); grid;
title(["Położenie biegunów dla filtra analogowego:" + filtr + typ]);
xlabel('real'); ylabel('imag');

% Porównanie z Matlabem
[NN,ww0] = buttord( vp, vs, apass, astop, 's' );
blad_N = N-NN; disp(['Blad rzedu: ' num2str(blad_N)]);

% Oblicz charakterystykę częstotliwościową H(w)=B(w)/A(w)
% zakres pulsacji unormowanej; pulsacja granicy pasma przepustowego =
1
w = 0 : 0.005 : 2;
H = freqs(b,a,w); % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);

figure(7*i-5); subplot(211);
plot(w,abs(H)); grid;
title(["Moduł prototypu anaglowego LowPass dla" + filtr + typ]);
xlabel('Pulsacja [rad/sek]');
subplot(212); plot(w,20*log10(abs(H))); grid;
title(["Moduł prototypu analogowego LowPass w dB dla " + filtr +
typ]);
xlabel('Pulsacja [rad/sek]'); ylabel('dB');

% Transformata częstotliwości filtra analogowego: prototyp unormowany
% --> wynikowy filtr
% LowPass to LowPass: s=s/w0
if (i==1) [z,p,wzm] = lp2lpTZ(z,p,wzm,wp); end
% LowPass to HighPass: s=w0/s
if (i==2) [z,p,wzm] = lp2hpTZ(z,p,wzm,wp); end
% LowPass to BandPass: s=(s^2+wc^2)/(dw*s)
if (i==3) [z,p,wzm] = lp2bpTZ(z,p,wzm,vc,dv); end
% LowPass to BandStop: s=(dw*s)/(s^2+wc^2)
if (i==4) [z,p,wzm] = lp2bstZ(z,p,wzm,vc,dv); end
b=wzm*poly(z); a=poly(p);

```

```
% Pokaż zera i bieguny po transformacji częstoliwości
figure(7*i-4)
plot( real(z), imag(z), 'o', real(p), imag(p), 'x' ); grid;
title(["Położenie biegunów dla filtra analogowego" + filtr + typ]);
xlabel('real'); ylabel('imag');

printsys(b,a,'s');
% Końcowa charakterystyka częstoliwościowa
NF = 1000; % ile punktów
fmin = 0; % dolna częstotliwość
fmax = 5000; % górna częstotliwość
f = fmin : (fmax-fmin)/(NF-1) : fmax; % wszystkie częstotliwości
w = 2*pi*f; % wszystkie pulsacje
H = freqs(b,a,w); % alternatywa: H = polyval( b, j*w ) ./ polyval(a, j*w);

figure(7*i-3); subplot(211);
plot( f,abs(H), f_ps, wzm_ps, 'ro' );
grid; title(["Moduł dla filtra: " + filtr + typ]);
xlabel('Częstotliwość [Hz]');
subplot(212);
plot(f,20*log10(abs(H)), f_ps, wzmdB_ps, 'ro' );
axis([fmin,fmax,-100,20]);
grid; title(["Moduł w dB filtra " + filtr + typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f,unwrap(angle(H))); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel('rad');

% Filtr cyfrowy
[zc, pc, wzmc] = bilinearTZ(z, p, wzm, fpr);
bc = wzmc*poly(zc); ac = poly(pc);

% Wykres zer filtra cyfrowego
NP = 1000; fi=2*pi*(0:1:NP-1)/NP; x=sin(fi); y=cos(fi);
figure(7*i-2)
plot(x,y, '-k', real(zc), imag(zc), 'or', real(pc), imag(pc), 'xb' );
title(["ZERA i BIEGUNY filtra cyfrowego " + filtr + typ]);
grid;

% Charakterystyka filtra cyfrowego
NF = 1000; fmin = 0; fmax = fm; f = fmin : (fmax-fmin)/(NF-1) : fmax;
w = 2*pi*f/fpr; H = freqz(bc,ac,w);
Habs=abs(H); HdB=20*log10(Habs); Hfa=unwrap(angle(H));
f_ps = (fpr/pi)*atan(pi*f_ps/fpr);

figure(7*i-1); subplot(211);
plot( f,Habs, f_ps, wzm_ps, 'ro' );
grid; title(["Moduł dla filtra cyfrowego: " + filtr + typ]);
xlabel('Częstotliwość [Hz]');
subplot(212);
plot(f, HdB, f_ps, wzmdB_ps, 'ro' ); axis([fmin,fmax,-100,20]);
grid; title(["Moduł w dB filtra cyfrowego " + filtr + typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f, Hfa); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel('rad');

% Odpowiedź filtra na delta Kroneckera
Nx=200; x = zeros(1,Nx); x(1)=1;
M=length(bc); N=length(ac);
ac=ac(2:N); N=N-1;
```

```

bx=zeros(1,M); by=zeros(1,N); y=[];
for n=1:Nx
    bx = [ x(n) bx(1:M-1)];
    y(n) = sum(bx .* bc) - sum(by .* ac);
    by = [ y(n) by(1:N-1) ];
end
n=0:Nx-1;
figure(7*i);
plot(n,y); grid;
title(["Odp. impulsowa filtra cyfrowego " + filtr + typ]);
xlabel('Probki - n'); ylabel('Ampl.');
end

%% Filtry na podstawie prototypu Czebyszewa I typu
typ = "Czebyszewa I ";
for i=1:4
    % filtr dolnoprzepustowy LP
    if i==1
        filtr = "LowPass ";
        fpass = 200; fstop = 300; % Hz
        fpass = fc2fa(fpass, fpr);
        fstop = fc2fa(fstop, fpr);
        ws = fstop/fpass;    % częstotliwość znormalizowana s=s'/w0, w0=2*pi*fpass

    % filtr gornoprzepustowy HP
    elseif i==2
        filtr = "HighPass ";
        fstop = 700; % częstotliwość przepustowej odpowiadającej astop
        fpass = 800; % częstotliwość zaporowej odpowiadającej apass
        fpass = fc2fa(fpass, fpr);
        fstop = fc2fa(fstop, fpr);
        ws = fpass/fstop; % transformacja częstotliwości s=w0/s', w0=2*pi*fpass

    % filtr srodkowoprzepustowy BP
    elseif i==3
        filtr = "BandPass ";
        fs1 = 300; % dolna częstotliwość stop
        fp1 = 400; % dolna częstotliwość pass
        fp2 = 600; % górna częstotliwość pass
        fs2 = 700; % górna częstotliwość stop
        % transformacja częstotliwości
        fs1 = fc2fa(fs1, fpr);
        fp1 = fc2fa(fp1, fpr);
        fs2 = fc2fa(fs2, fpr);
        fp2 = fc2fa(fp2, fpr);
        ws1t = (fs1^2 - fp1*fp2) / (fs1*(fp2-fp1));
        ws2t = (fs2^2 - fp1*fp2) / (fs2*(fp2-fp1));
        ws = min(abs(ws1t), abs(ws2t));

    % filtr srodkowozaporowy BS
    else
        filtr = "BandStop ";
        fp1 = 200; % dolna częstotliwość filtra pasmowego
        fs1 = 300; % dolna częstotliwość filtra pasmowego
        fs2 = 700; % górna częstotliwość filtra pasmowego
        fp2 = 800; % górna częstotliwość filtra pasmowego

        fs1 = fc2fa(fs1, fpr);
        fp1 = fc2fa(fp1, fpr);
        fs2 = fc2fa(fs2, fpr);
    end
end

```

```

fp2 = fc2fa(fp2, fpr);
% transformacja częstotliwości
ws1t = (fs1*(fp2-fp1)) / (fs1^2 - fp1*fp2);
ws2t = (fs2*(fp2-fp1)) / (fs2^2 - fp1*fp2);
ws = min(abs(ws1t), abs(ws2t));
end

% Przeliczenie na wartość bezwzględna
wzm_p = 10^(-apass/20);
wzm_s = 10^(-astop/20);

if ( (i==1) || (i==2))
    vp = 2*pi*fpass;
    vs = 2*pi*fstop;
    f_ps = [fpass, fstop];
    wzm_ps = [wzm_p, wzm_s];
    wzmdB_ps = [-apass, -astop];
else
    vp = 2*pi*[ fp1 fp2 ];
    vs = 2*pi*[ fs1 fs2 ];
    vc = 2*pi*sqrt(fp1*fp2);      % pulsacja środka
    % szerokość filtra wokół vc
    f_ps = [fp1,fp2,fs1,fs2];
    dv = 2*pi*(fp2-fp1);
    wzm_ps = [wzm_p, wzm_p, wzm_s, wzm_s];
    wzmdB_ps = [-apass, -apass, -astop, -astop];
end

disp(['num2str(i) ' - FILTR CYFROWY: ' filtr ' - ' typ']);

% Obliczenie parametrów N i w0
wp = 1;
Nreal = acosh(sqrt((10^(astop/10)-1) / ...
(10^(apass/10)-1))) / acosh(ws/wp);
N = ceil(Nreal);
epsi = sqrt(10^(apass/10)-1);
D = asinh(1/epsi)/N; R1 = sinh(D); R2 = cosh(D);

% Obliczenie biegunów trans. prototypu - funkcja buttap i zp2tf
dfi0 = (2*pi)/(2*N);                      % kąt „kawałka tortu”
fi = pi/2 + dfi0/2 + (0 : N-1)*dfi0;       % kąty biegunów
p1 = R1 * exp(j*fi);                      % bieguny R1
p2 = R2 * exp(j*fi);                      % bieguny R2
p = real(p1) +li*imag(p2);                 % Polaczono bieguny
z = [];                                     % zera
wzm = prod(-p);                            % wzmacnianie
a = poly(p);                                % bieguny --> wsp wielomianu mianownika A(z)
b = wzm;                                     % wielomian licznika B(z)
if (rem(N,2)==0) b = b*10^(-apass/20); end

figure(7*i-6+28)
plot( real(p), imag(p), 'x' ); grid;
title(["Położenie biegunów dla filtra analogowego:" + filtr + typ]);
xlabel('real'); ylabel('imag');

% Porównanie z Matlabem
[NN,ww0] = cheblord( vp, vs, apass, astop, 's' );
blad_N = N-NN; disp(['Blad rzedu: ' num2str(blad_N)]);

```

```

% Oblicz charakterystykę częstotliwościową H(w)=B(w)/A(w)
% zakres pulsacji unormowanej; pulsacja granicy pasma przepustowego =
1
w = 0 : 0.005 : 2;
H = freqs(b,a,w); % alternatywa: H=polyval( b,j*w )./polyval(a,j*w);

figure(7*i-5+28); subplot(211);
plot(w,abs(H)); grid;
title(["Moduł prototypu anaglowego LowPass dla " + filtr + typ]);
xlabel('Pulsacja [rad/sek]');
subplot(212); plot(w,20*log10(abs(H))); grid;
title(["Moduł prototypu analogowego LowPass w dB dla " + filtr +
typ]);
xlabel('Pulsacja [rad/sek]'); ylabel('dB');

% Transformata częstotliwości filtra analogowego: prototyp unormowany
% --> wynikowy filtr
% LowPass to LowPass: s=s/w0
if (i==1) [z,p,wzm] = lp2lpTZ(z,p,wzm,vp); end
% LowPass to HighPass: s=w0/s
if (i==2) [z,p,wzm] = lp2hpTZ(z,p,wzm,vp); end
% LowPass to BandPass: s=(s^2+wc^2)/(dw*s)
if (i==3) [z,p,wzm] = lp2bpTZ(z,p,wzm,vc,dv); end
% LowPass to BandStop: s=(dw*s)/(s^2+wc^2)
if (i==4) [z,p,wzm] = lp2bstZ(z,p,wzm,vc,dv); end
b=wzm*poly(z); a=poly(p);

% Pokaż zera i bieguny po transformacji częstotliwości
figure(7*i-4+28)
plot( real(z), imag(z), 'o', real(p), imag(p), 'x' ); grid;
title(["Położenie biegunów dla filtra analogowego" + filtr + typ]);
xlabel('real'); ylabel('imag');

printsys(b,a,'s');
% Końcowa charakterystyka częstotliwościowa
NF = 1000; % ile punktów
fmin = 0; % dolna częstotliwość
fmax = 5000; % górna częstotliwość
f = fmin : (fmax-fmin)/(NF-1) : fmax; % wszystkie częstotliwości
w = 2*pi*f; % wszystkie pulsacje
H = freqs(b,a,w); % alternatywa: H = polyval( b,j*w )./polyval(a,j*w);

figure(7*i-3+28); subplot(211);
plot( f,abs(H), f_ps, wzm_ps, 'ro' );
grid; title(["Moduł dla filtra: " + filtr + typ]);
xlabel('Częstotliwość [Hz]');
subplot(212);
plot(f,20*log10(abs(H)), f_ps, wzmdB_ps, 'ro' );
axis([fmin,fmax,-100,20]);
grid; title(["Moduł w dB filtra " + filtr + typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f,unwrap(angle(H))); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel(' [rad]');

% Filtr cyfrowy
[zc, pc, wzmc] = bilinearTZ(z, p, wzm, fpr);
bc = wzmc*poly(zc); ac = poly(pc);

% Wykres zer filtra cyfrowego

```

```

NP = 1000; fi=2*pi*(0:1:NP-1)/NP; x=sin(fi); y=cos(fi);
figure(7*i-2+28)
plot(x,y,'-k',real(zc),imag(zc),'or',real(pc),imag(pc),'xb');
title(["ZERA i BIEGUNY filtra cyfrowego " + filtr + typ]);
grid;

% Charakterystyka filtra cyfrowego
NF = 1000; fmin = 0; fmax = fm; f = fmin : (fmax-fmin)/(NF-1) : fmax;
w = 2*pi*f/fpr; H = freqz(bc,ac,w);
Habs=abs(H); HdB=20*log10(Habs); Hfa=unwrap(angle(H));
f_ps = (fpr/pi)*atan(pi*f_ps/fpr);

figure(7*i-1+28); subplot(211);
plot(f,Habs, f_ps, wzm_ps, 'ro');
grid; title(["Moduł dla filtra cyfrowego: " + filtr + typ]);
xlabel('Częstotliwość [Hz]');
subplot(212);
plot(f, HdB, f_ps, wzm_ps, 'ro'); axis([fmin,fmax,-100,20]);
grid; title(["Moduł w dB filtra cyfrowego " + filtr + typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f, Hfa); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel('rad');

% Odpowiedz filtra na delte Kroneckera
Nx=200; x = zeros(1,Nx); x(1)=1;
M=length(bc); N=length(ac);
ac=ac(2:N); N=N-1;
bx=zeros(1,M); by=zeros(1,N); y=[];
for n=1:Nx
    bx = [ x(n) bx(1:M-1)];
    y(n) = sum(bx .* bc) - sum(by .* ac);
    by = [ y(n) by(1:N-1) ];
end
n=0:Nx-1;
figure(7*i+28);
plot(n,y); grid;
title(["Odp. impulsowa filtra cyfrowego " + filtr + typ]);
xlabel('Probki - n'); ylabel('Ampł.');
end

%% Filtry na podstawie prototypu Czebyszewa II typu
typ = "Czebyszewa II ";
for i=1:4
    % filtr dolnoprzepustowy LP
    if i==1
        filtr = "LowPass ";
        fpass = 200; fstop = 300; % Hz
        fpass = fc2fa(fpass, fpr);
        fstop = fc2fa(fstop, fpr);
        ws = fstop/fpass; % częstotliwość znormalizowana s=s'/w0, w0=2*pi*fpass
    % filtr gornoprzepustowy HP
    elseif i==2
        filtr = "HighPass ";
        fstop = 700; % częstotliwość przepustowego odpowiadająca astop
        fpass = 800; % częstotliwość zaporowego odpowiadająca apass
        fpass = fc2fa(fpass, fpr);
        fstop = fc2fa(fstop, fpr);
        ws = fpass/fstop; % transformacja częstotliwości s=w0/s', w0=2*pi*fpass
    % filtr srodkowoprzepustowy BP
    elseif i==3

```

```

filtr = "BandPass ";
fs1 = 300; % dolna częstotliwość stop
fp1 = 400; % dolna częstotliwość pass
fp2 = 600; % górna częstotliwość pass
fs2 = 700; % górna częstotliwość stop
% transformacja częstotliwości
fs1 = fc2fa(fs1, fpr);
fp1 = fc2fa(fp1, fpr);
fs2 = fc2fa(fs2, fpr);
fp2 = fc2fa(fp2, fpr);
ws1t = (fs1^2 - fp1*fp2) / (fs1*(fp2-fp1));
ws2t = (fs2^2 - fp1*fp2) / (fs2*(fp2-fp1));
ws = min(abs(ws1t), abs(ws2t));
% filtr środkowozaporowy BS
else
    filtr = "BandStop ";
    fp1 = 200; % dolna częstotliwość filtra pasmowego
    fs1 = 300; % dolna częstotliwość filtra pasmowego
    fs2 = 700; % górna częstotliwość filtra pasmowego
    fp2 = 800; % górna częstotliwość filtra pasmowego

    fs1 = fc2fa(fs1, fpr);
    fp1 = fc2fa(fp1, fpr);
    fs2 = fc2fa(fs2, fpr);
    fp2 = fc2fa(fp2, fpr);
    % transformacja częstotliwości
    ws1t = (fs1*(fp2-fp1)) / (fs1^2 - fp1*fp2);
    ws2t = (fs2*(fp2-fp1)) / (fs2^2 - fp1*fp2);
    ws = min(abs(ws1t), abs(ws2t));
end
% Przeliczenie na wartość bezwzględna
wzm_p = 10^(-apass/20);
wzm_s = 10^(-astop/20);

if ( (i==1) || (i==2))
    vp = 2*pi*fpass;
    vs = 2*pi*fstop;
    f_ps = [fpass, fstop];
    wzm_ps = [wzm_p, wzm_s];
    wzmdB_ps = [-apass, -astop];
else
    vp = 2*pi*[ fp1 fp2 ];
    vs = 2*pi*[ fs1 fs2 ];
    vc = 2*pi*sqrt(fp1*fp2); % pulsacja środka
    % szerokość filtra wokół vc
    f_ps = [fp1,fp2,fs1,fs2];
    dv = 2*pi*(fp2-fp1);
    wzm_ps = [wzm_p, wzm_p, wzm_s, wzm_s];
    wzmdB_ps = [-apass, -apass, -astop, -astop];
end

disp([num2str(i) ' - FILTR CYFROWY: ' filtr ' - ' typ]);

% Obliczenie parametrów N i w0
wp = 1;
Nreal = acosh(sqrt((10^(astop/10)-1) / (10^(apass/10)-1))) /
acosh(ws/wp);
N = ceil(Nreal);
epsi = sqrt(1 / (10^(apass/10)-1));
D = asinh(1/epsi)/N; R1 = sinh(D); R2 = cosh(D);

```

```
% Obliczenie biegunów trans. prototypu - funkcja buttap i zp2tf
dfi0 = (2*pi)/(2*N); % kąt „kawałka tortu”
fi = pi/2 + dfi0/2 + (0 : N-1)*dfi0; % katy biegunow
p1 = R1 * exp(j*fi); % biegyny R1
p2 = R2 * exp(j*fi); % biegyny R2
p = real(p1) +1i*imag(p2); % Polaczzone biegyny
z = 1i*sin(fi) % zera
wzm = prod(-z) / prod(-p); % wzmacnienie
z = 1./z; p = 1./p;
a = poly(p); % biegyny --> wsp wielomianu mianownika A(z)
b = wzm*poly(z); % wielomian licznika B(z)

figure(7*i-6+56)
plot( real(p), imag(p), 'x' ); grid;
title(["Położenie biegunów dla filtra analogowego:" + filtr + typ]);
xlabel('real'); ylabel('imag');

% Porównanie z Matlabem
[NN,ww0] = cheb2ord( vp, vs, apass, astop, 's' );
blad_N = N-NN; disp(['Blad rzedu:' num2str(blad_N)]);

% Oblicz charakterystykę częstotliwościową H(w)=B(w)/A(w)
% zakres pulsacji unormowanej; pulsacja granicy pasma przepustowego =
1
w = 0 : 0.005 : 2;
H = freqs(b,a,w); % alternatywa: H = polyval(
b,j*w)./polyval(a,j*w);

figure(7*i-5+56); subplot(211);
plot(w,abs(H)); grid; title(["Moduł prototypu anaglowego LowPass dla"
+ filtr + typ]);
xlabel('Pulsacja [rad/sek]');
subplot(212); plot(w,20*log10(abs(H))); grid;
title(["Moduł prototypu analogowego LowPass w dB dla " + filtr +
typ]);
xlabel('Pulsacja [rad/sek]'); ylabel('dB');

% Transformata częstotliwości filtra analogowego: prototyp unormowany
% --> wynikowy filtr LowPass to LowPass: s=s/w0
if (i==1) [z,p,wzm] = lp2lpTZ(z,p,wzm,vp); end
% LowPass to HighPass: s=w0/s
if (i==2) [z,p,wzm] = lp2hpTZ(z,p,wzm,vp); end
% LowPass to BandPass: s=(s^2+wc^2)/(dw*s)
if (i==3) [z,p,wzm] = lp2bpTZ(z,p,wzm,vc,dv); end
% LowPass to BandStop: s=(dw*s)/(s^2+wc^2)
if (i==4) [z,p,wzm] = lp2bstZ(z,p,wzm,vc,dv); end
b=wzm*poly(z); a=poly(p);

% Pokaż zera i biegyny po transformacji częstotliwości
figure(7*i-4+56)
plot( real(z), imag(z), 'o', real(p), imag(p), 'x' ); grid;
title(["Położenie biegunów dla filtra analogowego" + filtr + typ]);
xlabel('real'); ylabel('imag');

printsys(b,a,'s');
% Konicowa charakterystyka częstotliwościowa
```

```

NF = 1000;      % ile punktów
fmin = 0;        % dolna częstotliwość
fmax = 5000;     % górna częstotliwość
f = fmin : (fmax-fmin)/(NF-1) : fmax; % wszystkie częstotliwości
w = 2*pi*f;     % wszystkie pulsacje
H = freqs(b,a,w); % alternatywa: H = polyval( b, j*w ) ./ polyval( a, j*w );

figure(7*i-3+56); subplot(211);
plot( f,abs(H), f_ps, wzm_ps,'ro' );
grid; title(["Moduł dla filtra: " + filtr + typ]);
xlabel('Częstotliwość [Hz]');

subplot(212);
plot(f,20*log10(abs(H)), f_ps, wzmdB_ps,'ro');
axis([fmin,fmax,-100,20]);
grid; title(["Moduł w dB filtra " + filtr + typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f,unwrap(angle(H))); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel(' [rad]');

% Filtre cyfrowy
[ zc, pc, wzmc ] = bilinearTZ(z, p, wzm, fpr);
bc = wzmc*poly(zc); ac = poly(pc);

% Wykres zer filtra cyfrowego
NP = 1000; fi=2*pi*(0:1:NP-1)/NP; x=sin(fi); y=cos(fi);
figure(7*i-2+56)
plot(x,y,'-k',real(zc),imag(zc), 'or', real(pc),imag(pc), 'xb');
title(["ZERA i BIEGUNY filtra cyfrowego " + filtr + typ]);
grid;

% Charakterystyka filtra cyfrowego
NF = 1000; fmin = 0; fmax = fm; f = fmin : (fmax-fmin)/(NF-1) : fmax;
w = 2*pi*f/fpr; H = freqz(bc,ac,w);
Habs=abs(H); HdB=20*log10(Habs); Hfa=unwrap(angle(H));
f_ps = (fpr/pi)*atan(pi*f_ps/fpr);

figure(7*i-1+56); subplot(211);
plot( f,Habs, f_ps, wzm_ps,'ro' );
grid; title(["Moduł dla filtra cyfrowego: " + filtr + typ]);
xlabel('Częstotliwość [Hz]');

subplot(212);
plot(f, HdB, f_ps, wzmdB_ps,'ro'); axis([fmin,fmax,-100,20]);
grid; title(["Moduł w dB filtra cyfrowego " + filtr + typ]);
xlabel('Częstotliwość [Hz]'); ylabel('dB');
plot(f, Hfa); grid; title('Faza');
xlabel('Częstotliwość [Hz]'); ylabel(' [rad]');

% Odpowiedź filtra na delta Kroneckera
Nx=200; x = zeros(1,Nx); x(1)=1;
M=length(bc); N=length(ac);
ac=ac(2:N); N=N-1;
bx=zeros(1,M); by=zeros(1,N); y=[];
for n=1:Nx
    bx = [ x(n) bx(1:M-1)];
    y(n) = sum(bx .* bc) - sum(by .* ac);
    by = [ y(n) by(1:N-1) ];
end
n=0:Nx-1;
figure(7*i+56);
plot(n,y); grid;

```

```

title(["Odp. impulsowa filtra cyfrowego " + filtr + typ]);
xlabel('Probki - n'); ylabel('Ampl.');
end

%%%%% DEFINICJE FUNKCJI %%%%%%%

% Transformacja biliniowa filtru analogowego do cyfrowego
function [zz, pp, wzm] = bilinearTZ(z, p, wzm, fpr)
pp = []; zz = [];
for k=1:length(z)
    zz = [ zz (2*fpr+z(k))/(2*fpr-z(k)) ];
    wzm = wzm*(2*fpr-z(k));
end
for k=1:length(p)
    pp = [ pp (2*fpr+p(k))/(2*fpr-p(k)) ];
    wzm = wzm/(2*fpr-p(k));
end
l1 = length(p) - length(z);
l2 = length(z) - length(p);
if (l1 > 0) zz = [ zz -1*ones(1, l1) ]; end
if (l2 > 0) pp = [ pp -1*ones(1, l2) ]; end
end

% PRzeliczenie częstotliwości cyfrowej na analogową
function fn = fc2fa(f, fpr)
fn = 2*fpr*tan(pi*f/fpr)/(2*pi);
end

%%% Funkcje z rozdz 6.

% Funkcja transformująca filtr LP znromalizowany na wymagany LP
function [zz,pp,wzm] = lp2lpTZ(z,p,wzm,w0)
% LowPass to LowPass TZ
zz = []; pp = [];
for k=1:length(z)
    zz = [ zz z(k)*w0 ];
    wzm = wzm/w0;
end
for k=1:length(p)
    pp = [ pp p(k)*w0 ];
    wzm = wzm*w0;
end
end

% Funkcja transformująca filtr LP znromalizowany na wymagany HP
function [zz,pp,wzm] = lp2hpTZ(z,p,wzm,w0)
% LowPass to HighPass TZ
zz = []; pp = [];
for k=1:length(z)
    zz = [ zz w0/z(k) ];
    wzm = wzm*(-z(k));
end
for k=1:length(p)
    pp = [ pp w0/p(k) ];
    wzm = wzm/(-p(k));
end
for k=1:(length(p)-length(z))
    zz = [ zz 0 ];
end
end

```

```
% Funkcja transformująca filtr LP znromalizowany na wymagany BP
function [zz,pp,wzm] = lp2bpTZ(z,p,wzm,w0,dw)
    % LowPass to BandPass TZ
    pp = []; zz = [];
    for k=1:length(z)
        zz = [ zz roots([ 1 -z(k)*dw w0^2])' ];
    wzm = wzm/dw;
    end
    for k=1:length(p)
        pp = [ pp roots([ 1 -p(k)*dw w0^2])' ];
        wzm = wzm*dw;
    end
    for k=1:(length(p)-length(z))
        zz = [ zz 0 ];
    end
end

% Funkcja transformująca filtr LP znromalizowany na wymagany BS
function [zz,pp,wzm] = lp2bsTZ(z,p,wzm,w0,dw)
    % LowPass to BandStop TZ
    zz = []; pp = [];
    for k=1:length(z)
        zz = [ zz roots([ 1 -dw/z(k) w0^2])' ];
        wzm = wzm*(-z(k));
    end
    for k=1:length(p)
        pp = [ pp roots([ 1 -dw/p(k) w0^2])' ];
        wzm = wzm/(-p(k));
    end
    for k=1:(length(p)-length(z))
        zz = [ zz roots([ 1 0 w0^2])' ];
    end
end
```

Tab. 8.10. Kod programów na podstawie rozdziału 12.

```
% Rodzaj 12 Zielinski
%                                         Mateusz Krupnik

clc; close all; clear all;
% Zadanie 1
% Ćwiczenie: Projektowanie nierekursywnych filtrów
% cyfrowych metodą próbkowania w dziedzinie częstotliwości

N = 41; % długosc filtra
for i=1:4
    if (i==1 | i==3)
        M = (N -1) / 2;
        M2 = M/2;
        M4 = M/4;
        nn = N;
        Z4 = zeros(1, M4); J4 = ones(1, M4);
    else
        M = (N -1) / 2;
```

```

M2 = M / 2;
nn = 2 * M;
end

z2 = zeros(1, M2); J2 = ones(1, M2);

if i==1
    Ar = [ 1 J2 z2 z2 J2];
    Ai = zeros(1, nn);
    typ = "I";
elseif i==2
    Ar = [J2 z2 z2 J2];
    Ai = zeros(1, nn);
    typ = "II";
elseif i==3
    Ar = zeros(1, nn);
    Ai = [0 Z4 J2 Z4 Z4 -J2 Z4];
    typ = "III";
else
    Ar = zeros(1, nn);
    Ai = [Z4 J2 Z4 Z4 -J2 Z4];
    typ = "IV";
end
A = Ar + li*Ai;

n = 0 : nn-1; f = n/nn; h = zeros(1, nn);
for k=0:nn-1
    h = h + A(k+1)*exp(li*2*pi*k/nn*(n-M));
end
h = real(h/nn);
ho = h.*blackman(nn)';

% Wykresy filtrów
figure(2*i-1)
title(["Odp. impulsowa " + typ]);
stem(n, h); hold on; stem(n, ho); hold off;
legend('Okno prost.', 'Okno Blackmana');

NF = 500; k=0:NF-1; fn=k/NF; wn=2*pi*fn;
for k=0:NF-1
    temp = exp(-li*2*pi*k/NF*(n-M))
    H(k+1) = temp * h';
    Ho(k+1) = temp * ho';
end
figure(i);
% dla filtra typu I i II
if (i==1 | i==2) Ax = Ar; Hx=real(H); Hxo=real(Ho); end
% dla filtra typu III i IV
if (i==3 | i==4) Ax = Ai; Hx=imag(H); Hxo=imag(Ho); end
figure(2*i)
sgtitle("Charakterystyki filtra typu " + typ);
subplot(211);
plot(f, Ax, 'ob', fn, Hx, fn, Hxo);
grid; title('real(H) lub imag(H)');
legend('Wymagania', 'Okno prost.', 'Okno Blackmana');
subplot(212);
plot(fn,20*log10(abs(H)), fn, 20*log10(abs(Ho)));
grid; title('Moduł |H| w dB');
xlabel('Częstot. [Hz]'); ylabel('Ampl.');
legend('Okno prost.', 'Okno Blackmana');

```

```

end

%% Ćwiczenie: Projektowanie nierekursywnych filtrów cyfrowych

% Projektowanie filtrów FIR metoda WAŻONEJ minimalizacji błędu
% średniokwadratowego pomiędzy zadaną charakterystyką, spróbkowaną w
% dziedzinie częstotliwości, a charakterystyką otrzymywana

% Wymagania dla filtra cyfrowego
M = 20; % połowa długości filtra N = 2*M + 1
K = 50; % liczba punktów charakterystyki >2M
% Wymagania częstotliwościowe, Ak
L1 = floor(K/4); % podział na 4
% jedynki, przesjściowe punkty, zera, przejściowe punkty, jedynki
Ak = [ ones(1, L1) 0.6 0.4 zeros(1, K-(2*L1-1)-4) 0.4 0.6 ones(1, L1-1)]';

% Wagi punktów charakterystyki
wp = 1; % waga odcinka Pass
wt = 1; % waga odcinka przejściowego
ws = 1; % waga odcinka Stop
% Wektor wag
w = [wp*ones(1, L1) wt wt ws*ones(1, K-(2*L1-1)-4) wt wt wp*ones(1, L1-1)];
W = diag(w);

% Wyznaczanie macierzy F -> W*F*h=W* (Ak + e)
F = [];
n = 0:M-1;
for k = 0 : K - 1
    F = [F; 2*cos(2*pi*(M-n)*k/K) 1];
end
% wyznaczenie odp impulsowej
h = (W*F) \ (W*Ak);
h = [ h; h(M:-1:1) ];
% Wykresy
n = 0:2*M;
figure(57);
stem(n, h); grid on;
title('Odpowiedz impulsowa filtra');
xlabel('Probki n'); ylabel('Ampl.');
NF = 500; wn = 0:pi/(NF-1):pi; fn = wn/(2*pi);
H = freqz(h, 1, wn);
figure(58);
sgtitle('Char. filtra cyfrowego metoda ważonej minimalizacji błędu');
subplot(311); plot(fn, abs(H)); grid; title('Moduł odpowiedzi częst.');
xlabel('Częstotliwość znormalizowana [Hz]'); ylabel('Ampl.');
subplot(312); plot(fn, 180/pi*unwrap(angle(H)));
grid; title('Faza odpowiedzi częst.');
xlabel('Częstotliwość znormalizowana [Hz]'); ylabel('deg.');
subplot(313); plot(fn, 20*log10(abs(H)));
grid; title('Moduł odpowiedzi częst.');
xlabel('Częstotliwość znormalizowana [Hz]'); ylabel('Ampl. [dB]');

%% Ćwiczenie: Projektowanie nierekursywnych filtrów cyfrowych w
% dziedzinie częstotliwości metoda
% aproksymacji Czebyszewa (algorytm Remeza)

L = 20; % liczba współczynników N=2L-1
Nr = 5; % szerokość pasma przepustowego 0 < Nr < L

```

```

wp = 1; ws = 1; %wagi pasm Pass i Stop
R = 200;      % Zbior testowy/zbio ekstremow
tol = 10^-8;    % tolerancja bledu

% Parametry
M = L + 1;  % liczba czestotliwosci ekstremow
K = 1+R*(M-1); % Liczba badanych czestotliwosci
fz = (0 : K-1)/(K-1); % zbior czestotliwosci znorm.
k11 = 1; k12 = 1+Nr*R; % granice pasma Pass
k21 = 1+(Nr+1)*R; k22 = K; % granice pasma Stop
K1 = 1+Nr*R+R/2;           % nr probki dla czestot. granicznej
fd = [fz(1:K1) fz(K1:K)]; % czest. charakter filtra
Hd = [ones(1, K1) zeros(1, K-K1)]; % wzmacnienia
Wg = [wp*ones(1, K1) ws*ones(1, K-K1)]; % wagi
i_maximum = 1:R:K;          % inkdesy czestotliwosci ekstremow

% Wybor startowego zbioru czestotliwości ekstremów
feMAX = fz(1:R:K);
sigmaMAX = 10^15; sigma = 0;

% obliczenia
n = 0 : L-1;
while(sigmaMAX-sigma > tol)
    disp(['Zmniejszenie bledu: ' num2str(sigmaMAX-sigma)]);
    H = Hd(i_maximum); W = Wg(i_maximum);
    fe = feMAX;
    % macierz cosinusów
    A = [];
    for m = 0:M-1
        A = [A; cos(pi*fe(m+1)*n) ((-1)^m/W(m+1)) ];
    end
    % rownanie wspol c
    c = A\H;
    h = c(1:L); sigma=abs(c(M));
    g = h'/2; g(1)=2*g(1); g = [fliplr(g(2:L)) g];
    figure(59);
    sgttitle('Odp impulsowa i czestotliwosciowa filtra');
    subplot(221); stem(h); title('Odp impulsowa - polowa');
    subplot(222); stem(g); title('Odp impulsowa - cala');

    for k=0:K-1
        H(k+1) = cos(pi*fz(k+1)*n)* h;
        Herr(k+1) = Wg(k+1) * (H(k+1) - Hd(k+1));
    end
    subplot(223); plot(fz, Hd, 'r', fz, H, 'b'); grid;
    title('Charakterystyka czestotliwosciowa'); ylabel('Ampl.');
    subplot(224); plot(fz, Herr); grid;
    title('Charakterystyka czestotliwosciowa - blad'); ylabel('Ampl.');

% szukanie ekstremow
Hmax= []; i_maximum = [];
for p = 1 : 2
    if (p==1) k1=k11; k2=k12; end
    if (p==2) k1=k21; k2=k22; end
    Hmax = [Hmax Herr(k1)]; i_maximum = [i_maximum k1];
    k = k1 + 1;
    while(Herr(k-1) == Herr(k)) k = k + 1; end
    if (Herr(k) < Herr(k+1))
        sgn = 1;
    else

```

```

        sgn = -1;
    end
    k=k+1;
    while(k<=k2)
        if (sgn==1)
            while( (k<k2) & (Herr(k-1) < Herr(k))) k = k + 1; end
        end
        if (sgn== -1)
            while( (k<k2) & (Herr(k-1) > Herr(k))) k = k + 1; end
        end
        sgn = -sgn;
        Hmax = [Hmax Herr(k)]; i_maximum = [i_maximum k];
        k=k+1;
    end
end
figure(60);
subplot(211);
plot(fz(i_maximum),Hmax,'or',fz,Herr,'b');
grid; title('Błąd charakterystyki i jego ekstremów');

% Wybranie M+1 największych
if length(Hmax)>M
    IM = []; G = abs(Hmax); LenG = length(G);
    while( LenG > 0)
        Gmx = max(G); imx = find(G==Gmx);
        LenGmx = length(imx);
        IM = [ IM i_maximum(imx)];
        G(imx) = 0; LenG = LenG - LenGmx;
    end
    IM = IM(1:M); IM = sort(IM); i_maximum = IM;
end
sigmaMAX = max(abs(Hmax));
feMAX = fz(i_maximum);
subplot(212);
plot(fz(i_maximum),Herr(i_maximum),'or',fz,Herr,'b');
grid; title('Błąd charakterystyki i M+1 największych ekstremów');
pause(5);

end

fz = fz/2;
figure(61);
subplot(211); stem(g); title('Wynikowa odp impulsowa filtra');
subplot(212); plot(fz(i_maximum),Herr(i_maximum),'or',fz,Herr,'b');
grid; title('Błąd H(f) + jego EKSTREMA');
figure(62);
subplot(211);
plot(fz,Hd,'r',fz,H,'b'); grid; title('Wynikowe H(f)');
subplot(212);
plot(fz,20*log10(H),'b'); grid; title('Wynikowe H(f) w dB');

%% Ćwiczenie: Projektowanie nierekursywnych filtrów cyfrowych metodą
% okien z zastosowaniem okna Kaisera
% Podaj parametry filtru (np. pasmowozaporowego)
fpr = 1000; % częstotliwość próbkowania [Hz]
fd1 = 150; % częstotliwość dolna 1 [Hz]
fd2 = 200; % częstotliwość dolna 2 [Hz]
fg1 = 300; % częstotliwość górnna 1 [Hz]
fg2 = 350; % częstotliwość górnna 2 [Hz]
dp = 0.001; % oscylacje w paśmie przepustowym np. 0.1, 0.01, 0.001

```

```

ds = 0.0001; % oscylacje w paśmie zaporowym np. 0.001, 0.001, 0.0001
typ = ["LowPass", "HighPass", "BandPass", "BandStop"];
% lp=LowPass, hp=HighPass, bp=BandPass, bs=BandStop

for i=1:length(typ)
    if i==1
        df=fd2-fd1; % filtr low pass
        fc=((fd1+fd2)/2)/fpr;
        wc=2*pi*fc;
    end
    if i==2
        df=fg2-fg1; % filtr high pass
        fc=((fg1+fg2)/2)/fpr;
        wc=2*pi*fc;
    end
    if (i==3 || i==4)
        df1=fd2-fd1; df2=fg2-fg1; % filtry pasmowe
        df = min(df1, df2);
        f1 = (fd1+df/2)/fpr;
        f2 = (fg2-df/2)/fpr;
        w1 = 2*pi*f1; w2 = 2*pi*f2;
    end
    d = min(dp, ds); A = -20*log10(d);
    % wzory na okno Kaisera w zależności od tlumienia
    if (A>=50) beta = 0.1102*(A-8.7); end
    if (A>21 & A<50) beta = (0.5842*(A-21)^0.4)+0.07886*(A-21); end
    if (A<=21) beta = 0; end
    if (A>21) D = (A-7.95)/14.36; end
    if (A<=21) D = 0.922; end
    % Wyznaczenie dlugosci
    N = ceil((D*fpr/df)+1); if (rem(N,2)==0) N=N+1; end
    M = (N-1)/2; m = 1:M; n=1:N; % wektory indeksowe
    % Generacja okna czasowego
    temp = beta * sqrt(1-((n-1-M).^2./M.^2));
    wb = besseli(0, temp) / besseli(0,beta);
    figure(63+2*i-1); subplot(311); plot(n, wb); grid;
    title(["Okno czas. Kaisera M=" + num2str(M) " Filtr: " + typ(i)]);

    % Tworzenie odp. impulsowej, wzory z tabelki
    % filtr LP
    if (i==1) h=2*fc*sin(wc*m)./(wc*m); h=[ fliplr(h) 2*fc h]; end
    % filtr HP
    if (i==2) h=-2*fc*sin(wc*m)./(wc*m); h=[ fliplr(h) 1-2*fc h]; end
    % filtr BP
    if (i==3)
        h = 2*f2*sin(w2*m)./(w2*m) - 2*f1*sin(w1*m)./(w1*m);
        h = [ fliplr(h) 2*(f2-f1) h];
    end
    % filtr BS
    if (i==4)
        h = 2*f1*sin(w1*m)./(w1*m) - 2*f2*sin(w2*m)./(w2*m);
        h = [ fliplr(h) 1+2*(f1-f2) h];
    end
    subplot(312); plot(n,h); grid;
    title(["Odp impulsowa filtra: " + typ(i)]);

    % Wymnożenie odp impulsowej z oknem
    hw = h.*wb;
    subplot(313); plot(n,hw,'b'); grid;
    title(["Iloczyn okna i odp. impulsowej filtra: " + typ(i)]);

```

```

% Charakterystyka częstotliwościowa
NF = 1000; fmin = 0; fmax = fpr/2; % wartości parametrów char.
f = fmin : (fmax-fmin)/(NF-1) : fmax; % częstotliwość
w = 2*pi*f/fpr; % pulsacja
HW = freqz(hw,1,w);
figure(63+2*i);
sgtitle(['Charakterystyka częstotliwościowa filtra: ' typ(i)]);
subplot(211); plot(f, abs(HW)); grid; ylabel('Ampl.');
xlabel('Częstotliwość [Hz]');
title('Moduł odp. częstotliwościowej');
subplot(212); plot(f, unwrap(angle(HW))); grid; ylabel('rad.');
xlabel('Częstotliwość [Hz]');
title('Faza odp. częstotliwościowej');

end

%% Algorytm interpolacji sygnału za pomocą dyskretnej transformacji
% Fouriera DFT (FFT)
M=24; N=16; n=0:N-1; x=sin(2*pi/8*n);
X = fft(x); % obliczenie fft sygnału rzeczywistego
% Wstawienie do widma w środku symetrii zer, skrajnie zer obliczyć
% jak 0.5 bo składowa została "rozbita"
X = [ X(1:N/2) 0.5*X(N/2+1) zeros(1,M-N-1) conj(0.5*X(N/2+1))
X(N/2+2:N)];
y = M/N*real(ifft(X));
figure(72); sgtitle('Filtr interpolujacy');
subplot(211); stem(x); title('Orginalny'); % sygnał wejściowy
subplot(212); stem(y); title('Interpolowany'); % interpol. sygnał wej.

%% Ćwiczenie: Projektowanie specjalnych filtrów cyfrowych metodą okien
% Parametry

typ = ["Hilberta", "różniczkujacy", "interpolujacy"];

for i=1:3
    M = 20; N = 2*M+1; n=1:M;
    % Generowanie polowy odpowiedzi impulsowej
    if(i==1)
        h = 2/pi*sin(pi*n/2).^2 ./ n; % Odp impulsowa filtra Hilberta
    end
    if i==2
        h = cos(pi*n)./n; % Odp impulsowa filtra rozn.
    end
    if i==3
        K = 5; wc=pi/K; fc=wc/(2*pi);
        h = 2*fc*sin(wc*n)./(wc*n); % ODp impuls filtra interp.
    end
    if (i==1 || i==2)
        h = [-h(M:-1:1) 0 h(1:M)]; % Odbicie odpowiedzi
    else
        h = K*[-h(M:-1:1) 2*fc h(1:M)]; % Odbicie i skalowanie przez K
    end

    % Mnożenie odp z oknem
    w = blackman(N)';
    hw = h.*w;

    % Widmo Fouriera

```

```

m = -M:1:M;
NF = 500; fn=0.5*(1:NF-1)/NF;
for k=1:NF-1
    H(k)=sum( h .* exp(-j*2*pi*fn(k)*m) );
    HW(k)=sum( hw .* exp(-j*2*pi*fn(k)*m) );
end

% wykresy
figure(72+2*i-1);
sgtitle(["Odpowiedz impulsowa filtra: " + typ(i)]);
subplot(211); stem(m,h); grid; title('h(n)');
subplot(212); stem(m,HW); grid; title('hw(n) - wymnozenie z oknem');
xlabel('n');

figure(72+2*i);
% Zastosowanie filtrów 1 i 2
if(i<3)
    % Sygnał testowy
    Nx=200; fx=50; fpr=1000; n=0:Nx-1; x=cos(2*pi*fx/fpr*n);
    y = conv(x, hw); % filtracja sygnału odpowiedzia
    % odcięcie stanów przejściowych (po N?1 próbek) z przodu i z tyłu
    % sygnału yz(n)
    yp = y(N:Nx);
    % odcięcie tych próbek w xz(n), dla których nie ma poprawnych
    % odpowiedników w yz(n)
    xp = x(M+1:Nx-M);
    if (i==1)
        z = xp + li*yp; % sygnał analityczny
        Ny = ceil(fpr/fx); k=1:Ny;
        subplot(311); plot(k, xp(k), 'b', k, yp(k), 'r');
        title('Sygnał analityczny'); legend('real(z)', 'imag(z)');
        subplot(312); plot(xp, yp); title('Imag(Real(z))'); grid;
        subplot(313); plot(abs(fft(z)));
        title('Widmo sygnału analitycznego'); grid;
    else
        % Filtrowanie różniczkujacy
        Ny = ceil(fpr/fx); k=1:Ny;
        plot(k, xp(k), 'b', k, yp(k), 'ro');
        yp
        title('Sygnał filtrowany filtrem różniczkujacym');
    end
end
if (i==3)
    % generacja sygnału testowego x(n)
    Nx=50; fx=50; fpr=1000; n=0:Nx-1; x=cos(2*pi*fx/fpr*n);
    xz=[]; KNx=K*Nx; xz=zeros(1,KNx);
    xz(1:K:KNx)=x(1:Nx); % dodanie zer
    % filtracja xz(n) za pomocą odp. impulsowej hw(n); otrzymujemy
    % Nx+N?1 próbek
    yz=conv(xz, hw);
    % odcięcie stanów przejściowych (po N?1 próbek) z przodu i z tyłu
    % sygnału yz(n)
    yp=yz(N:KNx);
    % odcięcie tych próbek w xz(n), dla których nie ma poprawnych
    % odpowiedników w yz(n)
    xp=xz(M+1:KNx-M);
    Ny=length(yp); k=1:Ny;
    plot(k,xp(k), 'or', k, yp(k), '-b');
    title('Sygnał filtrowany filtrem interpolujacym');
    grid; % porównanie
end

```

```

    end
end

```

Tab. 8.11. Kod programów na podstawie rozdziału 14.

```

% Rodzial 13 Zielinski
%                               Mateusz Krupnik

% Ćwiczenie: Filtry adaptacyjne typu LMS (NLMS) losowego gradientu
% Oznaczenia: x - syg filtrowany, d - sygnal odniesienia
%              y - przefiltrowany adaptacyjnie sygnal x,
%              e = d - y - sygnal bledu adaptacji
clear all; close all; clc;
% Wybor okna
okno_ = 2; % 1 - brak, 2-gaussa, 3-alfa*t, 4-exp(-alfa*t)
alg = 2;      %1 - LMS, 2 - NLMS
algo = ["LMS", "NLMS"];
% Parametry filtrów
% LMS
M = 50; mi = 0.1;    % mi <1 i >0
% NLMS
eng = 0.0; beta = 1 - 1/M;   % energia poczatkowa, wspolczynnik pamieci
gamma = 0.001;           % stala bezpieczenstwa mianownika!

% GEneracja sygnalu testowego
Nx = 1000; % probki
fpr = 1000; % Hz
A = 1;       % Ampl
f0 = 0;       % czestotliwosc poczatkowa sygnalu
df = 25;      % przyrost czest. na sek
dt = 1/fpr; t=0:dt:(Nx-1)*dt;
% Sygnał
s = A*cos(2*pi*(f0*t + df/2*t.^2));
% okna obwiedniowe
if (okno_==2) alfa=10; w=exp(-alfa*pi*(t-0.5).^2);end
if (okno_==3) alfa=5; w=alfa*t; end
if (okno_==4) alfa=5; w=exp(-alfa*t); end
if (okno_~=1) s = s.* w; end
okno = ["Brak", "Gaussa", "Liniowe", "Wykladnicze"];

for i=1:2
    if i==1
        % KAsowanie interferencji sieci - sinusoida 50Hz przesunieta w
        % fazie
        P = 0; % brak predykcji
        x = 0.1*sin(2*pi*50*t-pi/5); % sieć przesunięta w fazie
        d = s + 0.5*sin(2*pi*50*t); % sygnał + sieć
    else
        % Odszumianie sygnalu z szumu normlnaego
        P = 1; % rząd predykcji (do zmiany: 1,2,3,...)
        x = s + 0.25*randn(1,Nx); % sygnał + szum
        % odniesieniem sygnał "przyspieszony" o P próbek
        d = [ x(1+P:length(x)) zeros(1, P) ];
    end
    tempp = " przykład: " + num2str(i) + ...
            " Okno: " + okno(okno_) + " Alg: " + algo(alg);
    figure();
    sgtitle("Sygnały wejściowe" + tempp);

```

```

subplot(211); plot(t, x); grid;
title('Syg. wej.: x(t)'); xlabel('Czas [s]');
ylabel('Ampl.');
subplot(212); plot(t, d); grid;
title('Syg. wej.: d(t)=s(t)+x(t)'); xlabel('Czas [s]');
ylabel('Ampl.');

% Filtracja adaptacyjna, 4 wiersze dla 4 sygnałów
bx = zeros(1, M); % bufor wejściowy x
h = zeros(1, M); % Wagi filtra LMS
y = []; e = []; % wektory wyniku
for k=1:length(x)
    bx = [x(k) bx(1:M-1)];
    dest = h * bx';

    err = d(k) - dest;
    if (alg==1)
        h = h + (2*mi*err*bx);
    else
        eng = bx*bx';
        h = h + ((2*mi)/(gamma + eng)) * err * bx;
    end
    y = [y dest];
    e = [e err];
end
% Wykresy wyników
figure();
sgtitle("Sygnały wyj.," + tempp);
subplot(211); plot(t, y); grid;
title('Syg. wyj.: y(t)'); xlabel('Czas [s]'); ylabel('Ampl.');
subplot(212); plot(t, e); grid;
title('Syg. wyj.: e(t)=d-y'); xlabel('Czas [s]'); ylabel('Ampl.');

figure()
if (i==1)
    subplot(111); plot(t,s,'r',t,e,'b'); grid; xlabel('czas [sek]');
end
if (i==2)
    n=1:Nx-P;
    subplot(111); plot(t(n),s(n),'r',t(n),y(n),'b');
    grid; xlabel('czas [sek]');
end
title("Orginal (czerwony) i wynik filtracji (niebieski)" + tempp);
end

```

Tab. 8.12. Kod programów na podstawie rozdziału 17.

```

% Rozdział 17 Zielinski
% Mateusz Krupnik

% Ćwiczenie: Czasowo-częstotliwościowa reprezentacja Gabora
clc; clear all; close all;
% Parametry wejściowe
nw = 64; % Długość okna
L = 128; % Długość okna po uzupełnieniu zerami
dM = 4; % Przesunięcie w czasie
dN = 4; % Przesunięcie w częstotliwości
% dMdN < L !

```

```
% Okno czasowe do analizy (musi tłumić sygnał jak np. Gaussa)
w = blackman(nw)'; % Okno Blackmana
% Uzupełnienie zerami po bokach do długości L
w = [zeros(1, (L-nw)/2) w zeros(1, (L-nw)/2) ];

% Generacja odwrotnego okna analizy
M = L/dM; N = L/dN; ww = [ w w ];
H = []; k = 0 : L-1;
for p=0:dM-1
    for q=0:dN-1
        h = ww(k+q*N+1) .* exp(-1i*2*pi*p*k/dM); % Nowa funkcja bazy
        H = [H; h]; % Dodanie nowej funkcji bazy do bazy
    end
end

mi = zeros(dM*dN, 1); mi(1,1)=dM/N;
dw = pinv(H)*mi; dw=dw';

figure(1)
sgtitle(['Czasowo-częstotliwościowa reprezentacja Gabora']);
subplot(211); plot(real(w)); title('Okno czasowe Blackmana z zerami');
grid;
subplot(212); plot(real(dw)); title('Okno dualne do powyższego');
grid;

% Generacja sygnału zmiennego w częstotliwości z czasem
% Parametry
fpr = 128; f0 = 0; df = 24; dt = 1/fpr; n = 0:L-1; t = n*dt;
x = sin(2*pi*(f0*t+0.5*df*t.^2));

% Analiza
% Punkty czasu L/dM, punkty częst. L/dN
dwz = [dw zeros(1,2*L)];
for k=0:2*L/dM-1
    okno = dwz(L+1:L+L); % Wyciecie okna
    widmo = fft(x.*okno); % FFT
    tf(k+1, :) = widmo(1:dN:L); % Decymacja co krok częst.
    dwz = [zeros(1,dM) dwz(1:3*L-dM)]; % przesunięcie okna o krok czas.
end
figure(2)
sgtitle(['Czasowo-częstotliwościowa reprezentacja Gabora']);
subplot(2,2,[1, 2]); plot(t, x); title('Sygnał wejściowy');
xlabel('Czas [s]');
subplot(2,2,3); mesh(abs(tf)); title('Wykres 3D'); grid;
subplot(2,2,4); contour(abs(tf)); title('Kontur'); grid;

% Synteza sygnału
wz = [w zeros(1,2*L)]; % Ono syntezy uzupełnione zerami
y = zeros(1, L);
temp = (2*pi/L)*dN;
n = 0:1:L-1;
for m=0:1:2*L/dM-1
    ww = wz(L+1:L+L); % Okno z zerami
    for k=0:1:L/dN-1
        y = y + tf(m+1, k+1)*(ww .* (cos(temp*k*n) + li*sin(temp*k*n)));
    end
    wz = [zeros(1, dM) wz(1:3*L-dM)]; % Przesunięcie okna o krok czas.
end
y = real(y);
```

```

figure(3); sgtitle(['Czasowo-częstotliwościowa reprezentacja Gabora']);
subplot(311); plot(n, y); title('Sygnał wyjściowy z syntezy'); grid;
subplot(312); plot(n, x, n, y); title('Porównanie z wejściowym'); grid;
legend('WEJ x(t)', 'WYJ y(t)');
subplot(313); plot(n, y-x); title('Błąd y-x'); grid;

blad_max = max(abs(y-x))
blad_std = std(x-y)

%% Ćwiczenie: Krótkoczasowa transformacja Fouriera
% Parametry wejściowe
M=32; % połowa długości okna (całe okno N=2M-1)
Nx=128; % długość sygnału testowego
% Sygnał testowy z modulacją częstotliwości typu LFM i SFM
fpr=128; f0=0; df=32; fn=16; fm=3; dfm=12; dt=1/fpr; n=0:Nx-1; t=n*dt;
x1 = sin(2*pi*(f0*t+0.5*df*t.^2));
x2 = sin( 2*pi* (fn*t + (dfm/(2*pi*fm))*sin(2*pi*fm*t)) );
% Analiza TF ? krótkoczasowa reprezentacja Fouriera
x1 = hilbert(x1); x2 = hilbert(x2); % Sygnał analityczny
w = hanning(2*M-1)'; % Okno Hanninga
for n = M : Nx-M+1
    xx1 = x1(n-(M-1): 1 :n+(M-1)); xx1 = xx1 .* w; xx1 = [ xx1 0 ];
    X1(:,n-M+1) = fftshift(abs(fft(xx1))');
    xx2 = x2(n-(M-1): 1 :n+(M-1)); xx2 = xx2 .* w; xx2 = [ xx2 0 ];
    X2(:,n-M+1) = fftshift(abs(fft(xx2))');
end
% Rysunek widma TF
t_=t(M:Nx-M+1); f=fpr/(2*M)*(-M:M-1);
figure(4); sgtitle('Krótkoczasowa transformacja Fouriera sygnału LFM');
subplot(2,2, [1,2]); plot(t, x1); title('Sygnał LFM'); grid;
subplot(223); mesh(t_,f,X1); view(-40,70); axis tight;
title('Wykres analizy w 3D');
xlabel('Czas [s]'); ylabel('Częstotliwość [Hz]');
subplot(224); imagesc(t_,f,X1); title('Rzut z góry');
xlabel('Czas [s]'); ylabel('Częstotliwość [Hz]');

figure(5); sgtitle('Krótkoczasowa transformacja Fouriera sygnału SFM');
subplot(2,2, [1,2]); plot(t, x2); title('Sygnał SFM'); grid;
subplot(223); mesh(t_,f,X2); view(-40,70); axis tight;
title('Wykres analizy w 3D');
xlabel('Czas [s]'); ylabel('Częstotliwość [Hz]');
subplot(224); imagesc(t_,f,X2); title('Rzut z góry');
xlabel('Czas [s]'); ylabel('Częstotliwość [Hz]');

%% Ćwiczenie: Generacja funkcji skalujących i falek.
clear all;
niter = 10; % liczba iteracji
c = 0; d = 1; % {c=1, d=0} ? funkcja skalująca, {c=0, d=1} ? falka
% definicja współczynników filtrów h0 i h1 systemu falkowego Db4 (17.62)
% (17.57)
h0 = [ (1+sqrt(3))/(4*sqrt(2)) (3+sqrt(3))/(4*sqrt(2)) ...
        (3-sqrt(3))/(4*sqrt(2)) (1-sqrt(3))/(4*sqrt(2)) ];
N = length(h0); n = 0:N-1;
h1 = (-1).^n .* h0(N:-1:1);
% synteza ? według schematu drzewa filtrów z rysunku 17.15
c = [ 0 c 0 ]; % aproksymacje
d = [ 0 d 0 ]; % detale
c = conv(c,h0) + conv(d,h1);
for n = 1 : niter

```

```

for k = 1:length(c)
    c0(2*k-1) = c(k);
    c0(2*k) = 0;
end
c0 = [ 0 c0 ];
c = conv(c0, h0);
end
figure(6);
plot(c); title('Przykładowa falka');

%% Ćwiczenie: Transformacja falkowa
clear all;
% Parametry programu
niter = 3; % liczba iteracji
nx = 2^niter*32; % długość sygnału
% Definicja współczynników filtra LP syntezy h0s, np. Db4
h0s = [ (1+sqrt(3))/(4*sqrt(2)) (3+sqrt(3))/(4*sqrt(2)) ...
         (3-sqrt(3))/(4*sqrt(2)) (1-sqrt(3))/(4*sqrt(2)) ];
% Oblicz pozostałe filtry
N = length(h0s); n = 0:N-1;
h1s = (-1).^n .* h0s(N:-1:1); % filtr HP syntezy
h0a = h0s(N:-1:1); h1a=h1s(N:-1:1); % filtry LP i HP analizy
% Sygnał testowy
x1 = sin(2*pi*(1:nx)/32);
x2 = rand(1,nx);
% Analiza
cc1 = x1;
cc2 = x2;
for m=1:niter
    c01 = conv(cc1,h0a); % filtracja LP x1
    d01 = conv(cc1,h1a); % filtracja HP x1

    k1=N:2:length(d01)-(N-1); kp1=1:length(k1);
    ord1(m)=length(kp1); dd1(m,kp1) = d01( k1 );
    k1=N:2:length(c01)-(N-1); cc1=c01( k1 );

    c02 = conv(cc2,h0a); % filtracja LP x2
    d02 = conv(cc2,h1a); % filtracja HP x2

    k2=N:2:length(d02)-(N-1); kp2=1:length(k2);
    ord2(m)=length(kp2); dd2(m,kp2) = d02( k2 );
    k2=N:2:length(c02)-(N-1); cc2=c02( k2 );
end
% Synteza sygnałów
c1=cc1; c2=cc2;
for m=niter:-1:1
    c01=[]; d01=[];
    for k = 1:length(c1)
        c01(2*k-1)=c1(k); c01(2*k)=0;
    end
    c1 = conv(c01,h0s); nc1=length(c1);
    for k = 1:ord1(m)
        d01(2*k-1) = dd1(m,k); d01(2*k) = 0;
    end
    d1 = conv(d01,h1s); nd1=length(d1);
    c1 = c1(1:nd1);
    c1 = c1 + d1;

    c02=[]; d02=[];
    for k = 1:length(c2)

```

```

c02(2*k-1)=c2(k); c02(2*k)=0;
end
c2 = conv(c02,h0s); nc2=length(c2);
for k = 1:ord2(m)
    d02(2*k-1) = dd2(m,k); d02(2*k) = 0;
end
d2 = conv(d02,h1s); nd2=length(d2);
c2 = c2(1:nd2);
c2 = c2 + d2;
end

% Wykresy końcowe
n1 = 2*(N-1)*niter : length(c1)-2*(N-1)*niter+1;
figure(7); sgtitle('Analiza sygnału sinusoidalnego');
subplot(311);
plot(x1); grid; title('Sygnał wejściowy');
subplot(312);
plot(n1,c1(n1)); title('Sygnał wyjściowy'); grid;
subplot(313); grid; plot(n1, x1(n1)-c1(n1));
title('Błąd analizy');

n2 = 2*(N-1)*niter : length(c2)-2*(N-1)*niter+1;
figure(8); sgtitle('Analiza sygnału losowego');
subplot(311);
plot(x2); grid; title('Sygnał wejściowy');
subplot(312);
plot(n2,c2(n2)); title('Sygnał wyjściowy'); grid;
subplot(313); grid; plot(n2, x2(n2)-c2(n2));
title('Błąd analizy');

```

Tab. 8.13. Kod programów na podstawie rozdziału 22.

```

% Rozdział 17 Zielinski
%                               Mateusz Krupnik
% Ćwiczenie: Wykorzystanie transformacji 2D DFT i 2D DCT do filtracji
% obrazu
clear all; close all; clc;
% Inicjalizacja ? wczytaj obraz
[x,cmap] = imread('cameraman.tif');
% wczytaj obraz do "x" i jego paletę kolorów do "cmap"
imshow(x,cmap), title('Obraz'); % pokaż obraz wykorzystując jego paletę
[M, N] = size(x); % odczytaj liczbę wierszy i kolumn; założenie M=N !!!
x = im2double(x); % zamień reprezentację pikseli

% Macierz transformacji 1D DCT oraz 1D DFT
n=0:N-1; % numery próbek funkcji bazowych
c = [sqrt(1/N) sqrt(2/N)*ones(1,N-1)];
f = 1/sqrt(N); % współczynniki normalizujące
for k=0:N-1           % wyznacz macierz transformacji
    C(k+1,n+1) = c(k+1) * cos(pi*k*(n+1/2) / N); % funkcje bazowe 1D DCT
    F(k+1,n+1) = f * exp(j*2*pi/N*k*n); % funkcje bazowe 1D DFT
end

% JEDNA LINIA - transformacja DFT i DCT wiersza
Nr = 100; K = 2; y = x; % numer linii, szerokość znacznika, kopia obrazu
linia = x(Nr,1:N);      % pobranie linii

```

```

y(Nr-K:Nr+K,1:N) = 0*ones(2*K+1,N); % zaznacz wybraną linię na czarno
% pokaż obraz z czarną linią
figure(2)
subplot(221); imshow(y,cmap); title('Obraz');
% pokaż wykres linii obrazu
subplot(222); plot(linia); title('Jedna linia');
% DFT linii obrazu
subplot(223); plot( abs(fft(linia))/N ); title('|DFT|');
grid;
% DCT (Matlab) linii
% subplot(212); plot( dct(linia)/sqrt(N) ); title('DCT');
% DCT (nasze) linii
subplot(224); plot( (conj(C)*linia')/sqrt(N) ); title('DCT');
grid;

% FILTRACJA za pomocą 2D DCT
% maska częstotliwościowa, lewy górny róg
K = 64; H = zeros(M,N); H(1:K,1:K) = ones(K,K);
% X = dct2(x); % 2D DCT ? MATLABA
X = conj(C) * x * conj(C).'; % 2D DCT ? NASZE (dla DCT conj(C)=C)
Y = X .* H; % iloczyn widma DCT i maski
% y = idct2(Y); % 2D IDCT ? MATLABA
y = C.' * Y * C; % 2D IDCT ? NASZE
XdB = skaladB( X );
YdB = skaladB( Y ); % wyskalowanie intensywności pikseli w dB
figure(3)
subplot(221); imshow(x, cmap); title('Obraz'); % dalej tylko wizualizacja
subplot(222); imshow(XdB, cmap); title('Widmo DCT');
subplot(223); imshow(YdB, cmap); title('Widmo DCT + Maska');
subplot(224); imshow(y(1:128,65:192), cmap); title('Fragment wyniku');

% FILTRACJA 2D za pomocą 2D DFT (fftshift2D - przestawianie ćwiartek widma
% 2D DFT, patrz rys. 22.11a) maska częstotliwościowa H (MxN)
K = 32; H = zeros(M,N);
% środek = (M/2+1, N/2+1)
H(M/2+1-K : M/2+1+K, N/2+1-K : N/2+1+K) = ones(2*K+1,2*K+1);
h = fftshift2D(real(ifft2(fftshift2D(H)))); % odpowiedź impulsowa maski
figure(4)
subplot(121);
imshow(255*H,cmap); title('Maska Freq'); % rysunek maski
subplot(122);
mesh(h); title('Odpowiedź impulsowa'); % rysunek jej odp. impulsowej

% X = fft2(x)/N; % transformacja Fouriera 2D DFT ? MATLABA
X = conj(F) * x * conj(F).'; % transformacja Fouriera 2D DFT ? NASZA
Xp = fftshift2D(X); % przestawienie miejscami ćwiartek widma
Yp = Xp .* H; % filtracja = iloczyn widma 2D DFT i maski 2D
Y = fftshift2D(Yp); % powrotne przestawienie ćwiartek widma
% y1 = ifft2(Y)*N; % odwrotna transformacja Fouriera 2D IDFT MATLABA
y1 = F.' * Y * F; % odwrotna transformacja Fouriera 2D IDFT NASZA
y1 = real(y1); % część rzeczywista, urojona równa零
y1f = y1(1:128,65:192); % wybranie fragmentu obrazu do wizualizacji

XdB = skaladB( X ); XpdB = skaladB( Xp );
YdB = skaladB( Y ); YpdB = skaladB( Yp );
figure(5)
subplot(231); imshow(x, cmap); title('1. Obraz');
subplot(232); imshow(XdB, cmap); title('2. 2D DFT');
subplot(233); imshow(XpdB, cmap); title('3. Po przestawieniu');

```

```

subplot(234); imshow(YpdB, cmap); title('4. Po filtrze');
subplot(235); imshow(YdB, cmap); title('5. Po przestawieniu');
subplot(236); imshow(ylf, cmap); title('6. Fragment wyniku');

% FILTRACJA 2D za pomocą splotu 2D
L = 32;
y2 = conv2(x, h(M/2+1-L:M/2+1+L, N/2+1-L:N/2+1+L), 'same');
figure(6)
subplot(121); imshow(y1, cmap);
title('Obraz po filtrze FREQ - splot cykliczny');
subplot(122); imshow(y2, cmap);
title('Obraz po filtrze CONV - splot liniowy');

%% Ćwiczenie: Projektowanie filtrów 2D
clear all;
L = 15; % szerokość macierzy wag filtra (nieparzysta: 3, 5, 7, 9, ... )
K = (L-1)/2; df = 0.5/K; % zmienne pomocnicze do generacji wag

m = ones(L,1)*(-K:K); % i opisu osi rysunków
n = (-K:K)'*ones(1,L);
fm = ones(L,1)*(-0.5:df:0.5);
fn = (-0.5:df:0.5)'*ones(1,L);

% Wczytaj obraz do filtracji
[x,cmap] = imread('cameraman.tif');
imshow(x,cmap), title('Obraz');
[N, M] = size(x);
x = im2double(x);

% FILTRY POCHODZĄCE OD FUNKCJI GAUSSA
sigma = 1.4; df = 0.5/K;
g0 = 1/(2*pi*sigma^2) .* exp(-(m.^2+n.^2)/(2*sigma^2)); % funkcja Gaussa
g1m = -m/(sigma^2) .* g0; % pochodna względem osi m
g1n = -n/(sigma^2) .* g0; % pochodna względem osi n
g2 = (m.^2 + n.^2 - 2*sigma^2)/(sigma^4) .* g0; % laplasjan funkcji Gaussa
figure(7); sgtitle('Filtr pochodzace od f. Gaussa');
subplot(221); mesh(m,n,g0); title('Filtr Gaussa');
subplot(222); mesh(m,n,g2); title('Laplasjan f. Gaussa');
colormap([0 0 0]);
subplot(223); imshow(conv2(x,g0,'same'),cmap );
subplot(224); imshow(conv2(x,g2,'same'),cmap );
figure(8); sgtitle('a');
subplot(231); mesh(m,n,g1m); title('Gradient "m"');
subplot(232); mesh(m,n,g1n); title('Gradient "n"');
colormap([0 0 0]);
subplot(234); imshow(conv2(x,g1m,'same'),cmap );
subplot(235); imshow(conv2(x,g1n,'same'),cmap );
subplot(236);
imshow(sqrt(conv2(x,g1m,'same')).^2+conv2(x,g1n,'same')).^2, cmap);

%% METODA OKIEN
chka = 0; % 0 = charakterystyka prostokątna, 1 = kołowa
w = hamming(L); w = w * w'; % okno 2D
figure(9);
subplot(111); mesh(m,n,w); colormap([0 0 0]); title('2D Okno');
for chka=0:1
    if(chka==0)
        % Charakterystyka prostokątna - odp. impulsowa dwóch filtrów

```

```

% LowPass
f0=0.25; wc=pi*f0;
sinc=sin(wc*(-K:K))./(pi.*(-K:K));
sinc(K+1)=f0; lp1=sinc'*sinc;

f0=0.50; wc=pi*f0;
sinc=sin(wc*(-K:K))./(pi.*(-K:K));
sinc(K+1)=f0; lp2=sinc'*sinc;
chkat = "prostokatna"
else
    % Charakterystyka kołowa - odp. impulsowa dwóch filtrów LowPass
    f0=0.25; wc=pi*f0;
    lp1=wc*besselj(1,wc*sqrt(m.^2 + n.^2))./(2*pi*sqrt(m.^2+n.^2));
    lp1(K+1,K+1)=wc^2/(4*pi);
    f0=0.50; wc=pi*f0;
    lp2=wc*besselj(1,wc*sqrt(m.^2+n.^2))./(2*pi*sqrt(m.^2+n.^2));
    lp2(K+1,K+1)=wc^2/(4*pi);
    chkat = "kołowa"
end
lp = lp1; % LowPass bez okna 2D
lpw = lp .* w; % z oknem
hp = - lp1; hp(K+1,K+1) = 1 - lp1(K+1,K+1); % HighPass bez okna 2D
hbw = hp .* w; % z oknem
bp = lp1 - lp2; % BandPass bez okna 2D
bpw = bp .* w; % z oknem
bs = - bp; bs(K+1,K+1) = 1 - bp(K+1,K+1); % BandStop bez okna 2D
bsw = bs .* w; % z oknem
for typ = 1 : 4
    % pokaż odp. impulsowa, jej widmo i przefiltrowany obraz
    switch (typ) % wybierz typ filtra
        case 1, h = lp; hw = lpw; filtr = "LowPass"; % LP
        case 2, h = hp; hw = hbw; filtr = "HighPass"; % HP
        case 3, h = bp; hw = bpw; filtr = "BandPass"; % BP
        case 4, h = bs; hw = bsw; filtr = "BandStop"; % BS
    end
    figure(9+(chka*4)+typ)
    sgttitle("Filtracja: " + chkat + ", " + filtr);
    subplot(321);
    mesh(m,n,h); title('Filtr h(m,n)');
    subplot(322);
    mesh(m,n,hw); title('Filtr hw(m,n)');
    subplot(323);
    mesh(fm,fn,abs(fftshift2D(fft2(h))) ); title('|H(fm,fn)|');
    subplot(324);
    mesh(fm,fn,abs(fftshift2D(fft2(hw))) ); title('|Hw(fm,fn)|');
    colormap([0 0 0]);

    subplot(325); y = conv2(x, h, 'same');
    imshow(y,[min(min(y)),max(max(y))]);
    subplot(326); y = conv2(x, hw, 'same');
    imshow(y,[min(min(y)),max(max(y))]);
end
end

%% FILTRY PROJEKTOWANE W DZIEDZINIE CZĘSTOTLIWOŚCI
% Zmienne pomocnicze do generacji wag i opisu osi rysunków
N=L+1; df = 0.5/(K+1);
m = ones(L+1,1)*(-(K+1):K); n = (- (K+1):K)'*ones(1,L+1);
fm = ones(L+1,1)*(-0.5:df:0.5-df); fn = (-0.5:df:0.5-df)'*ones(1,L+1);
% Okno 2D - jego kształt i widmo

```

```
w = hamming(N); w = w * w';
% Zadana charakterystyka częstotliwościowa - kołowa lub prostokątna
Q = round(K/2); % szerokość filtra LP
H = zeros(N,N);
for k = N/2+1-Q : N/2+1+Q % kołowa
    for l = N/2+1-Q : N/2+1+Q
        if( (k-N/2-1)^2 + (l-N/2-1)^2 <= Q^2)
            H(k,l) = 1;
        else
            H(k,l) = 0;
        end
    end
end

% H(N/2+1-Q : N/2+1+Q, N/2+1-Q : N/2+1+Q) = ones(2*Q+1,2*Q+1); %
% prostokątna Zaprojektowanie filtra i sprawdzenie jego działania
h = real(ifft2(fftshift2D(H))); h = fftshift2D(h); % odpowiedź impulsowa
hw = h .* w; % odp. impulsowa z oknem
Hw = abs(fftshift2D(fft2(hw))); % jej widmo
y = conv2(x, hw, 'same'); % filtracja
% Rysunki
figure(18)
sgtitle('Okno filtracji i jego ch. częst.');
subplot(121);
mesh(m,n,w); title('Okno 2D w(m,n)');
subplot(122);
mesh(fm,fn,abs(fftshift2D(fft2(w)))); title('|W(fm,fn)|');
colormap([0 0 0]);
figure(19);
sgtitle('Filtr projektowany w dz. częst.');
subplot(221); mesh(fm,fn,H); title('Zadane H(fm,fn)');
subplot(222); mesh(m,n,h); title('Filtr 2D h(m,n)');
subplot(223); mesh(fm,fn,Hw); title('Ch-ka |Hw(fm,fn)|');
subplot(224); mesh(m,n,hw); title('Filtr 2D hw(m,n) z oknem');
colormap([0 0 0]);
figure(20);
sgtitle('Filtracja obrazu charakterystyka projektowana w d. częst.');
subplot(121); imshow(y, cmap);
title('Cały obraz po filtrze'); y = y(1:128,65:192);
subplot(122); imshow(y,[min(min(y)),max(max(y))]);
title('Tylko fragment');

%% FUNKCJE ZAGNIEZDZONE %%
function Y = fftshift2D(X)
    % przedstawianie ćwiartek widma 2D DFT
    [M N] = size(X);
    Y(M/2+1:M,N/2+1:N) = X(1:M/2,1:N/2);
    Y(1:M/2,1:N/2) = X(M/2+1:M,N/2+1:N);
    Y(M/2+1:M,1:N/2) = X(1:M/2,N/2+1:N);
    Y(1:M/2,N/2+1:N) = X(M/2+1:M,1:N/2);
end
function XdB = skaladB(X)
    % skalowanie intensywności pikseli obrazu w decybelach
    XdB = log10(abs(X)+1); maxXdB = max(max(XdB)); minXdB = min(min(XdB));
    XdB = (XdB-minXdB)/(maxXdB-minXdB)*255;
end
```