

**Wydział Inżynierii Mechanicznej i Robotyki**

**Informatyka w inżynierii mechanicznej**

Sprawozdanie z przedmiotu Cyfrowe Przetwarzanie Sygnałów

285608, Mateusz Krupnik

Spis treści

[Tabele z kodami 2](#_Toc43880809)

[1. Wstęp 3](#_Toc43880810)

[1. Laboratorium 1 3](#_Toc43880811)

[2. Laboratorium 2 4](#_Toc43880812)

[3. Laboratorium 3 6](#_Toc43880813)

[4. Laboratorium 4 9](#_Toc43880814)

[5. Laboratorium 5 19](#_Toc43880815)

[6. Laboratorium 6 34](#_Toc43880816)

[7. Laboratorium 7 54](#_Toc43880817)

# Tabele z kodami

[Tab. 1.1. Kod programu „Krupnik\_Mateusz\_Lab\_1.m”. 3](#_Toc43880823)

[Tab. 2.1. Kod programu „Krupnik\_Mateusz\_Lab\_2.m”. 4](#_Toc43880824)

[Tab. 2.2. Parametry sygnałów sinusoidalnych. 6](#_Toc43880825)

[Tab. 3.1. Kod programu „Krupnik\_Mateusz\_Lab\_3.m”. 6](#_Toc43880826)

[Tab. 4.1. Kod programu „Krupnik\_Mateusz\_Lab\_4”. 10](#_Toc43880827)

[Tab. 4.2. Wzory sygnałów używanych w programie. 16](#_Toc43880828)

[Tab. 5.1. Kod programu „Krupnik\_Mateusz\_Lab\_5\_1”. 20](#_Toc43880829)

[Tab. 5.2. Kod programu "Krupnik\_Mateusz\_Lab\_5\_2.m". 29](#_Toc43880830)

[Tab. 6.1. Kod programu „Laboratoria\_nr\_6\_1.m”. 35](#_Toc43880831)

[Tab. 6.2. Kod programu "Krupnik\_Mateusz\_Lab\_6\_2.m". 44](#_Toc43880832)

[Tab. 6.3. Kod programu "Krupnik\_Mateusz\_Lab\_6\_3.m". 46](#_Toc43880833)

[Tab. 6.4. Kod programu "Krupnik\_Mateusz\_Lab\_6\_4.m". 48](#_Toc43880834)

[Tab. 6.5. Kod programu "Krupnik\_Mateusz\_Lab\_6\_5.m". 50](#_Toc43880835)

[Tab. 7.1. Kod programu "Krupnik\_Mateusz\_Lab\_7.m". 54](#_Toc43880836)

# 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 płytkę 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.

# 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\_Mateusz\_Lab\_1.m”. Kolejne ćwiczenia posiadają analogiczną numerację.

Podstawowe polecenia to funkcje:

* fopen – funkcja otwierająca plik o podanej nazwie, z podanym trybem (zapis, odczyt),
* fprint – 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, wykreślane są jego wykresy i dokonywany jest zapis do pliku. Następnie następuje otwarcie ponowne pliku i odczyt danych i ponowne wykreślenie wykresu.

Tab. 1.1. Kod programu „Krupnik\_Mateusz\_Lab\_1.m”.

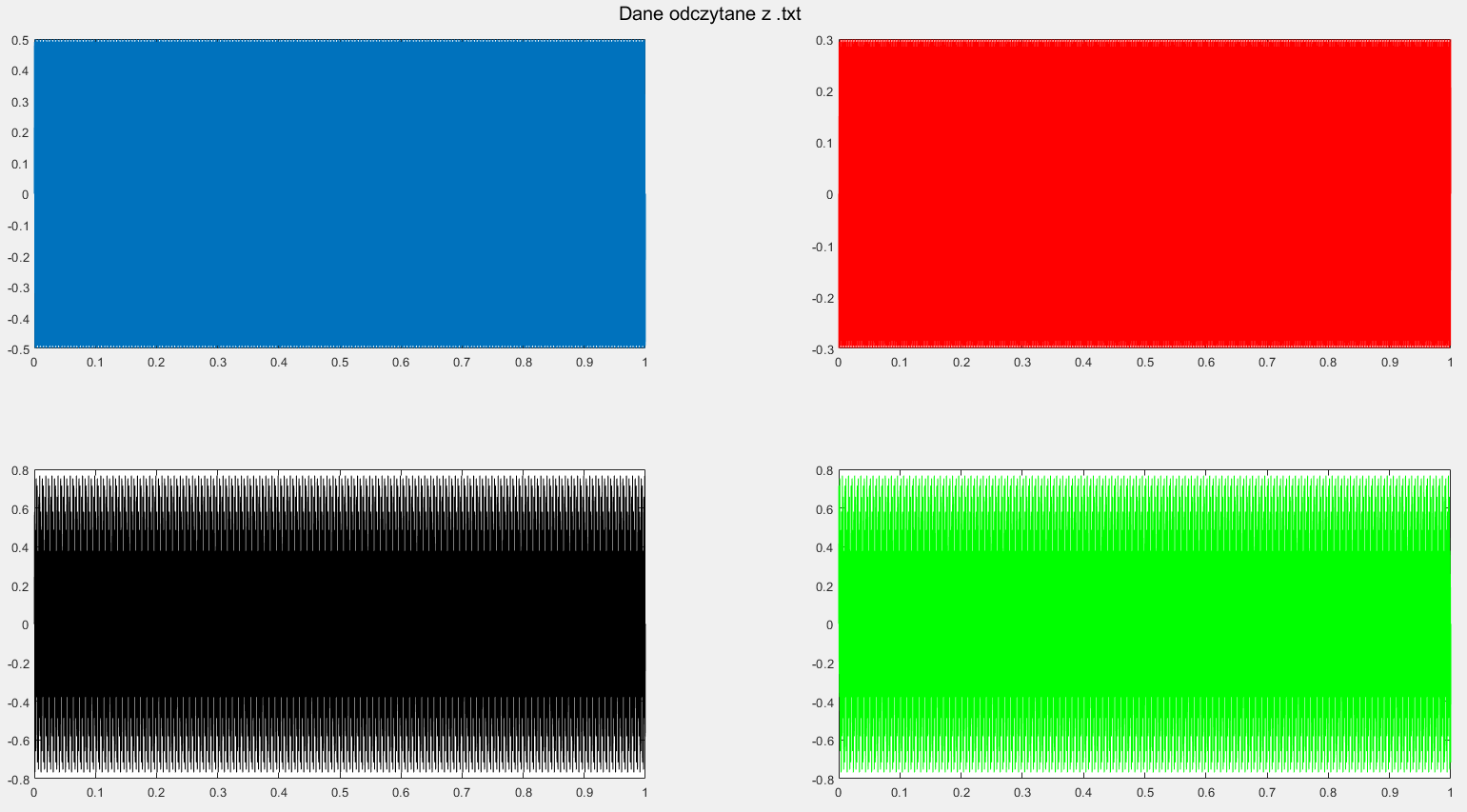
|  |
| --- |
| %% Lab 1 - operacje wejscia wyjscia - Mateusz Krupnik  % Generowanie przebiegu sinosuidalnego  clc; close all; clear all;  t=0:0.001:1;  A=0.7;  f=100;  omega=2\*pi\*f;  y=A\*sin(omega\*t);  % wykres  figure(1)  plot(t,y)  % Otwarcie pliku w trybie zapisywania  uchwyt=fopen('uchwyt.txt','w');  % Zapis kolumny czasu i przebiegu sinusoidalnego  fprintf(uchwyt,'%12.4f %12.4f\n',[t;y]);  % Zamknięcie wsztstkich 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,:)) |

# Laboratorium 2

W tym ćwiczeniu ponownie generowane są sygnały sinusoidalne, a funkcja sound powoduje ich reprezentacje 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\_Mateusz\_Lab\_2.m”.

|  |
| --- |
| %% Lab 2 - operacje wejscia wyjscia 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 wartosci o dł 12 znaków, 4  % znaki precyzji, \n - nowy wiersz  uchwyt=fopen('dane1.txt','w');  fprintf(uchwyt,'%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 dziesietnej lub wykladniczje, usuniecie zer  % z konca zapisu, %e - notacja wykladnicza, %f - dzisiętna  fclose('all');  uchwyt=fopen('dane1.txt','r');  D=fscanf(uchwyt,'%g %g %g %g %g \n',[5 inf]);  fclose('all');  % Zapis danych w postaci binarnej, a następnie ich odczyranie, inf - odczyt  % do ostatniej kolumny  uchwyt1=fopen('dane1.bin','w');  fwrite(uchwyt1,[t;y1;y2;y3;y4],'float');  fclose('all');  uchwyt1=fopen('dane1.bin','r');  y5=fread(uchwyt1,[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')    %% Wynzaczanie parametrów sygnałów za pomocą stworzonych funkcji  % Wyznaczenie wartości minimalnej i maksymalnej  % Wywoływane 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(y3) 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.

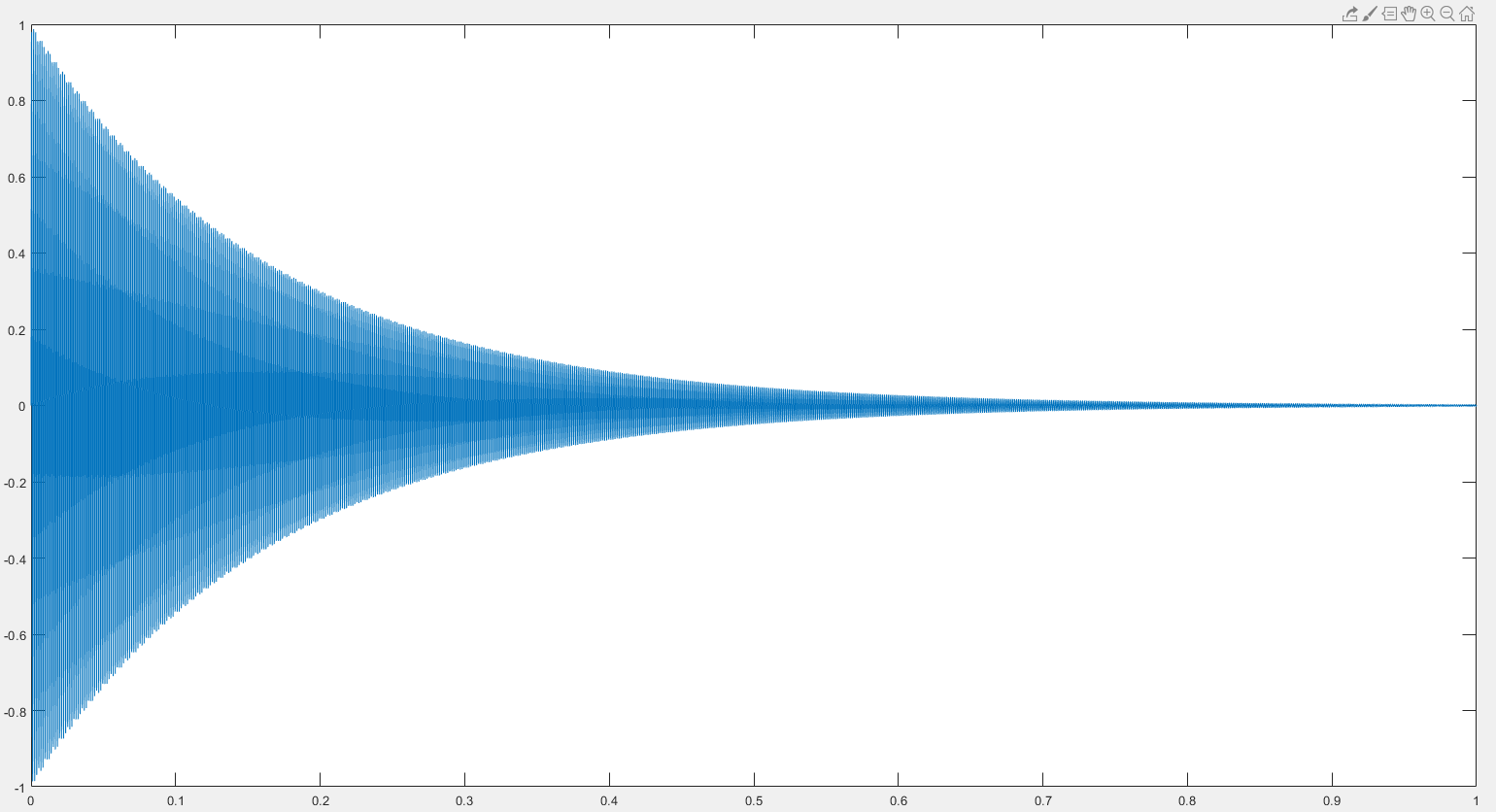
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sygnał  Parametr | Niebieski (1) | Czerwony (2) | Czarny (3) | Zielony (4) |
| Minimum | -0.5000 | -0.2994 | -0.7675 | -0.7675 |
| Maksimum | 0.5000 | 0.2994 | 0.7675 | 0.7675 |
| Średnia (x10-14) | -0.1417 | -0.1480 | -0.1480 | -0.1354 |
| Energia | 1250.0 | 450.00 | 1700.0 | 1700.0 |

# Laboratorium 3

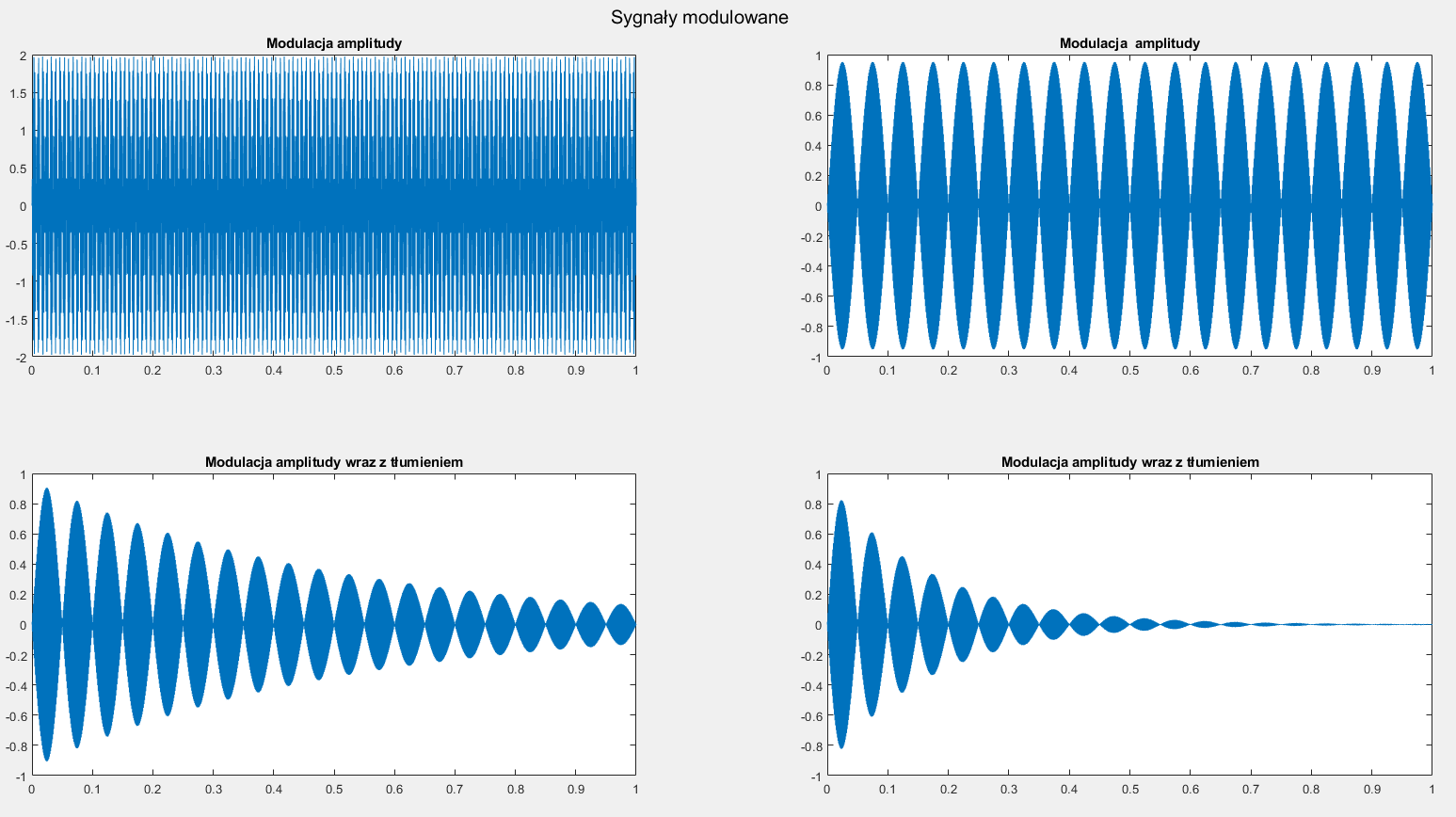
W ramach tych ćwiczeń realizowane zostały dodatkowo operacje na plikach dźwiękowych. Wykorzystywane funkcje to m. in. audiowrite służąca do zapisu sygnału w postaci pliku dźwiękowego np. o rozszerzeniu .wav. Ponownie funkcja sound pozwala na odtworzenie dźwięku. Generowane zostały sygnały sinusoidalne, 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\_Mateusz\_Lab\_3.m”.

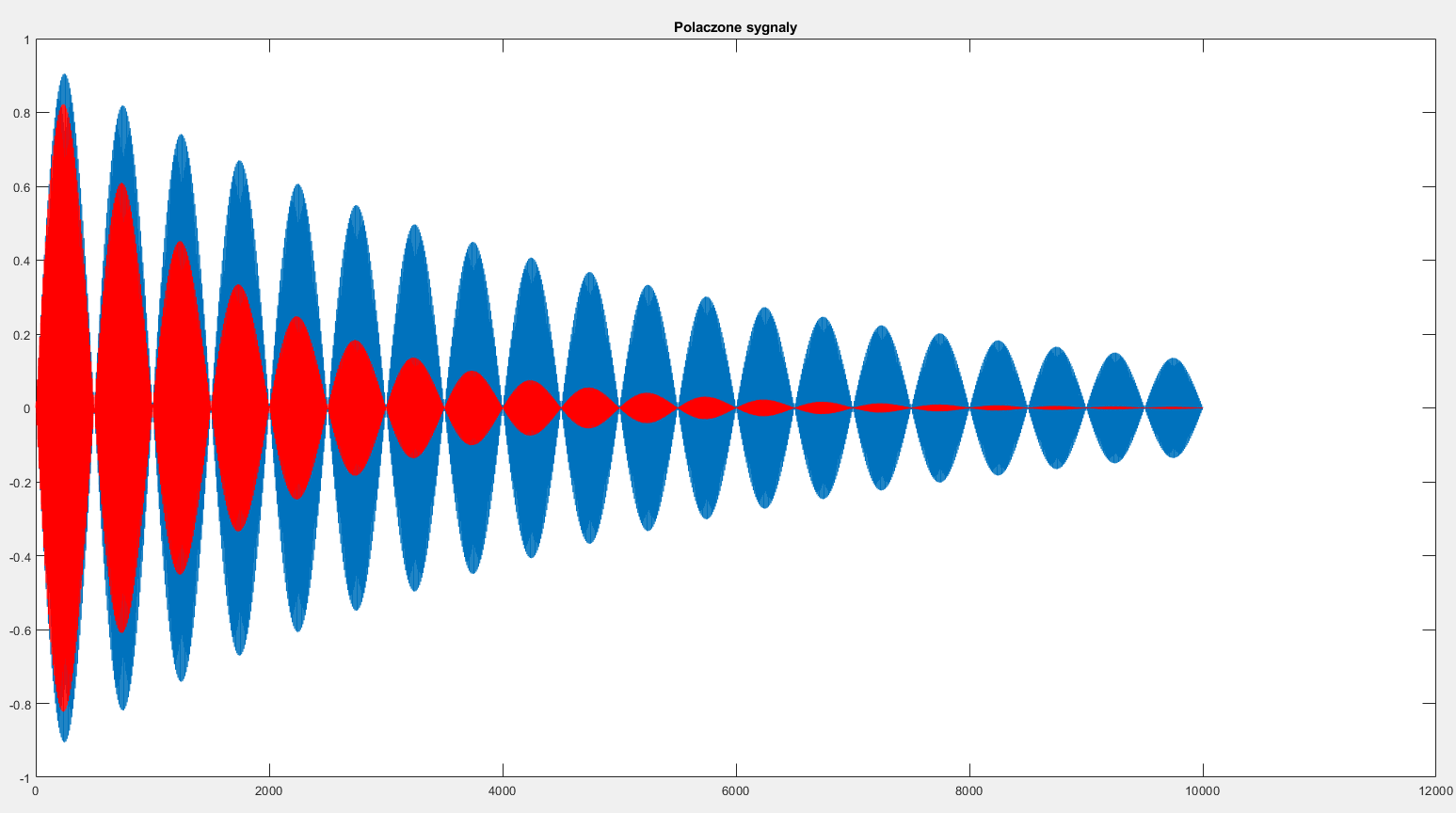
|  |
| --- |
| %% Lab 3 - praca z plikiem dziwiękowym - Mateusz Krupnik  clc; clear all; close all;  % Generowanie przebiegu, zapis i odczyt w postaci pliku wav  A=0.5;  B=-0.3;  f1=700;  fs=10000;  t=0:(1/fs):1;  y1=A\*sin(2\*pi\*f1\*t);  audiowrite('plik.wav', y1, fs);  clear all; % usuniecie danych  %% 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 dzwięku 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 sygnałow 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 dzwięku  [Y1, Fs] = audioread('plik2.wav');  figure(3)  plot(Y1(:, 1)); hold on; plot(Y1(:, 2), 'r');  hold off; title('Polaczone sygnaly');  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 dzwięku  [Y3, Fs] = audioread('plik3.wav');  figure(4)  plot(Y3(:, 1)); hold on; plot(Y3(:, 2), 'r.-');  hold off; title('Polaczone sygnaly');  sound(Y3, Fs); pause(t(end)); |



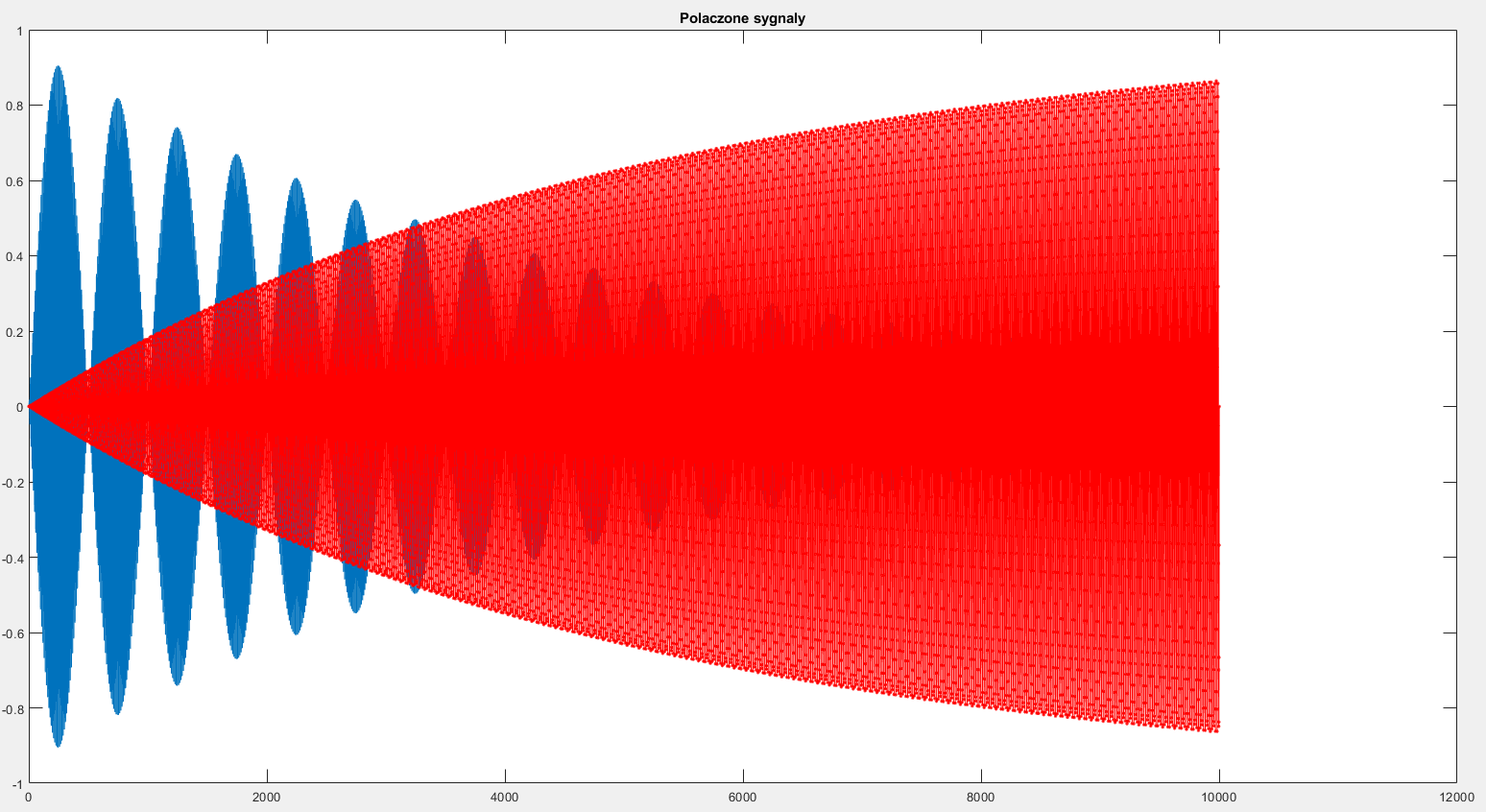
Rys. 3.1. Przykładowy przebieg sygnału sinusoidalnego z tłumieniem wykładniczym.



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.

# 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 ½ - 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:

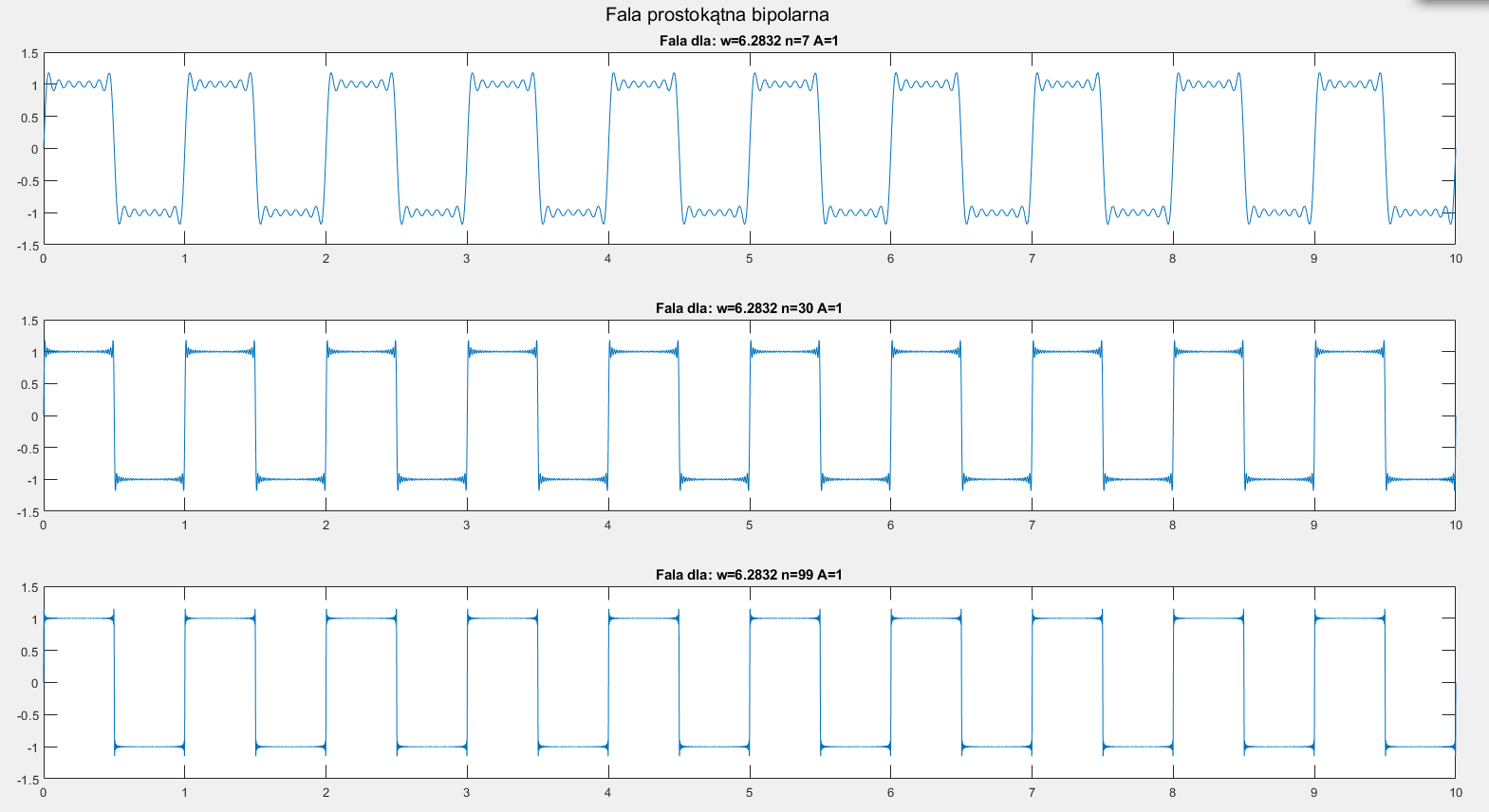
w – częstość [rad./s], A – amplituda [-], t - wektor czasu [s], n – rząd ciągu [-], f – częstotliwość [Hz], tau – wypełnienie [-] (od 0 do 1).

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.

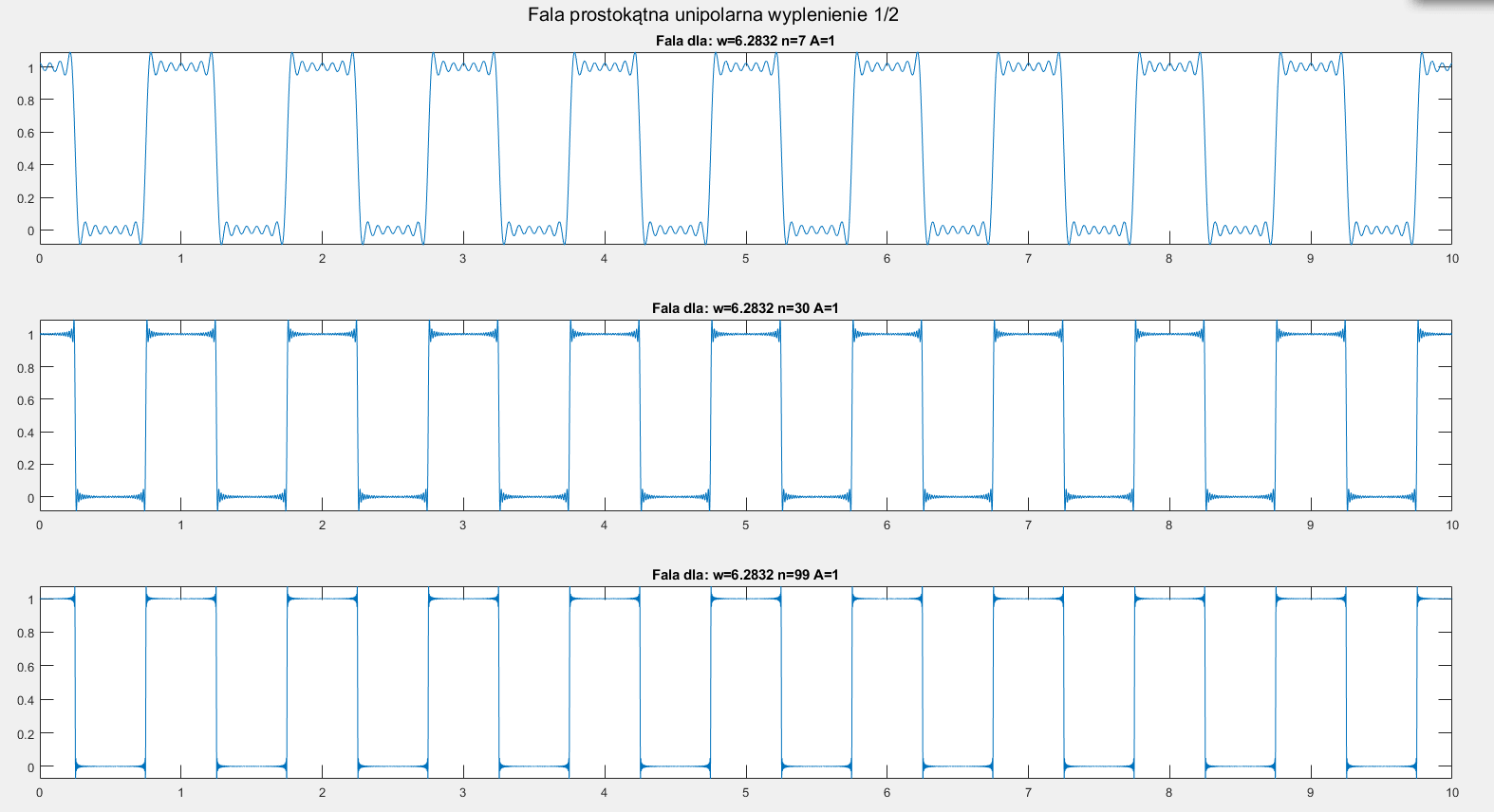
Tab. 4.1. Kod programu „Krupnik\_Mateusz\_Lab\_4”.

|  |
| --- |
| % Lab 4. Tworzenie generatorow sygnalow podstawowych  % Wywolanie nastepuje blokowo.  % Do działania wymagane są deklaracje funkcji:  % sbp, sup\_1\_2, swd, swj, tbp, tbpp, tup, tupp  % Deklaracje są zakomentowane na końcu pliku gdyby  % m pliki się zgubiły.  % 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 prostąktny unipolarny wypełnienie 1/2  % Generowanie wykresu  y = sup\_1\_2(w, A, t, n);    % Wykresy  figure(2)  sgtitle('Fala prostokątna unipolarna wyplenienie 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 prostąktny unipolarny o dowolonym wypelnieniu  tau = 0.2; % okres, wypelnienie  % Generowanie wykresu  y = sup\_wyp(f, A, t, n, tau);    % Wykresy  figure(3)  sgtitle(['Fala prostokątna unipolarna wyplenienie ' 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 trojkatna 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 piloksztaltny  % Generowanie wykresu  y = tbpp(w, A, t, n);    % Wykresy  figure(5)  sgtitle(['Fala trojkatna 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 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    %% Sygnal trojkatny unipolarna piloksztaltna  % Generowanie wykresu  y = tupp(w, A, t, n);    % Wykresy  figure(7)  sgtitle(['Fala trojkatna unipolarna piłokształtna']);  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 jednopołowkowy  % Generowanie wykresu  y = swj(w, A, t, n);    % Wykresy  figure(9)  sgtitle(['Fala sinusoidalna wyprostowana jednopołowkowa']);  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)  % Funcja generująca fale prostkatna bipolarna  % w - czestość, A - amplituda  % t - wektor czasu, n - rzad ciagu  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  x = x\*4\*A/pi;  end    % Definicja funkcji 2  function x = sup\_1\_2(w, A, t, n)  % Funcja generująca fale prostkatna unipolarna o wypelnieniu 1/2  % w - częstość, A - amplituda  % t - wektor czasu, n - rzad ciagu  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)  % Funcja generująca fale prostk. unipolarna o dowolnym wypelnieniu  % f - czestotliwosc, A - amplituda  % t - wektor czasu, n - rzad ciagu  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)  % Funcja generująca fale trojkatna bipolarna  % w - częstość, A - amplituda  % t - wektor czasu, n - rzad ciagu  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)  % Funcja generująca fale trojkatna bipolarna pilokształtna  % w - częstość, A - amplituda  % t - wektor czasu, n - rzad ciagu  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)  % Funcja generująca fale trojkatna unipolarna  % w - częstość, A - amplituda  % t - wektor czasu, n - rzad ciagu  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)  % Funcja generująca fale trojkatna unipolarna pilokształtna  % w - częstość, A - amplituda  % t - wektor czasu, n - rzad ciagu  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)  % Funcja generująca fale sinusoidalna wyprostowana dwupołowkową  % w - częstość, A - amplituda  % t - wektor czasu, n - rzad ciagu  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)  % Funcja generująca fale sinusoidalna wyprostowana jednopołowkową  % w - częstość, A - amplituda  % t - wektor czasu, n - rzad ciagu  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\*(-2)/pi + A/pi + sin(w\*t)\*A/2;  end  end |

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

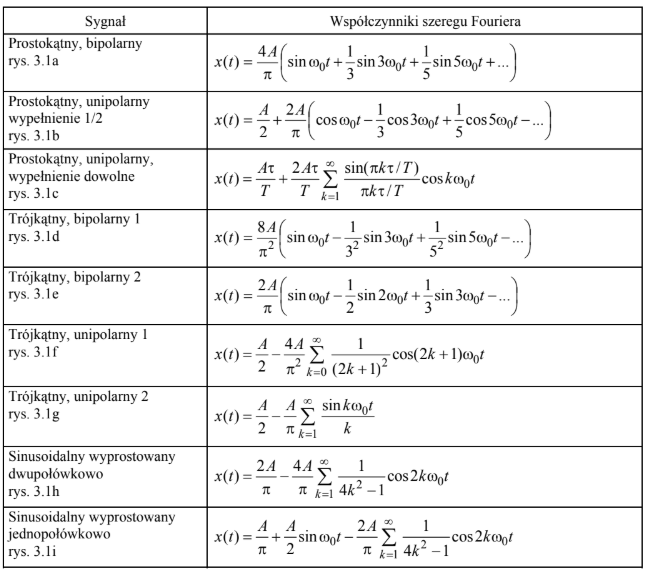


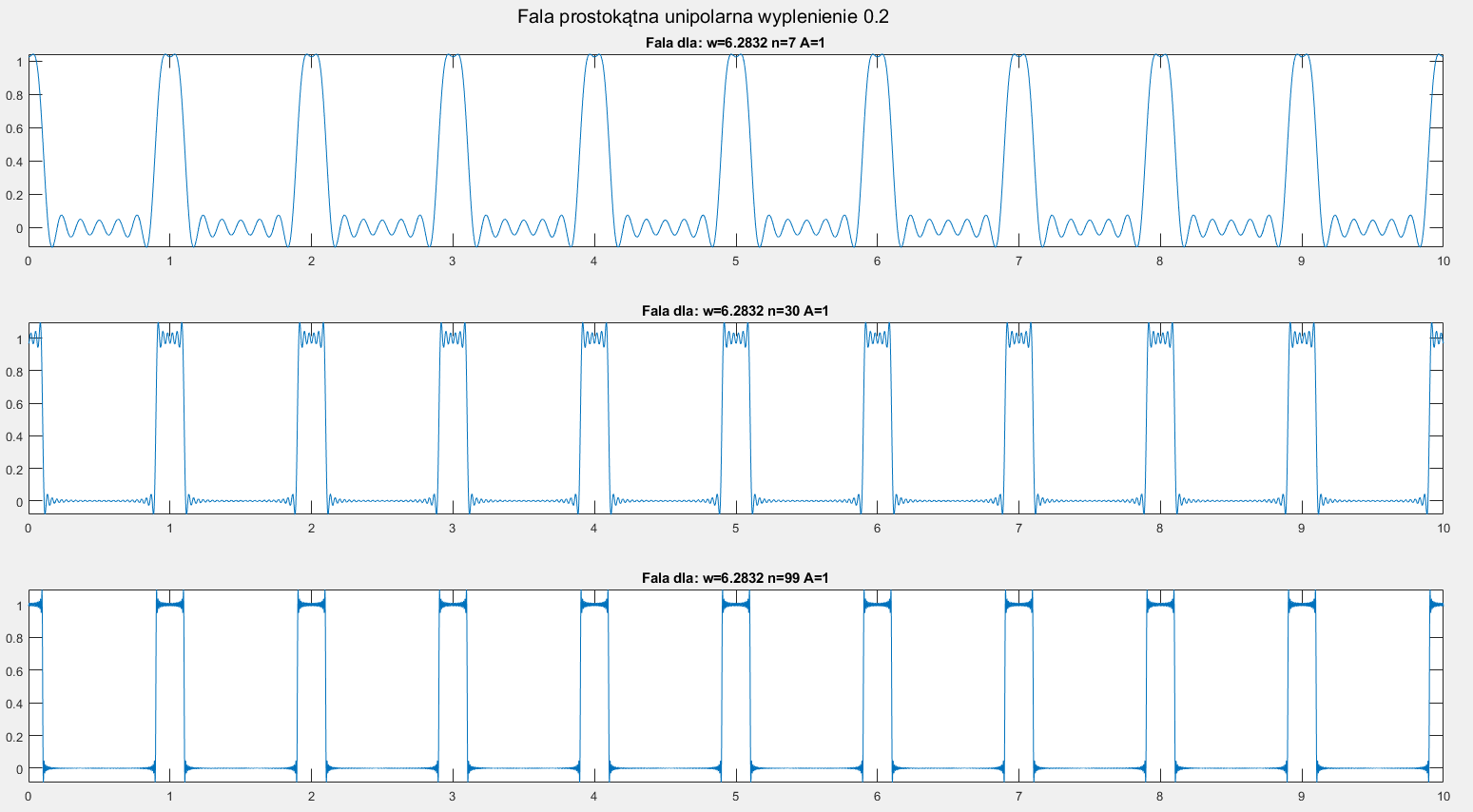
Rysunek 4.1. Fala prostokątna bipolarna.



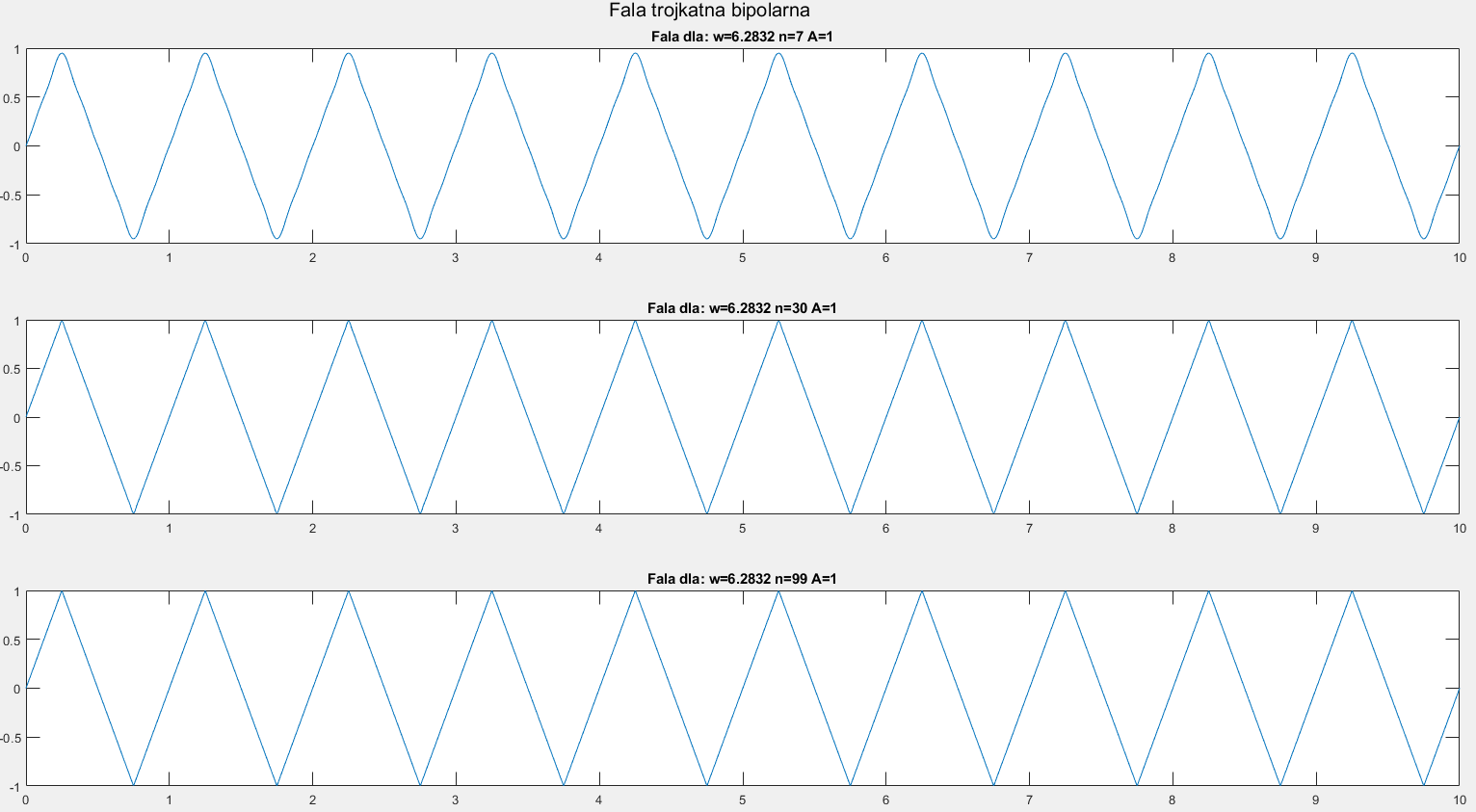
Rysunek 4.2. Fala unipolarna o wypełnieniu 1/2.

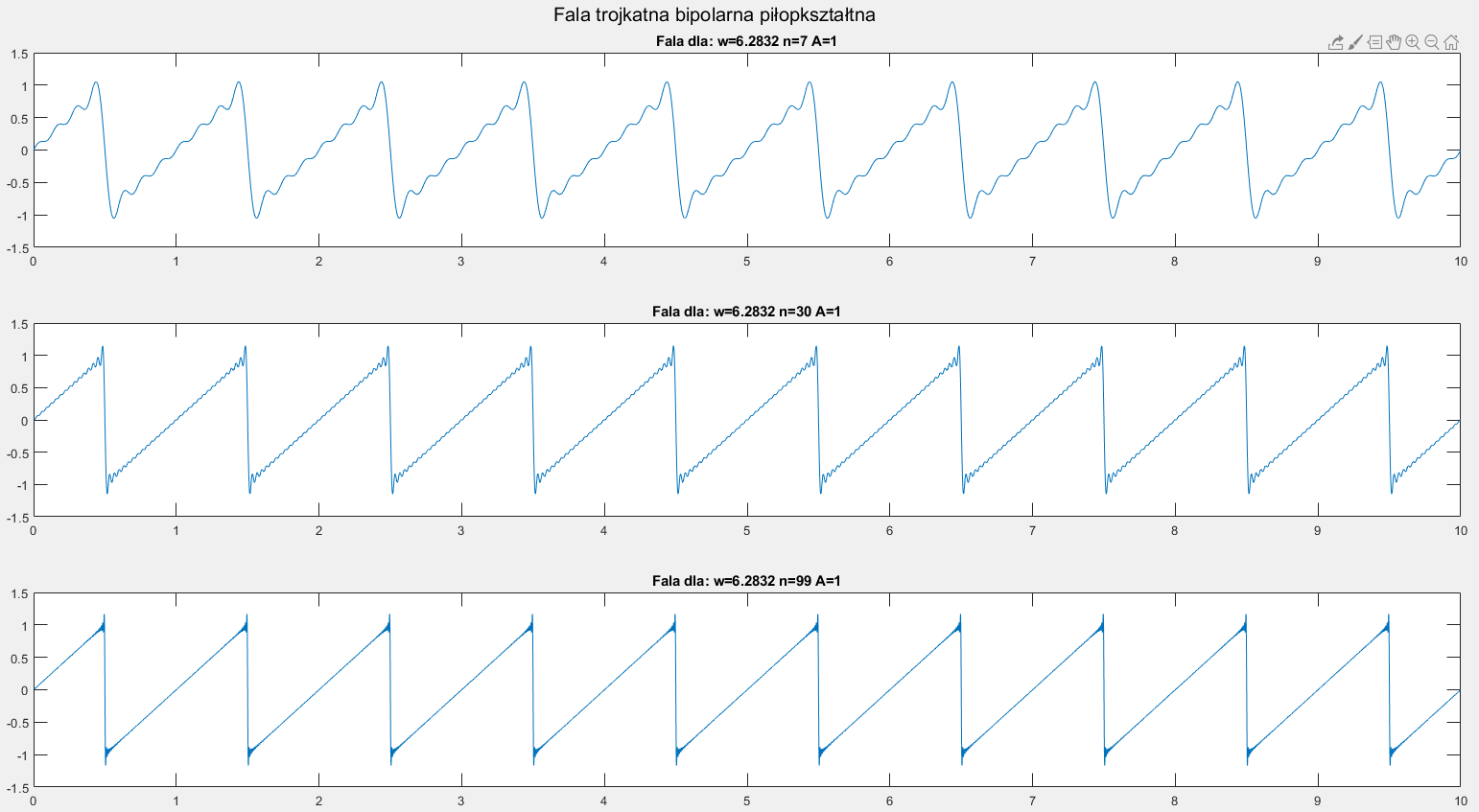
Tab. 4.2. Wzory sygnałów używanych w programie.





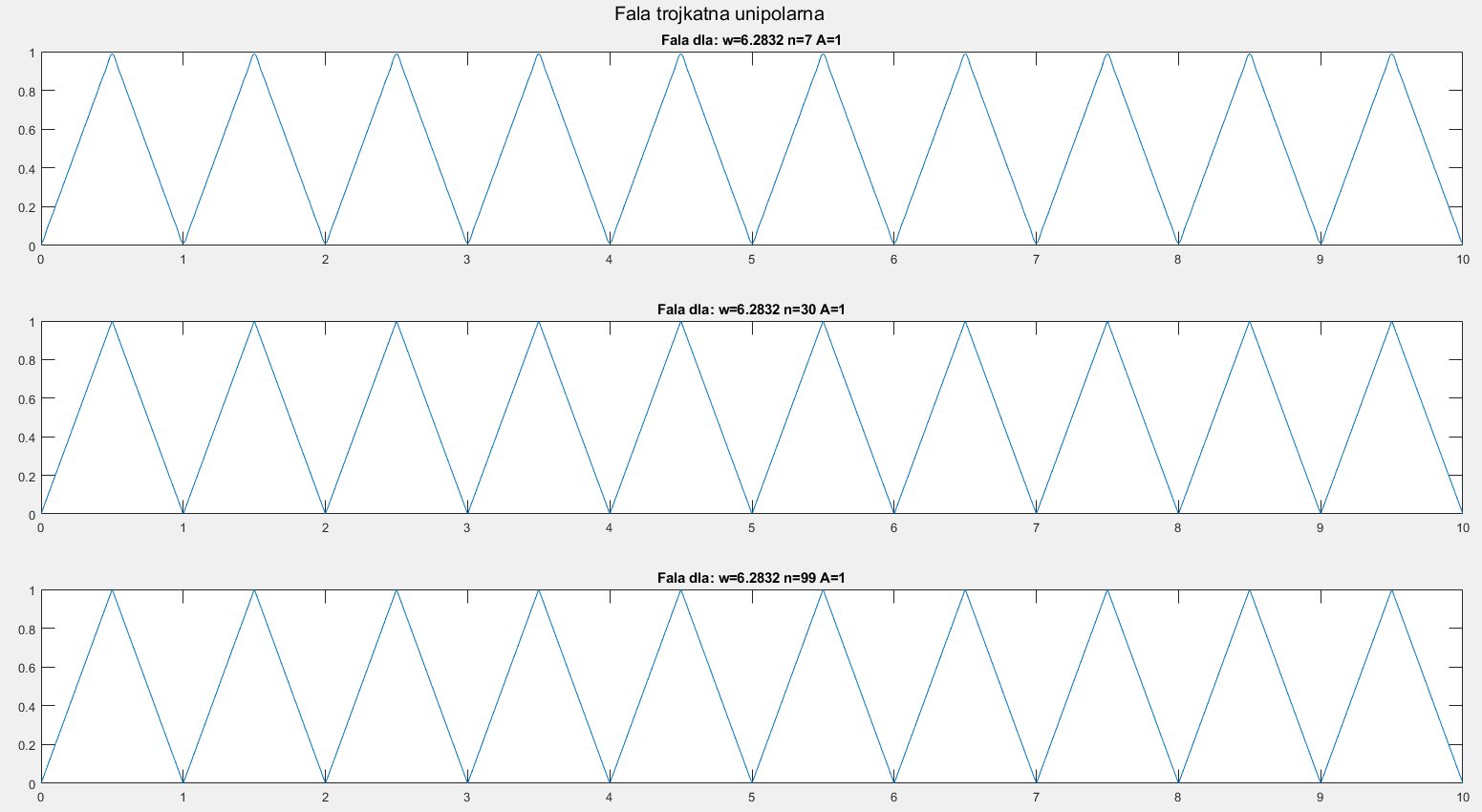
Rysunek 4.3. Fala unipolarna o wypełnieniu dowolnym.

Rysunek 4.4. Fala trójkątna bipolarna.

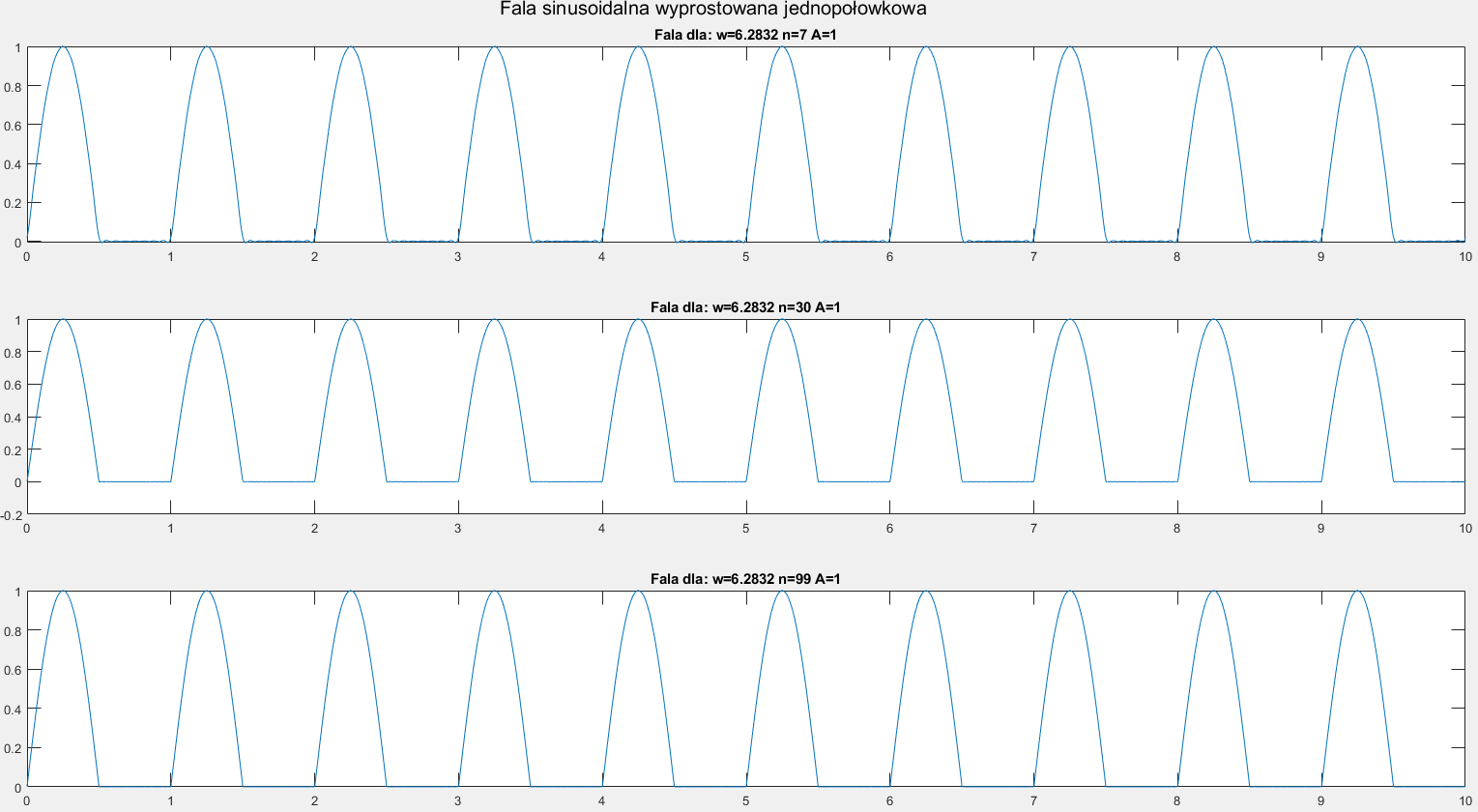


Rysunek 4.5. Fala piłokształtna bipolarna.

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.

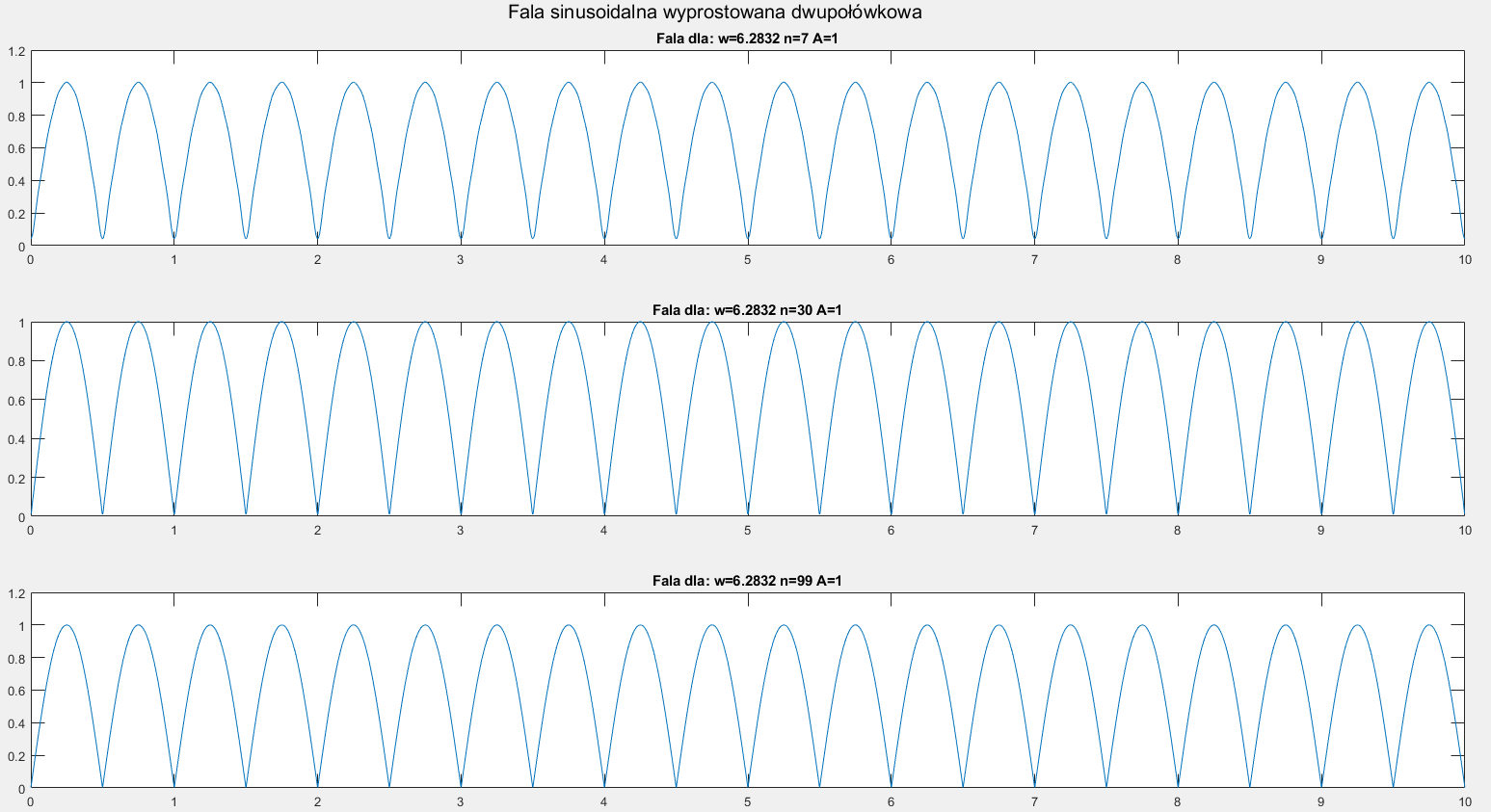


Rysunek 4.6. Fala trójkątna unipolarna.

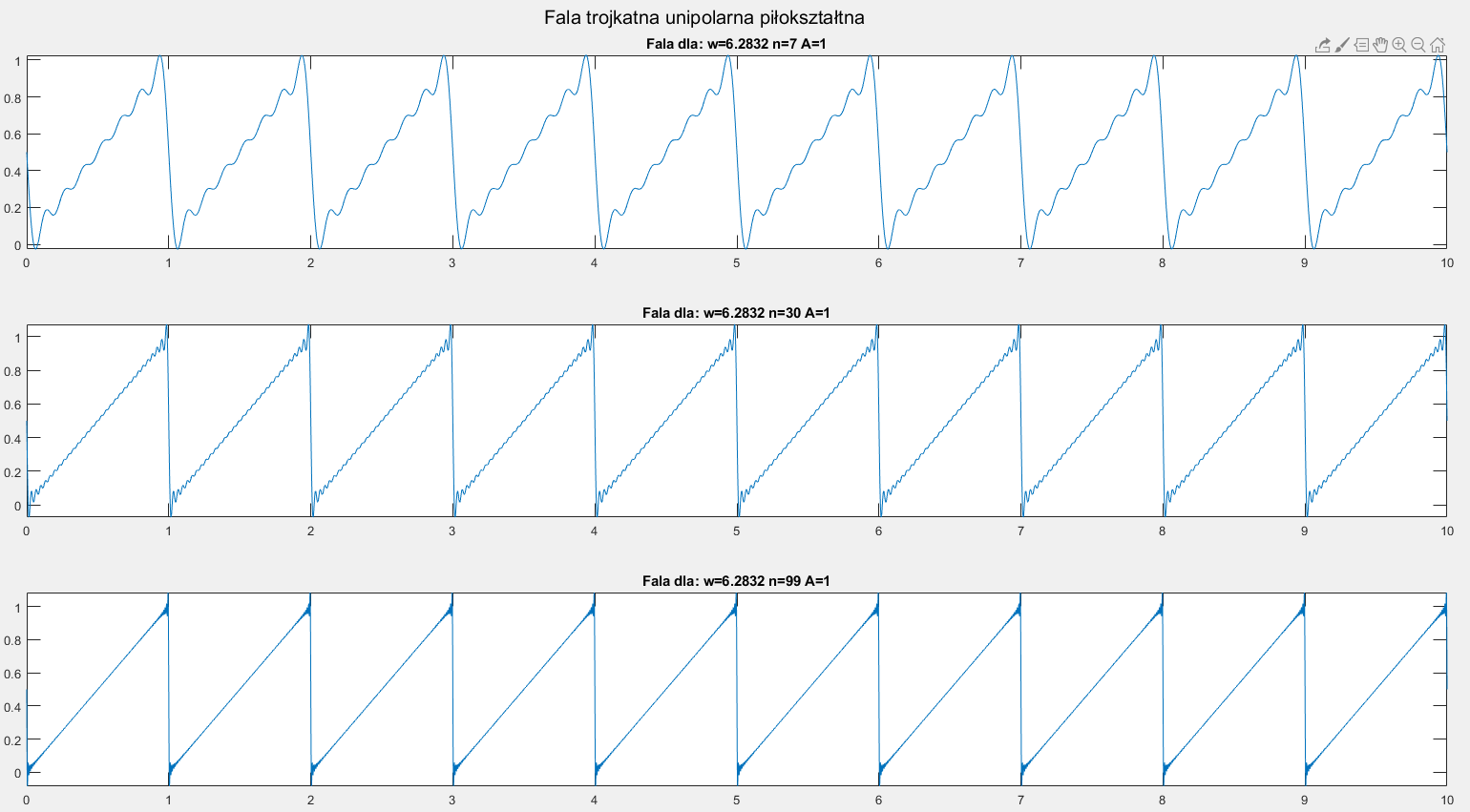


Rysunek 4.7. Fala sinusoidalna jednopołówkowa, wyprostowana.

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



Rysunek 4.8. Fala sinusoidalna dwupołówkowa, wyprostowana.



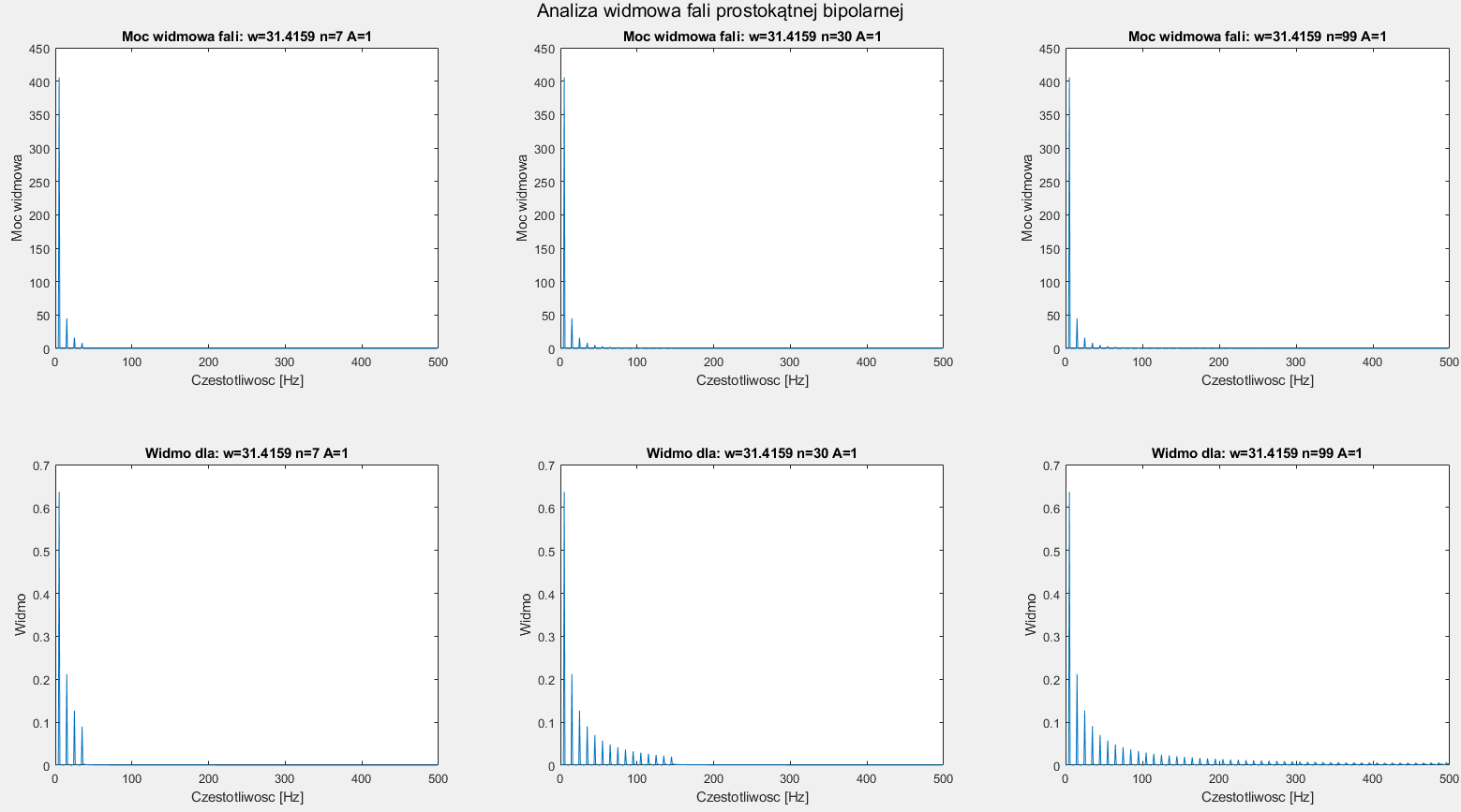
Rysunek 4.9. Fala piłokształtna unipolarna.

# Laboratorium 5

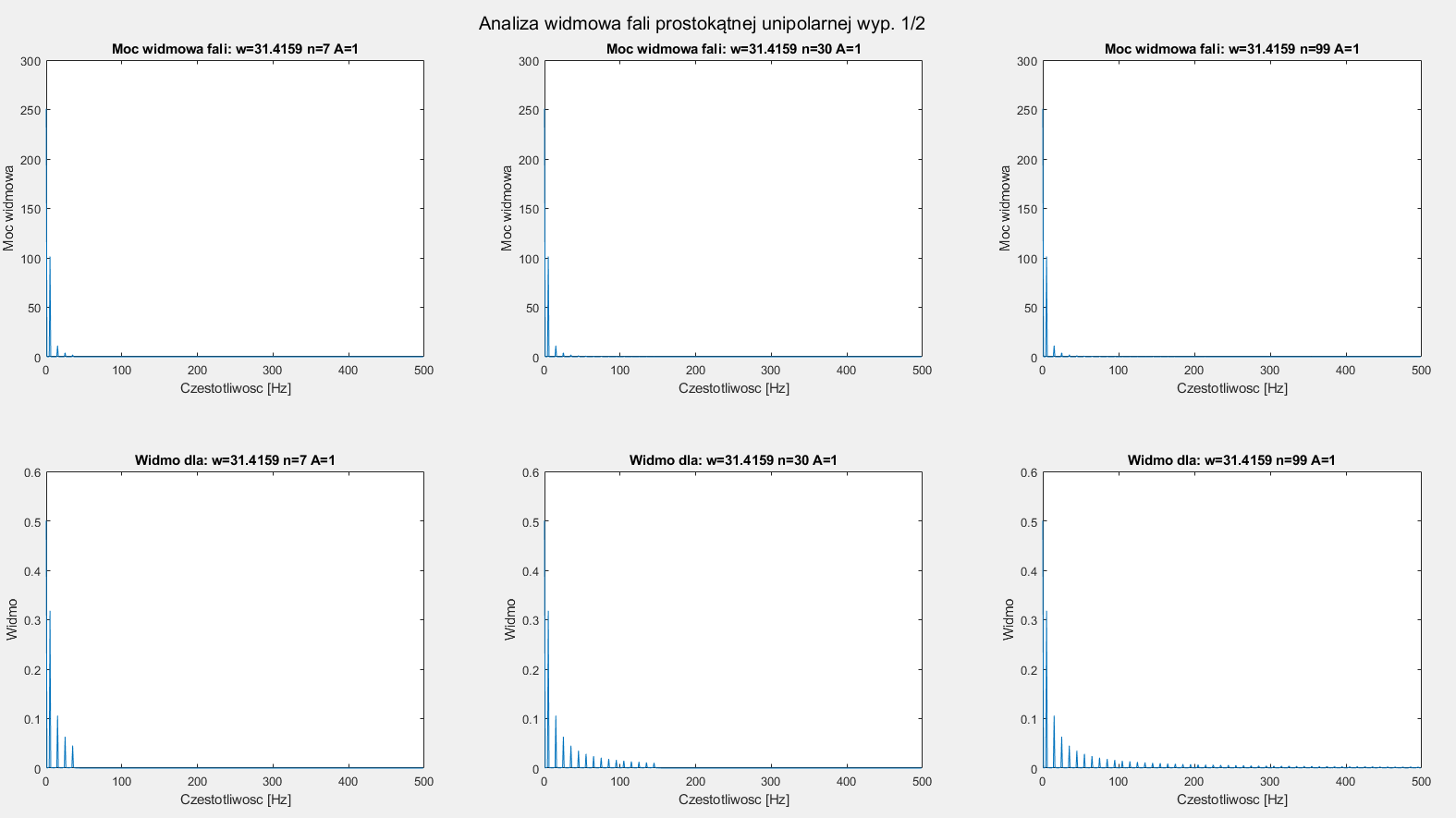
W ramach tego ćwiczenia zrealizowana została analiza częstotliwościowa fal obliczony i pokazanych w poprzednim rozdziale. Dodatkowo dokonana zostanie analiza układów RLC i RC. Stworzona została funkcja [f, M, W]=fft\_from\_signal(x, fs). Funkcja ta znajduje się na końcu pliku oraz w osobnym pliku tak 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\_Mateusz\_Lab\_5\_1”.

|  |
| --- |
| % Lab 5. Procedura szybiej transfomraty Fouriera  % Defincija funkcji obliczającej FFT z sygnałów  % znajduje sie 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 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)]);  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('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 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, wypelnienie 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 wyplenienie ' 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 wyplenieniu ' 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)]);  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 bipolarny  % Generowanie wykresu  y = tbp(w, A, t, n);    % Wykresy  figure(7)  sgtitle(['Fala trojkatna 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)]);  xlabel('Czas [s]');  end    % Obliczenie 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    %% Sygnal trojkatny bipolarny piloksztaltny  % Generowanie wykresu  y = tbpp(w, A, t, n);    % Wykresy  figure(9)  sgtitle(['Fala trojkatna 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    % Obliczenie FFT  [f\_w, M, W] = fft\_from\_signal(y, fs);  figure(10);  sgtitle(['Analiza widmowa fali trojkatnej bipolarnej piłopkształtnej']);  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 piloksztaltna  % Generowanie wykresu  y = tupp(w, A, t, n);    % Wykresy  figure(13)  sgtitle(['Fala trojkatna unipolarna piłokształtna']);  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 piłokształtnej']);  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 jednopołowkowy  % Generowanie wykresu  y = swj(w, A, t, n);    % Wykresy  figure(17)  sgtitle(['Fala sinusoidalna wyprostowana jednopołowkowa']);  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 jednopołowkowej']);  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 sepctrum, 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 |

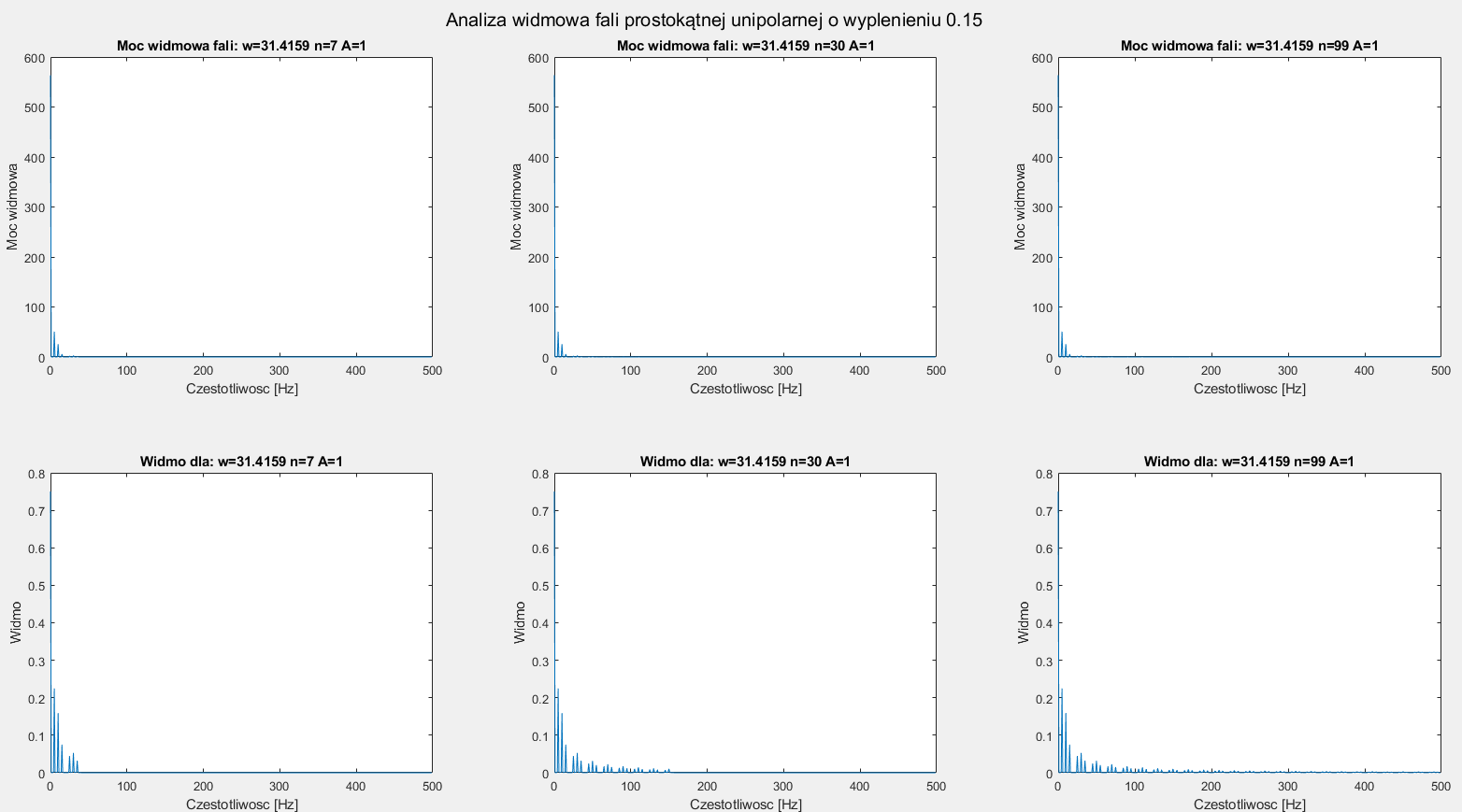


Rys. 5.1. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej bipolarnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

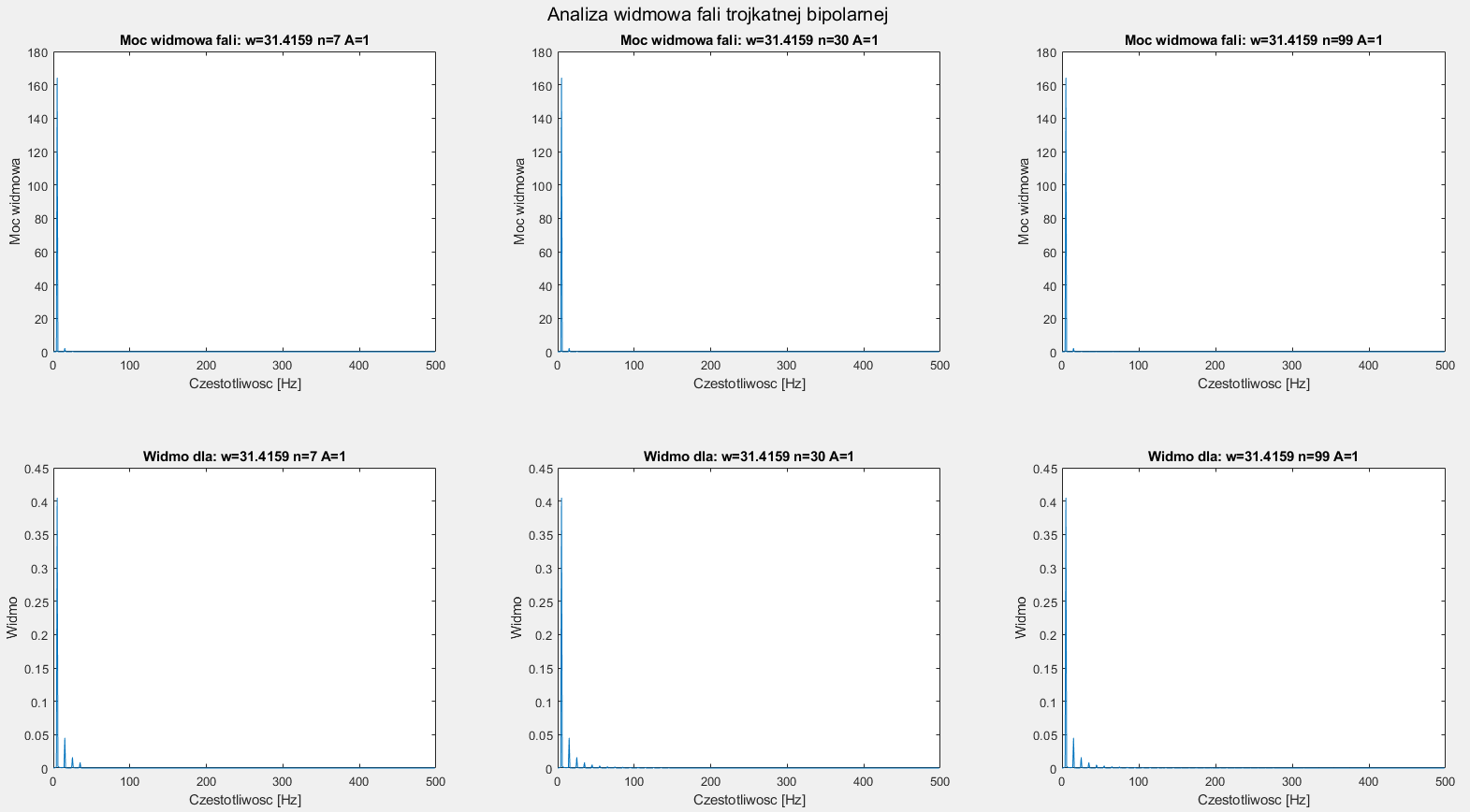


Rys. 5.2. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej unipolarnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

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.

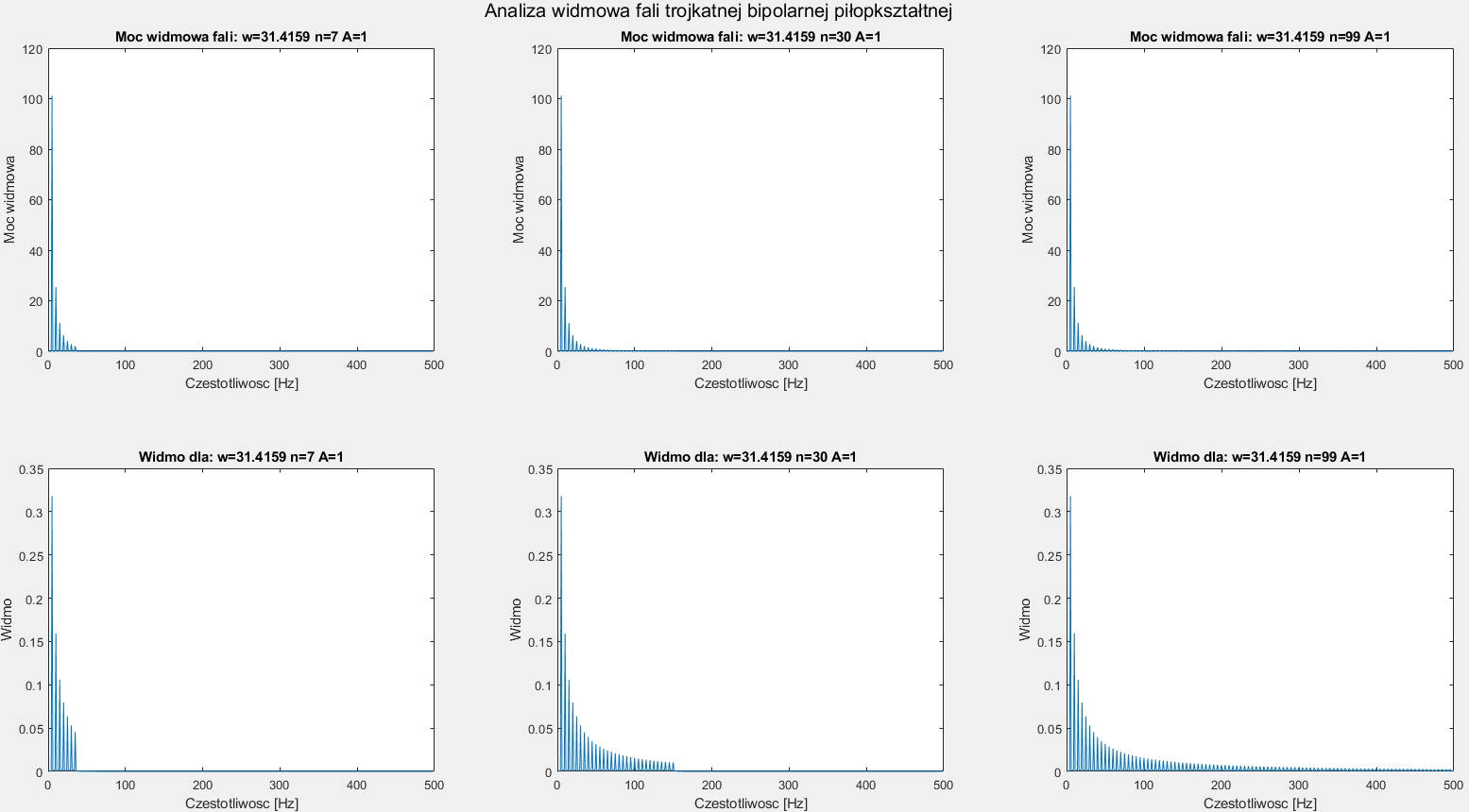


Rys. 5.3. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej unipolarnej o wypełnieniu 0.15. Każda kolumna odpowiada większęmu rzędowi ciągu.

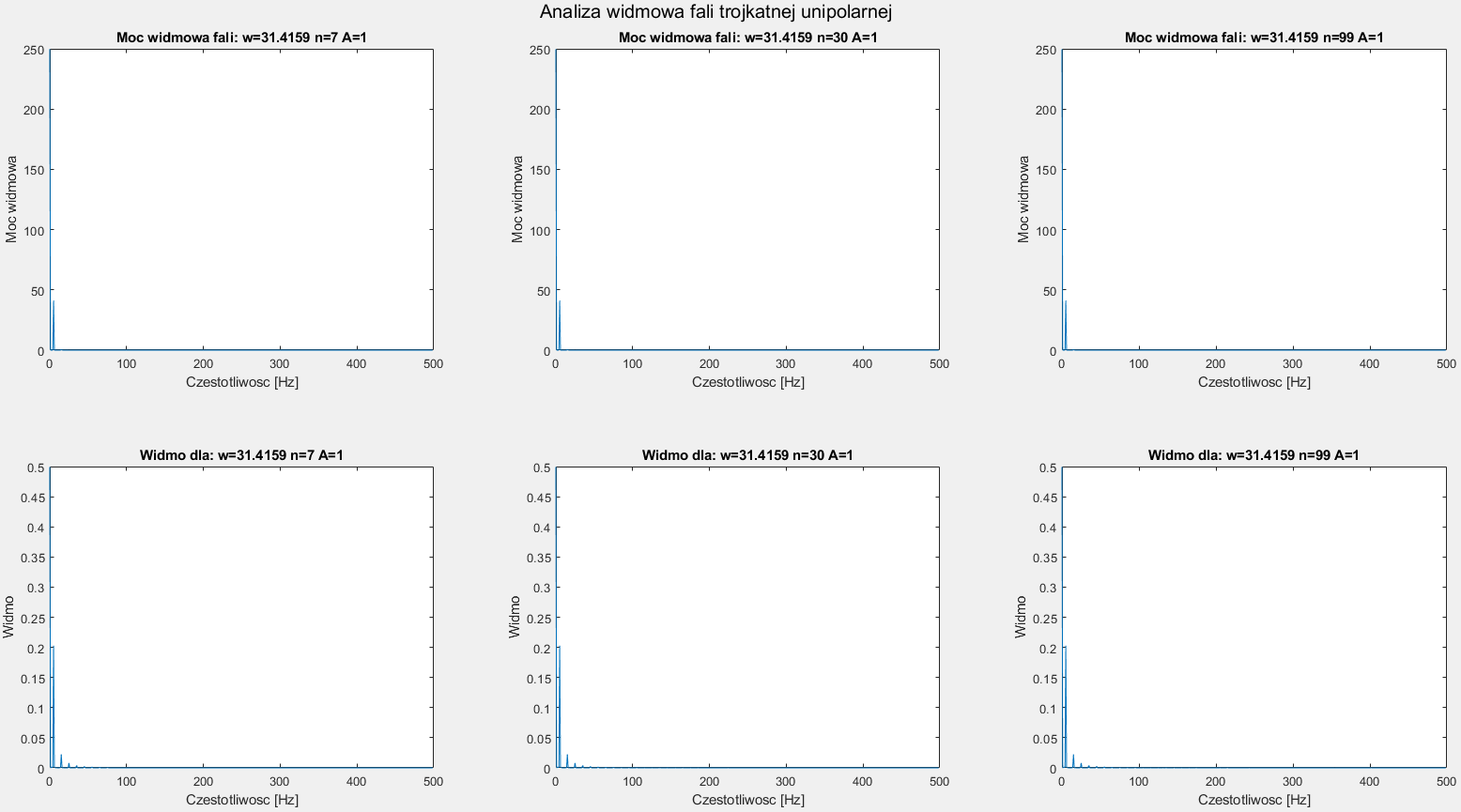


Rys. 5.4. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej bipolarnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

Na wykresie 6.4. można zauważyć 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.

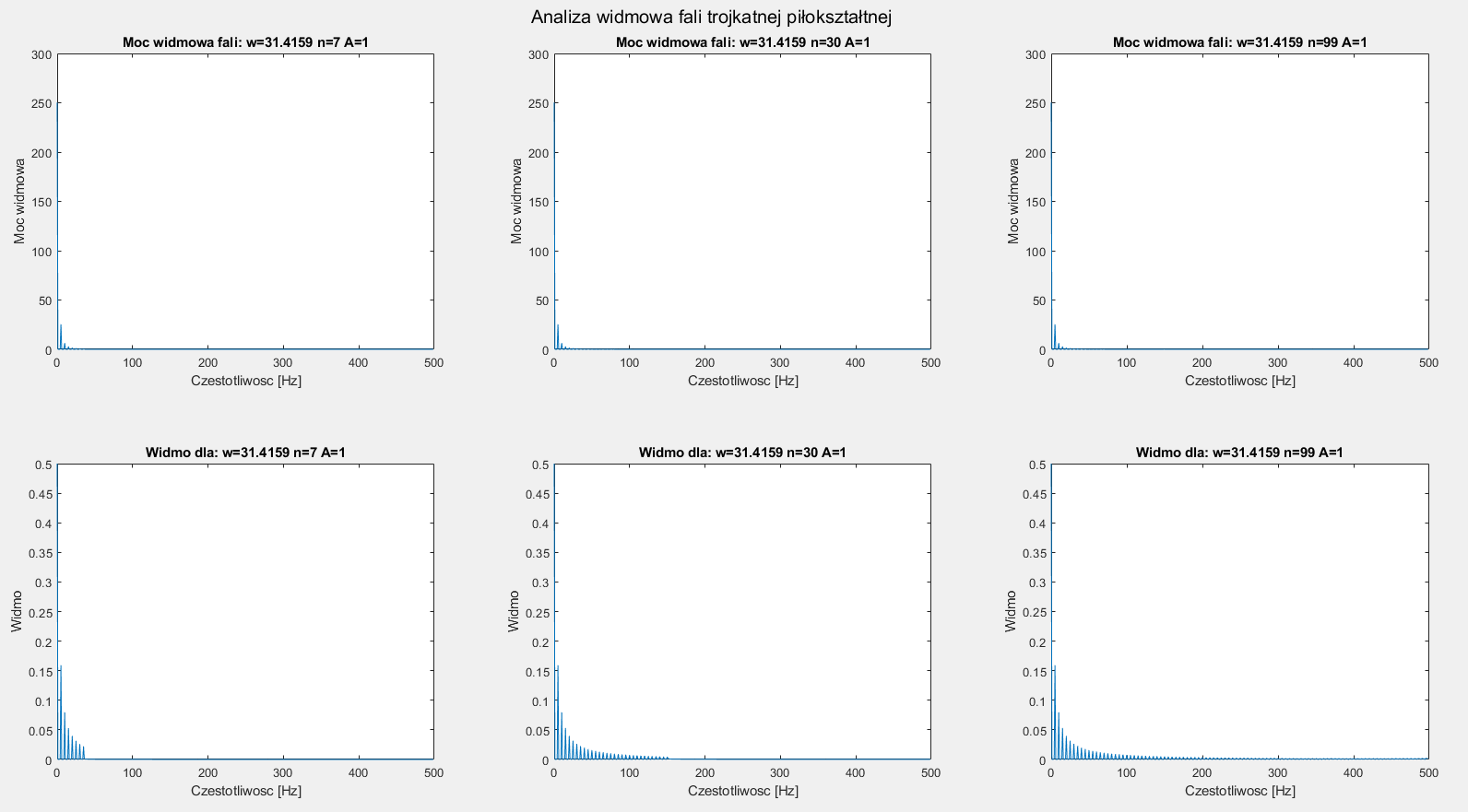


Rys. 5.5. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej bipolarnej piłokształtnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

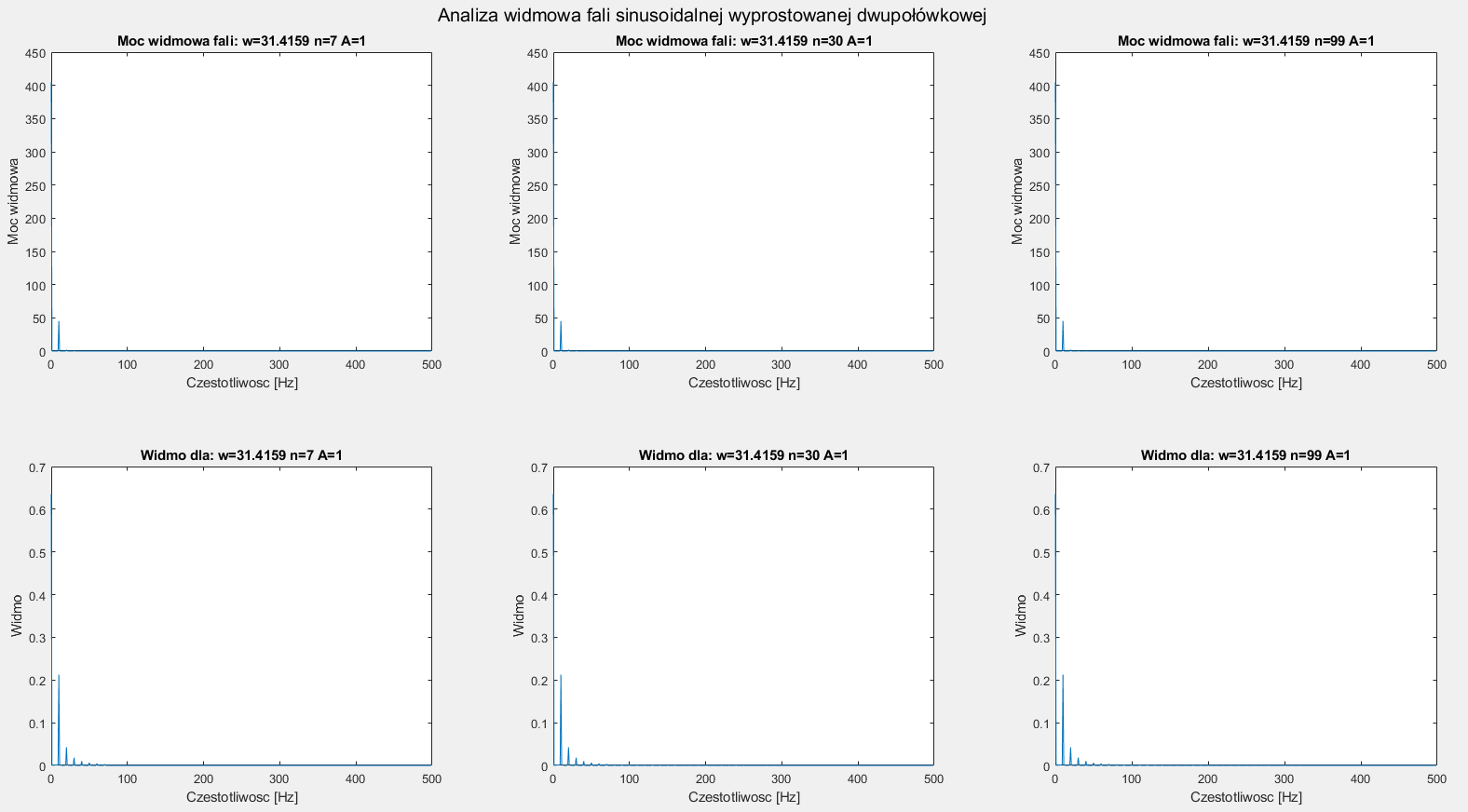


Rys. 5.6. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej unipolarnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

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.

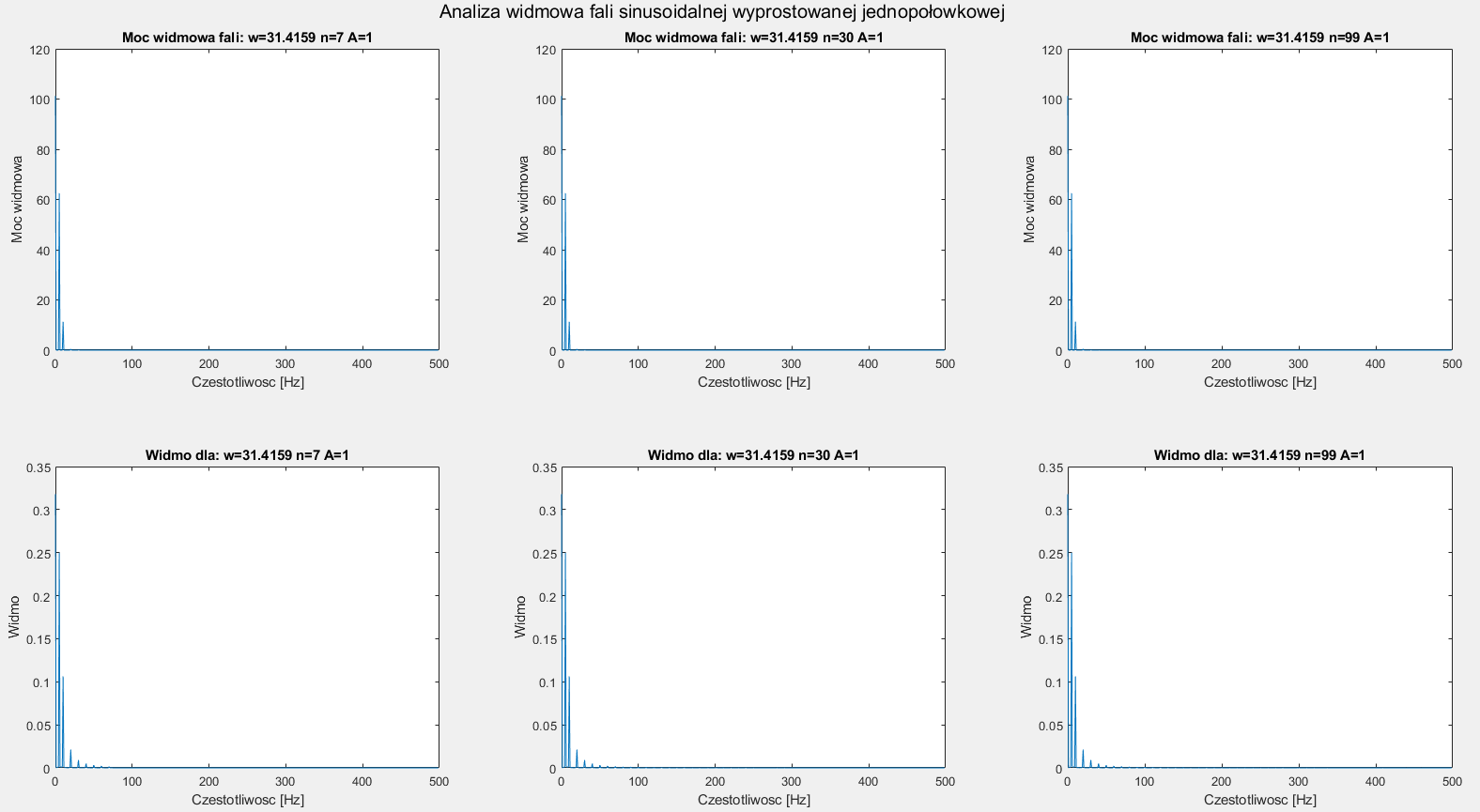


Rys. 5.7. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej unipolarnej piłokształtnej. Każda kolumna odpowiada większęmu rzędowi ciągu.



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

Sygnały sinusoidalnie pochodne aproksymowana są już w pierwszych elementach ciągu.



Rys. 5.9. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali sinusoidalnej jednopołówkowej wyprostowanej. Każda kolumna odpowiada większęmu rzędowi ciągu.

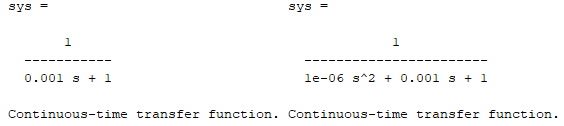
Następnie po analizie sygnałów podstawowych przeprowadzona została analiza układów RC i RLC.

Tab. 5.2. Kod programu "Krupnik\_Mateusz\_Lab\_5\_2.m".

|  |
| --- |
| % Lab 5. Analiza układów RC i RLC  % Mateusz Krupnik  %% Analiza układów RC  % 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 - czestotliwosciowa  figure(2)  impulse(L,M) % odpowiedz impulsowa ukladu  figure(3)  step(L,M) % odpowiedz skokowa ukladu  figure(4)  iopzplot(sys) % wykres zer i biegunow dla ukladow wej/ wyj  [z,p,k]=tf2zp(L,M) % konwersja Transfer Function na zera i bieguny    %% Analiza układu RLC  % Dane ukladu  R=1000; C=10^(-6); Li=1;  L=[1]; M=[(Li\*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 - czestotliwosciowa  figure(6)  impulse(L,M) % odpowiedz impulsowa ukladu  figure(7)  step(L,M) % odpowiedz skokowa ukladu  figure(8)  iopzplot(sys) % wykres zer i biegunow dla ukladow wej/ wyj  [z,p,k]=tf2zp(L,M) % konwersja Transfer Function na zera i bieguny |

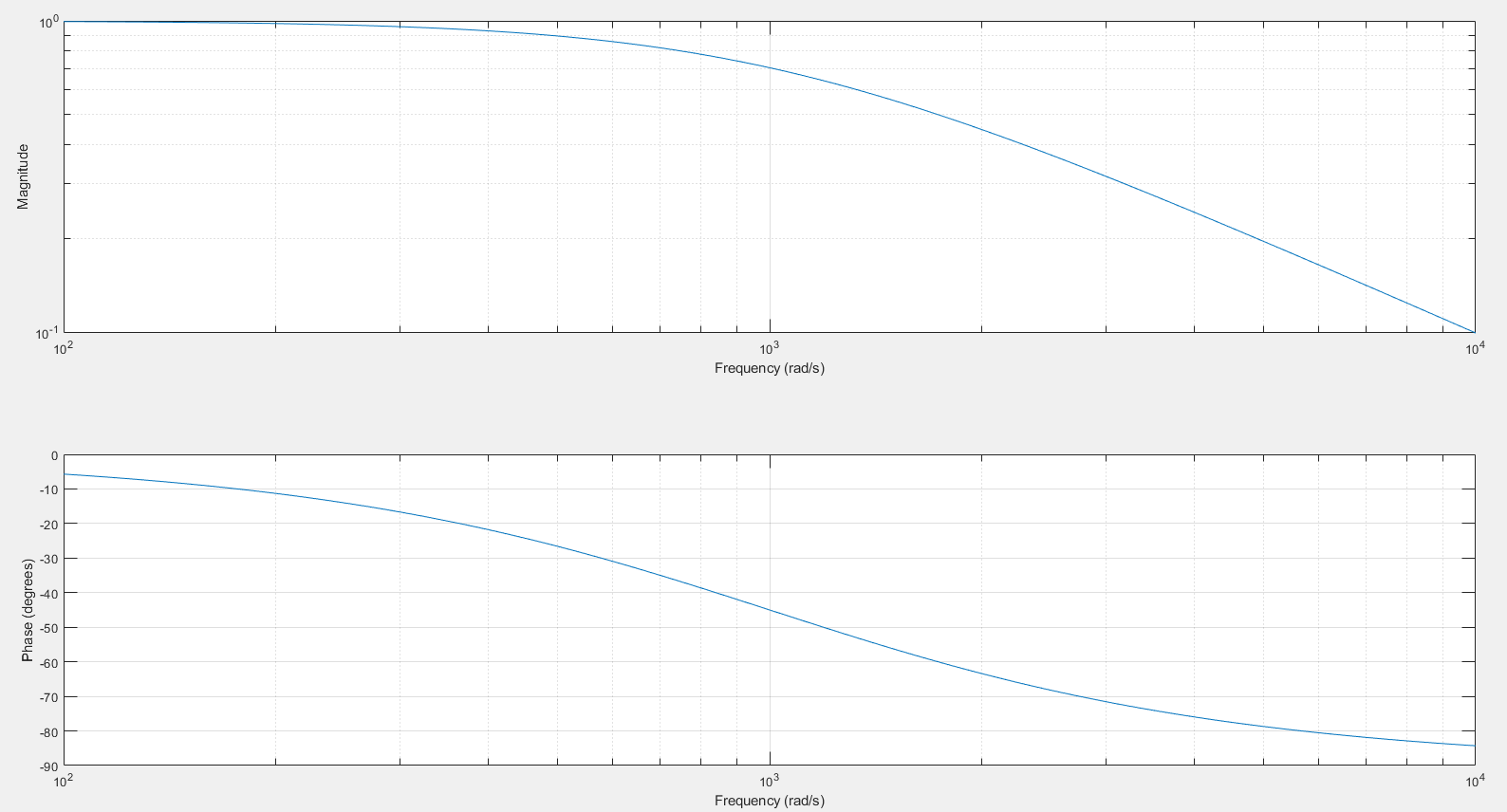
Przyjęto parametry dla układu RC: R=1000[Ω], C=10-6[F] oraz dla układu RLC: R=1000[Ω], C=10-6[F], L=1[H]. Za pomocą tych wartości zdefiniowane zostały liczniki i mianowniki transmitancji układów danymi wzorami poniżej.

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.

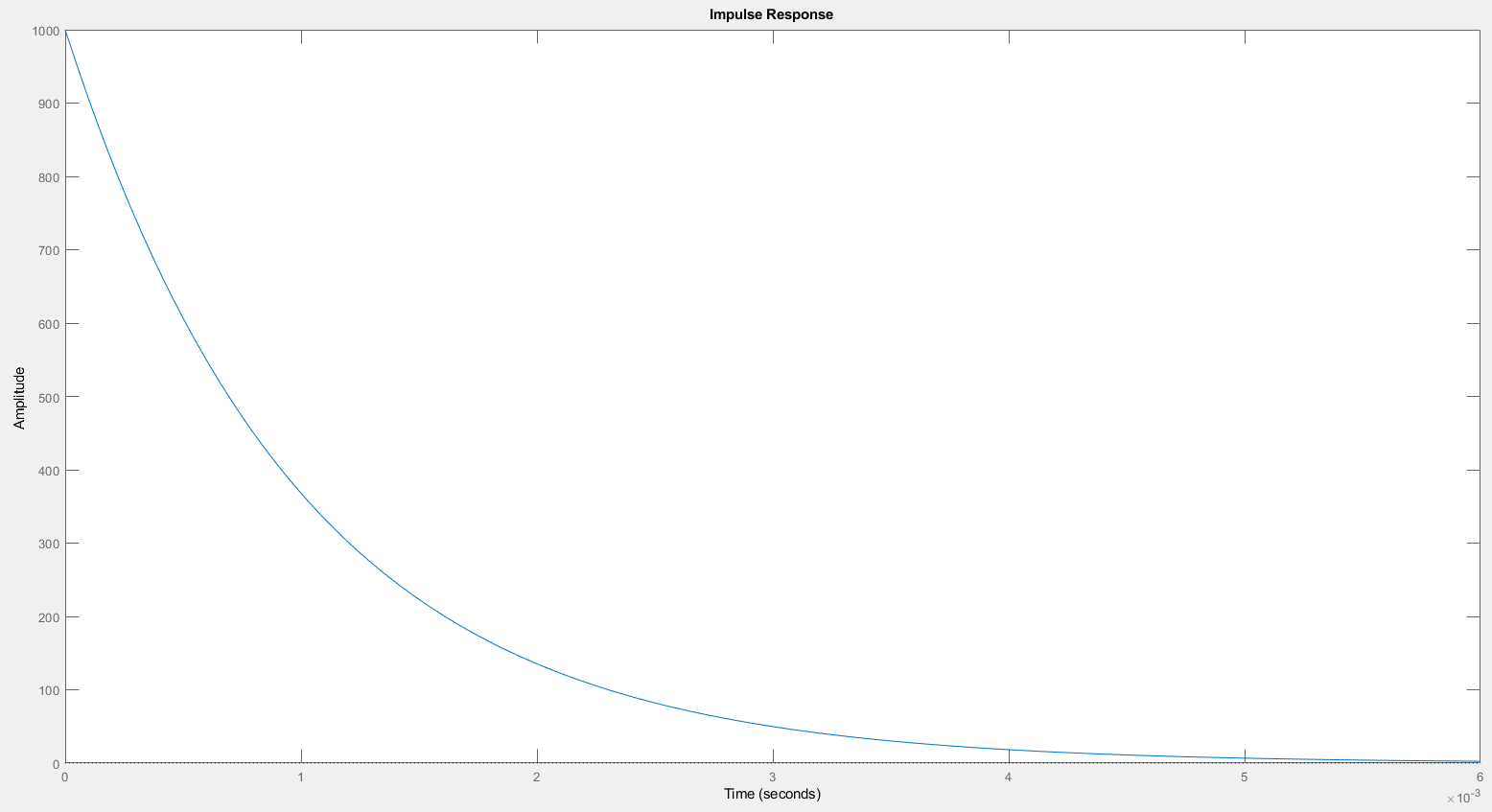


Rys. 5.10. Wynik z funkcji tf(L, M).

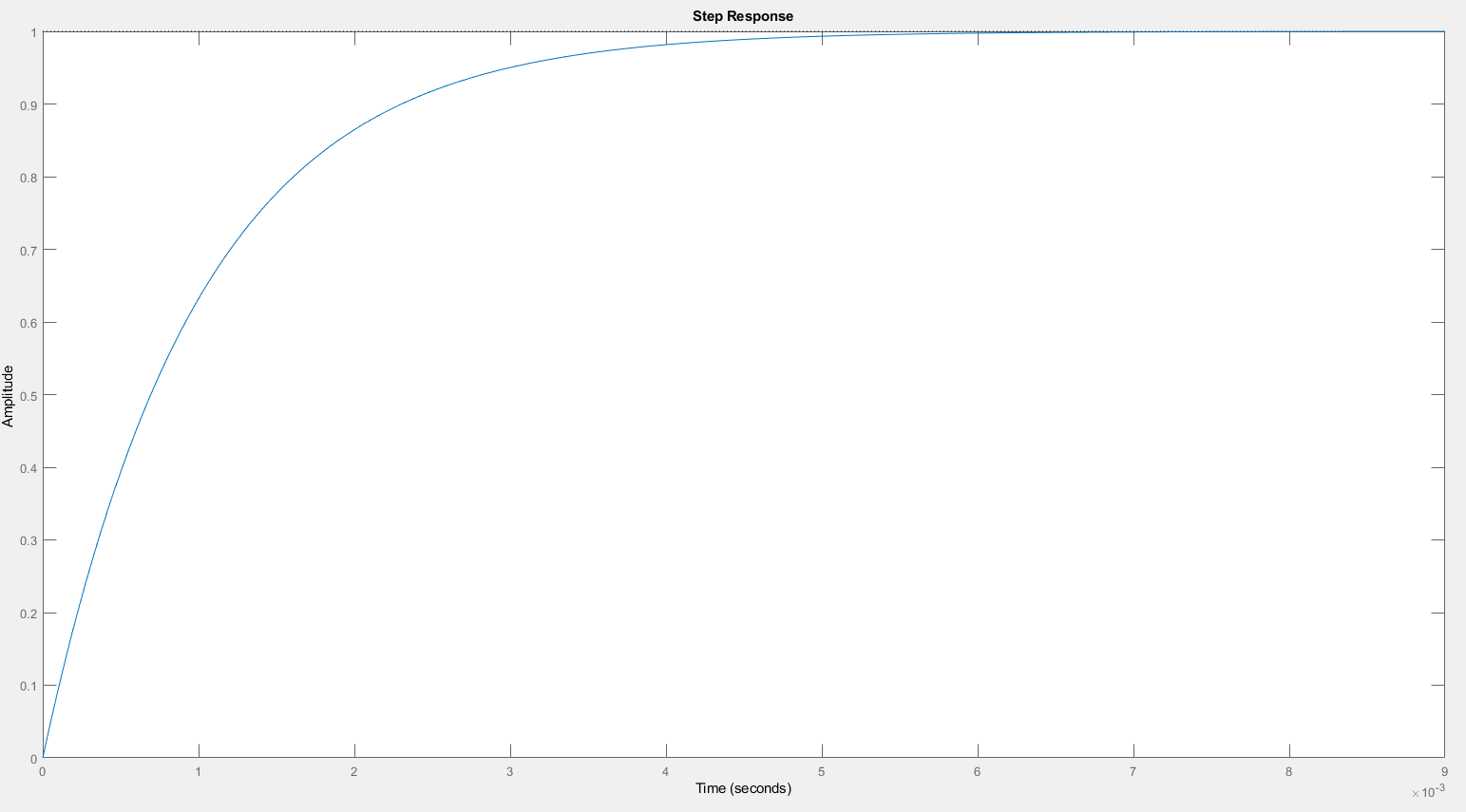
Następnie dla stworzonych modeli wyznaczone zostały charakterystyki częstotliwościowe i czasowe za pomocą funkcji freqs(L, M), step(L, M), impulse(L, M). Są to funkcje zwracające charakterystyki Bodego (amplitudowo i fazowo-częstotliwościowe), skokową i impulsową. Ostatnim etapem było wykreślenie zer i biegunów układu za pomocą funkcji iopzplot(sys) oraz transformacja układu opisanego wielomianami na postać biegunową. Służy do tego funkcja tf2zp(sys).



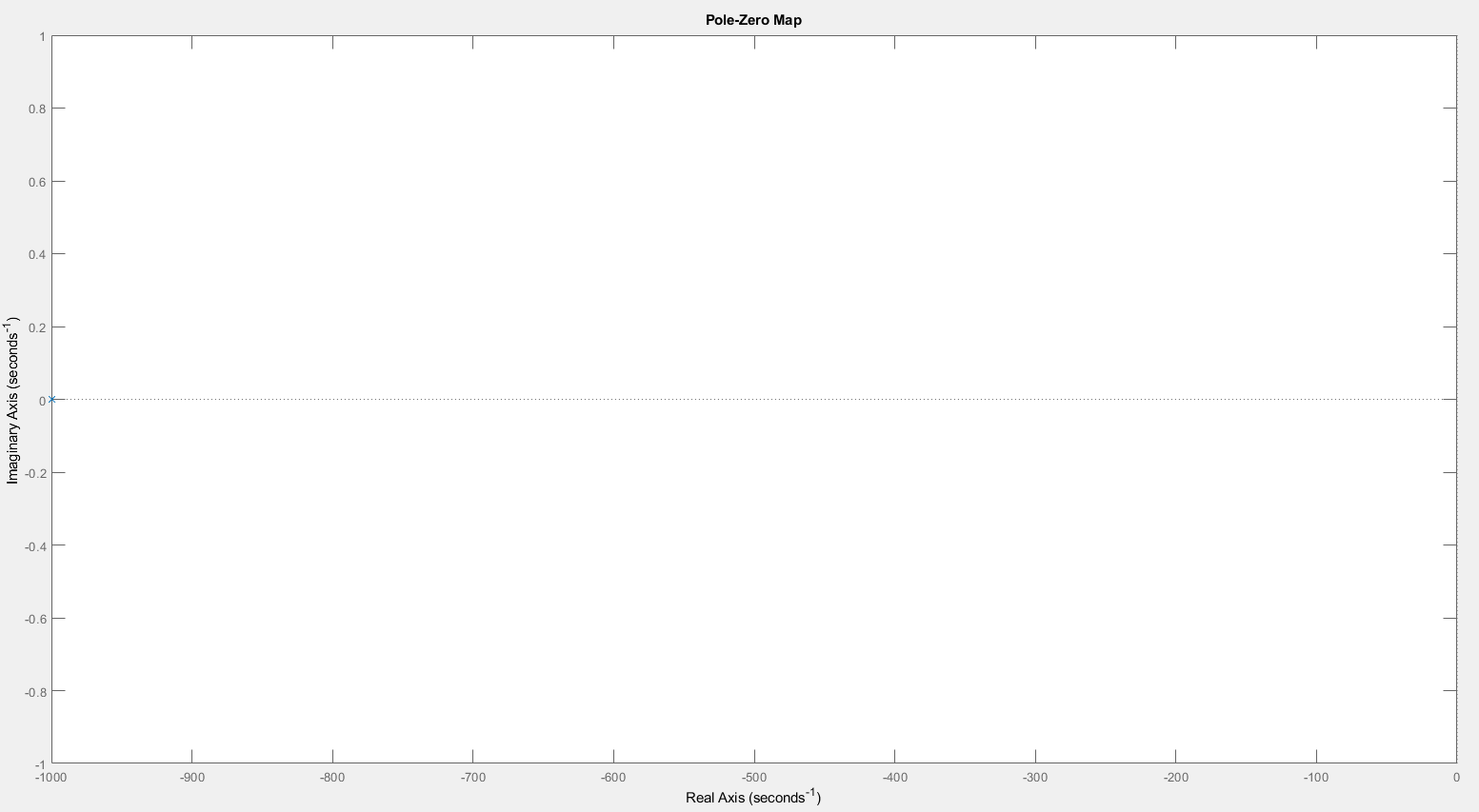
Rys. 5.11. Charakterystyki Bodego dla układu RC.



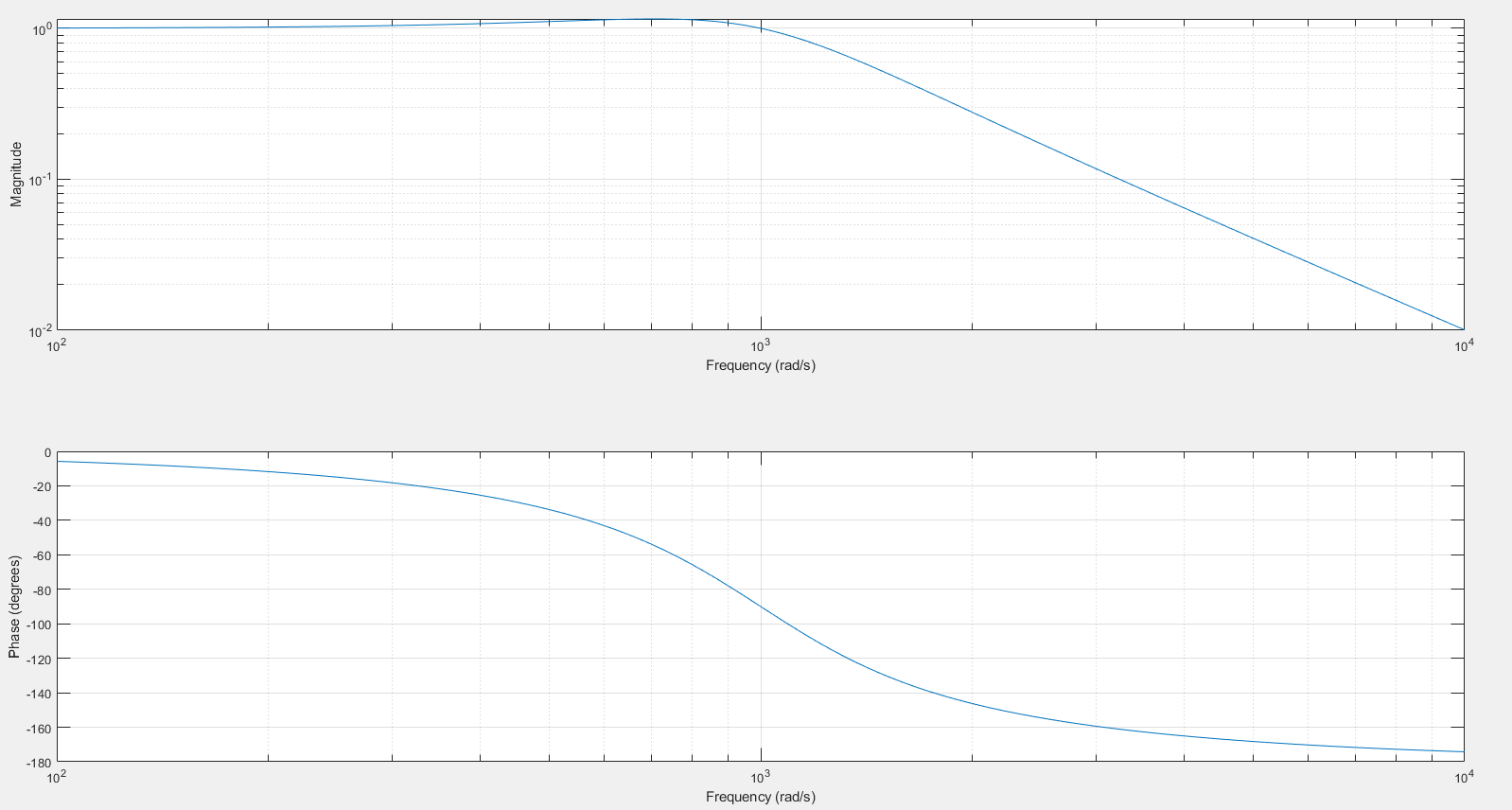
Rys. 5.12. Odpowiedź impulsowa układu RC.



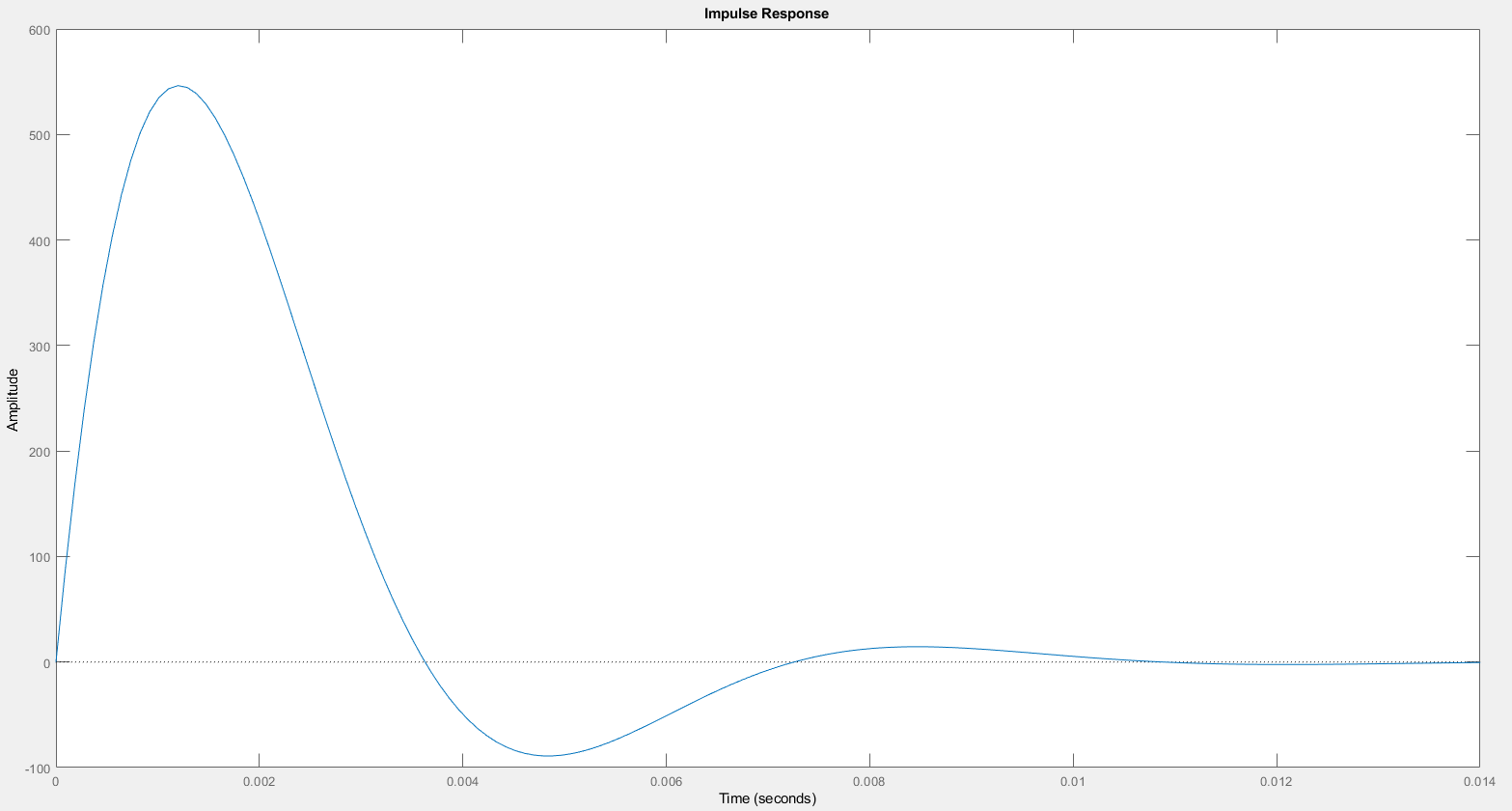
Rys. 5.13. Odpowiedź skokowa układu RC.



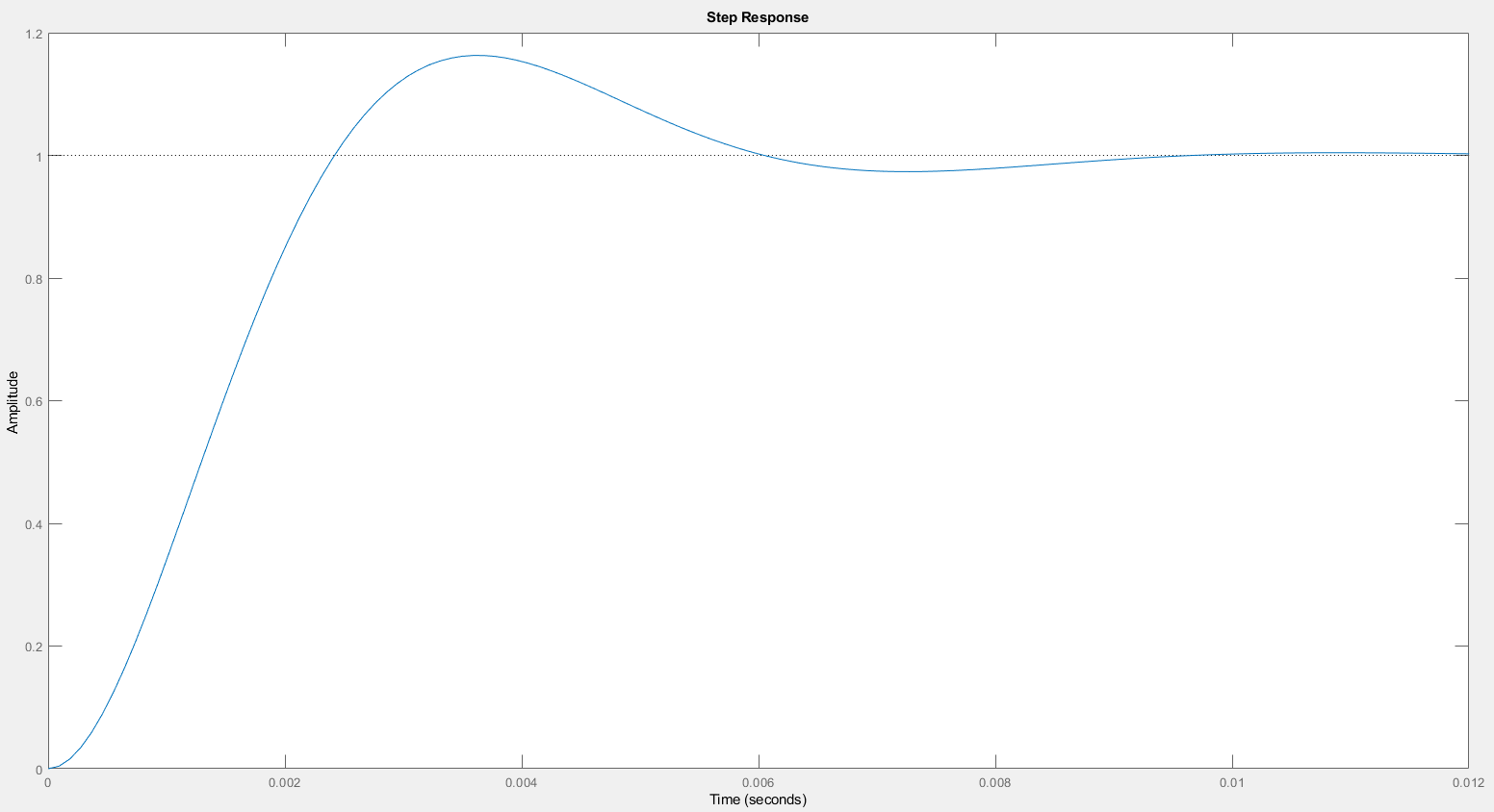
Rys. 5.14. Biegun układu RC (prawie w zerze).



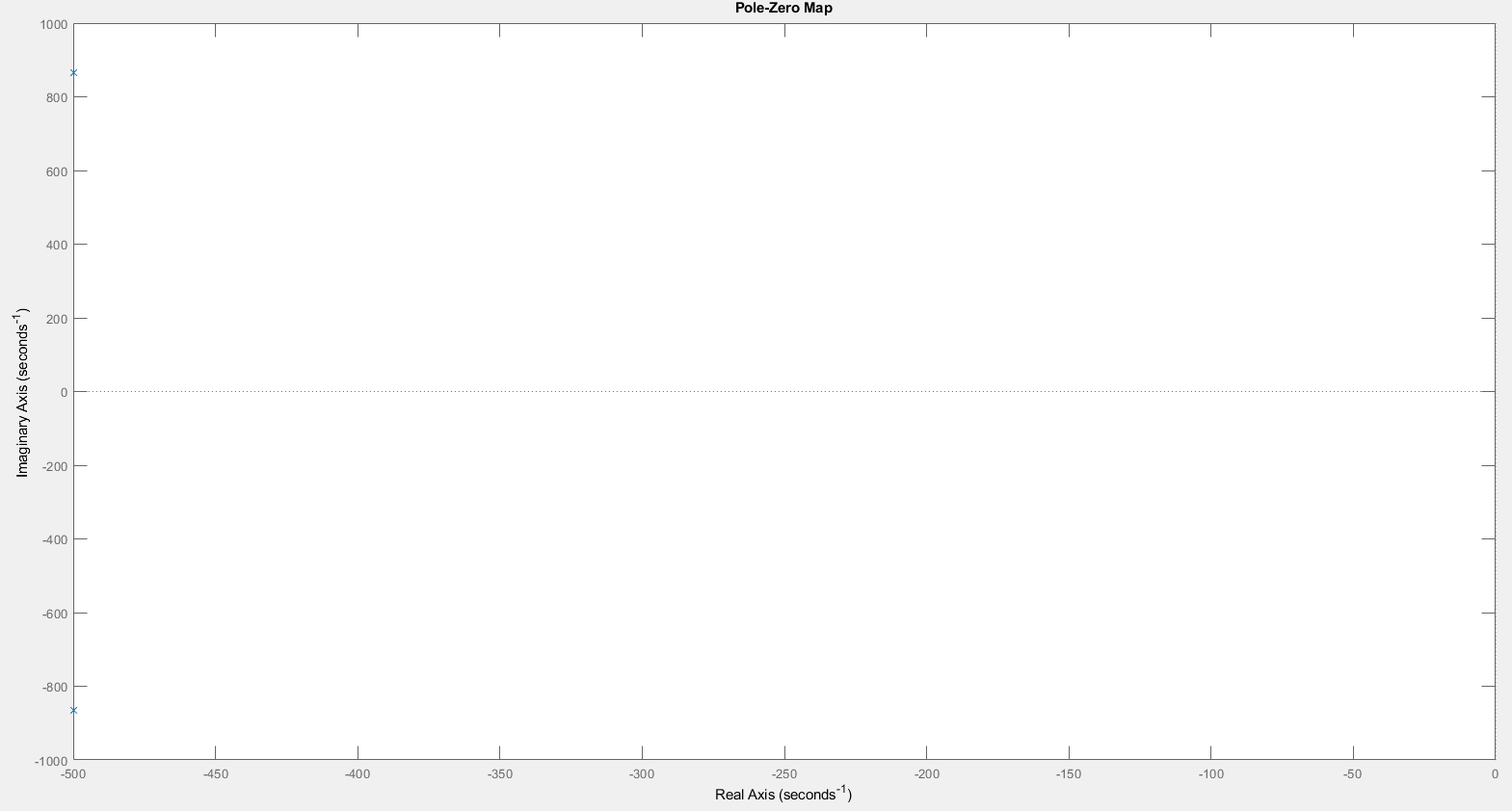
Rys. 5.15. Charakterystyki Bodego dla układu RLC.



Rys. 5.16. Odpowiedź impulsowa układu RLC.



Rys. 5.17. Odpowiedź skokowa układu RLC.



Rys. 5.18. Bieguny układu RLC, wzajemnie sprzężone.

Na podstawie analizy transmitancji układu możemy określić jego stabilność. Bieguny 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.

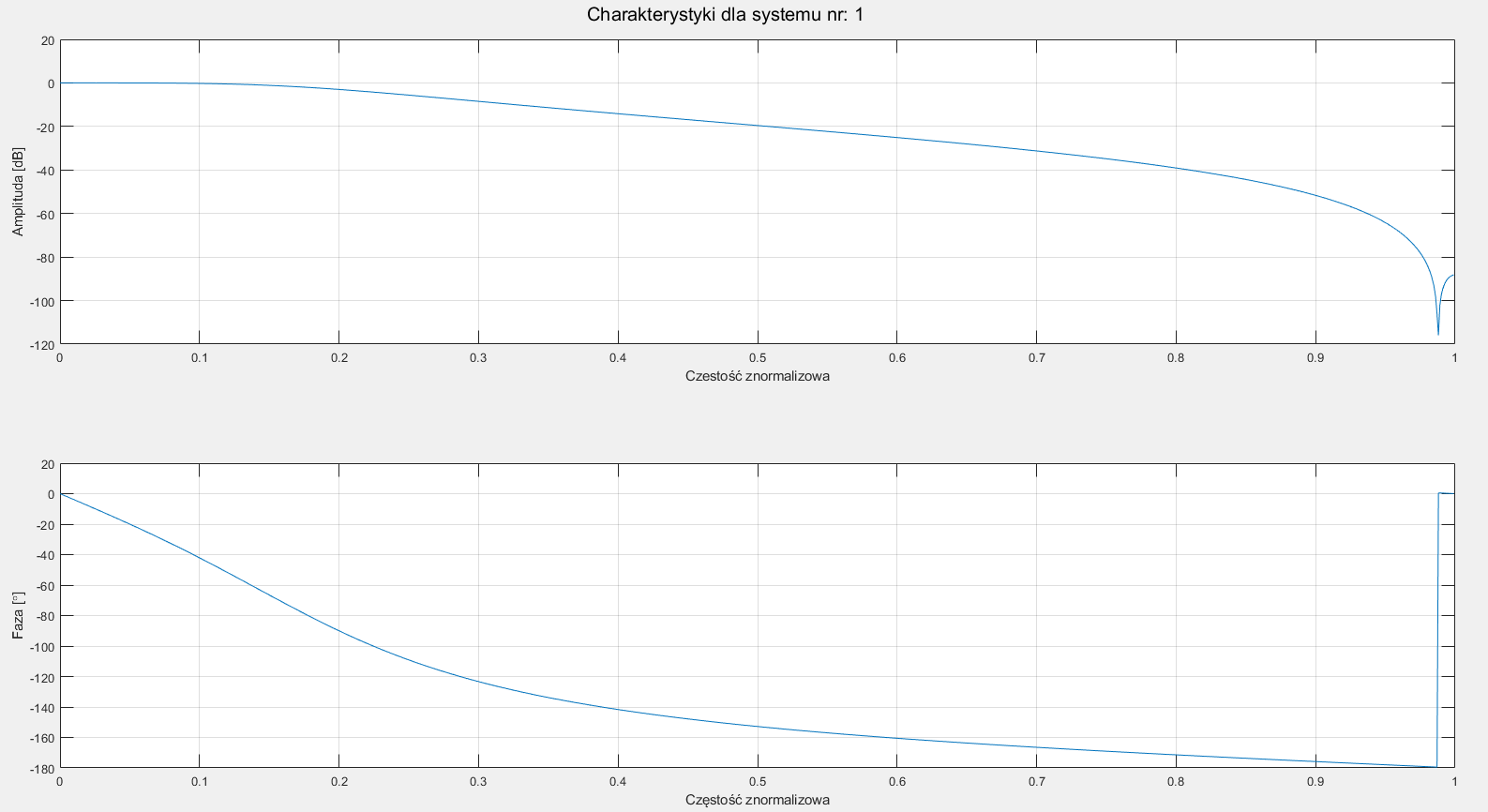
# Laboratorium 6

W ramach tego ćwiczenia laboratoryjnego wykreślane 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”.

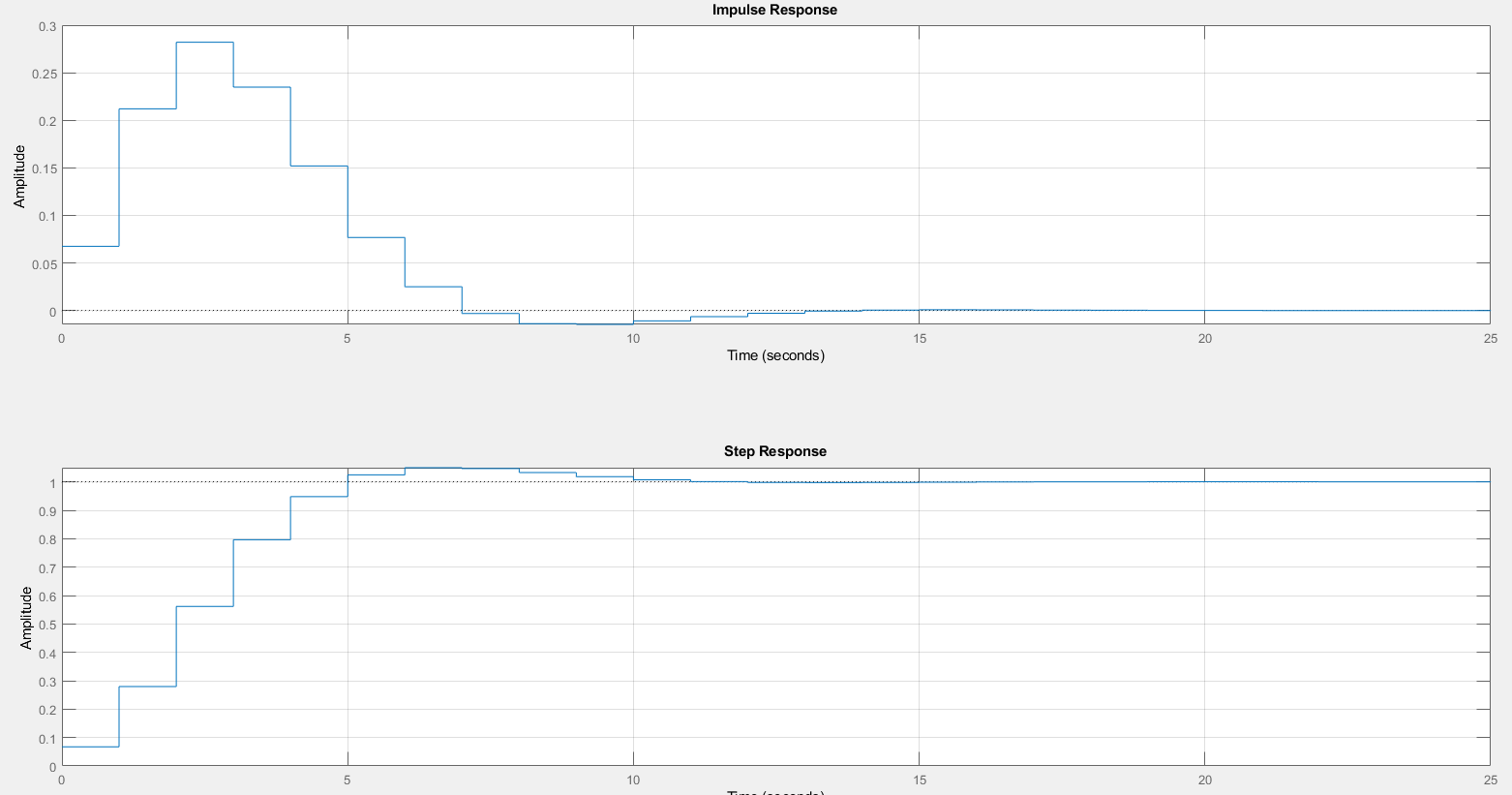
|  |
| --- |
| % Lab 6. Dla zadanych transmitancji wykreślić ich  % odpowiedzi impulsowe, skokowe, charakterystyki  % amplitudowe i fazowe oraz zbadac stabilność.  % Mateusz Krupnik    % CZESC 1: Wykreślenie charakterystyk dla 4 zestawów współczynnikow  % transmitacji układów (KODY 1-4 w Lab. 6)  clc; 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('Czestość znormalizowa');  subplot(212); plot(w, F);  ylabel('Faza [\circ]'); grid;  xlabel('Częstość znormalizowa');  figure(4\*i-2)  subplot(211);  dimpulse(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 biegunow  [z, p, k] = tf2zpk(l, m);  end |

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

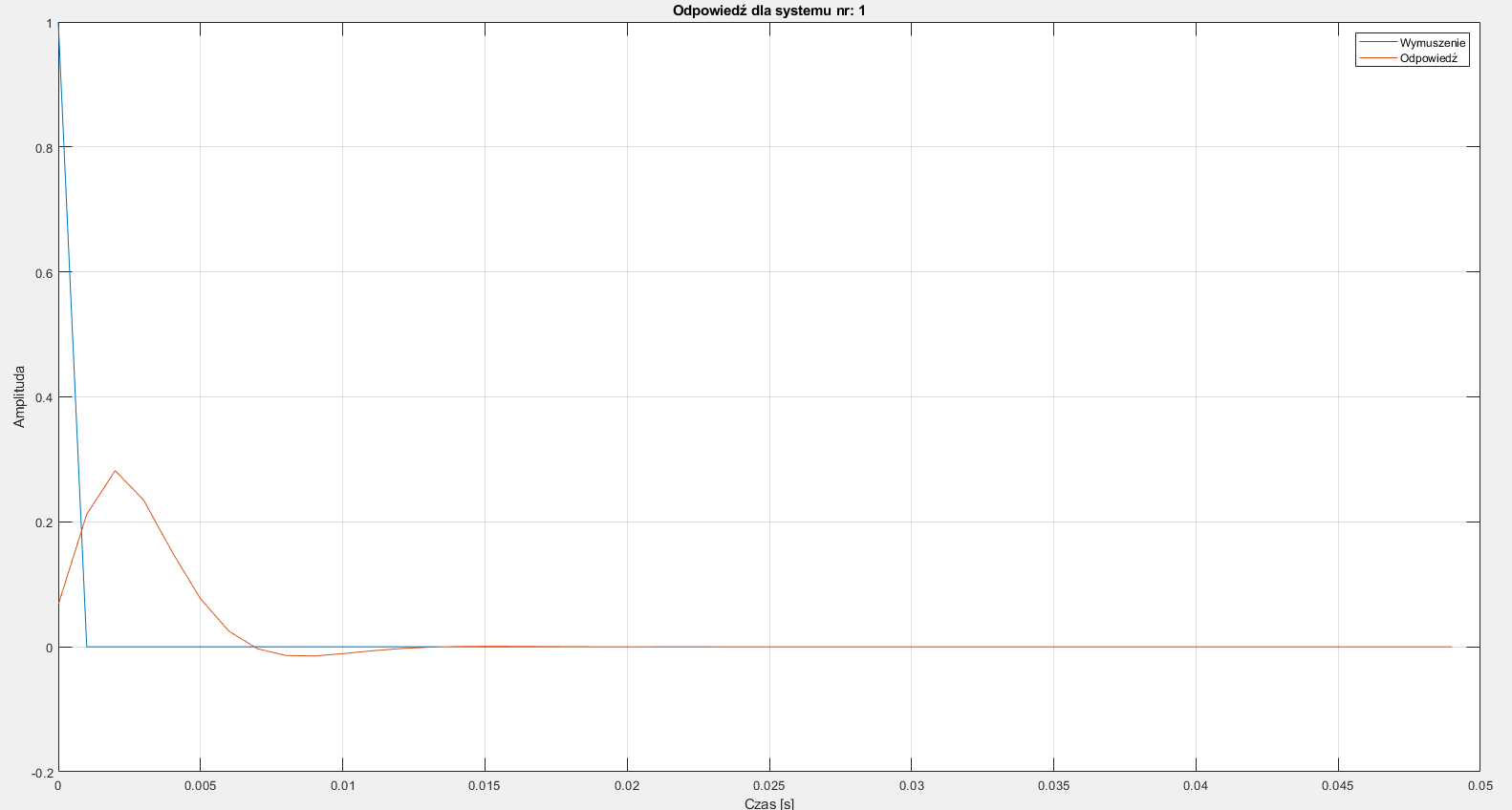


Rys. 6.1. Charakterystyki amplitudowo-częstotliwościowe i fazowo-częstotliwościowe dla układu nr 1.

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.



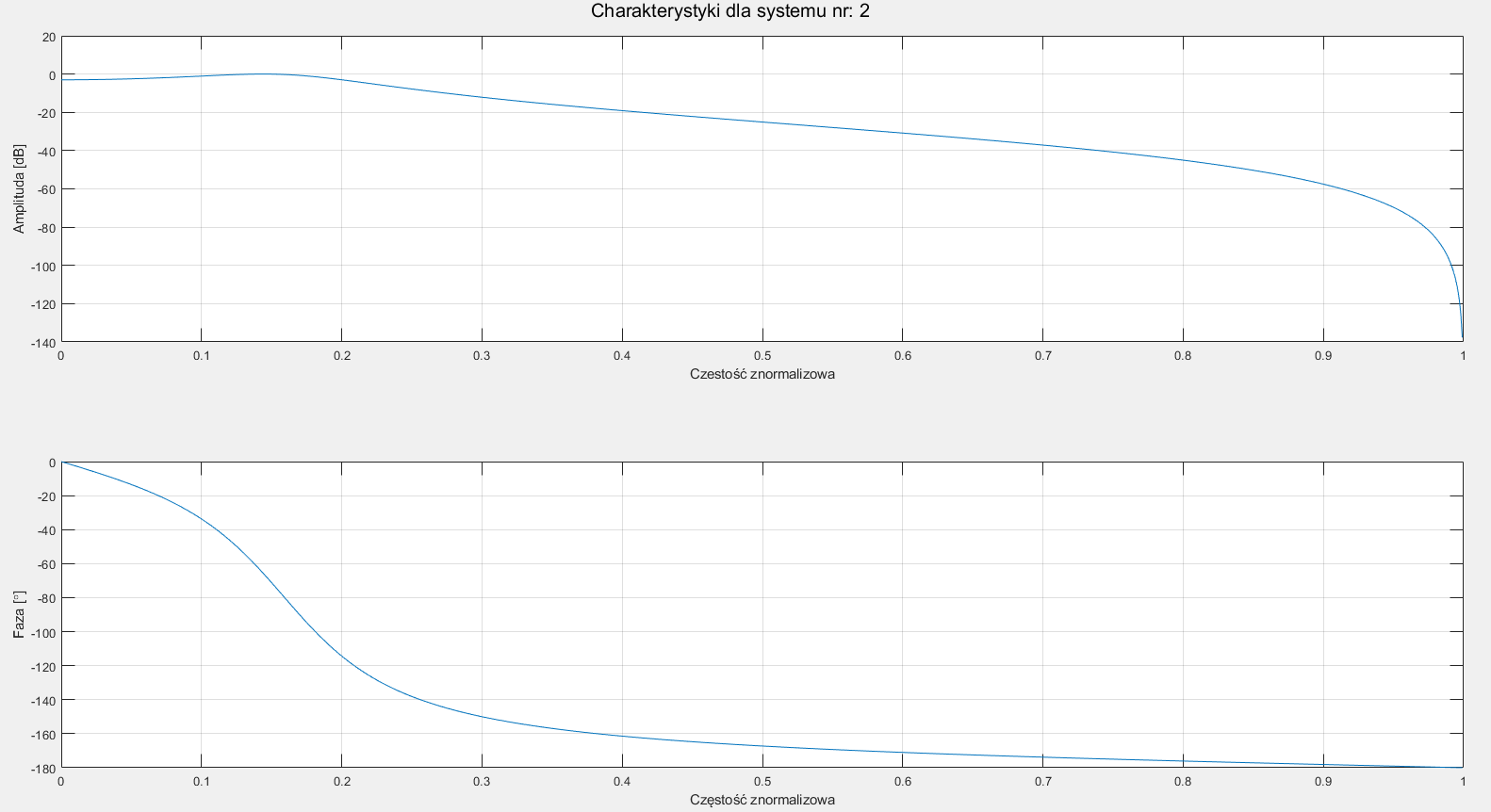
Rys. 6.3. Odpowiedź układu nr 1 na pobudzenie deltą Kroneckera.

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 przestawia rozkład zer i biegunów na płaszczyźnie zespolonej po transformacji bilinowej. Jak widać zera mianownika leżą na okręgu jednostkowym, a więc układ jest stabilny.

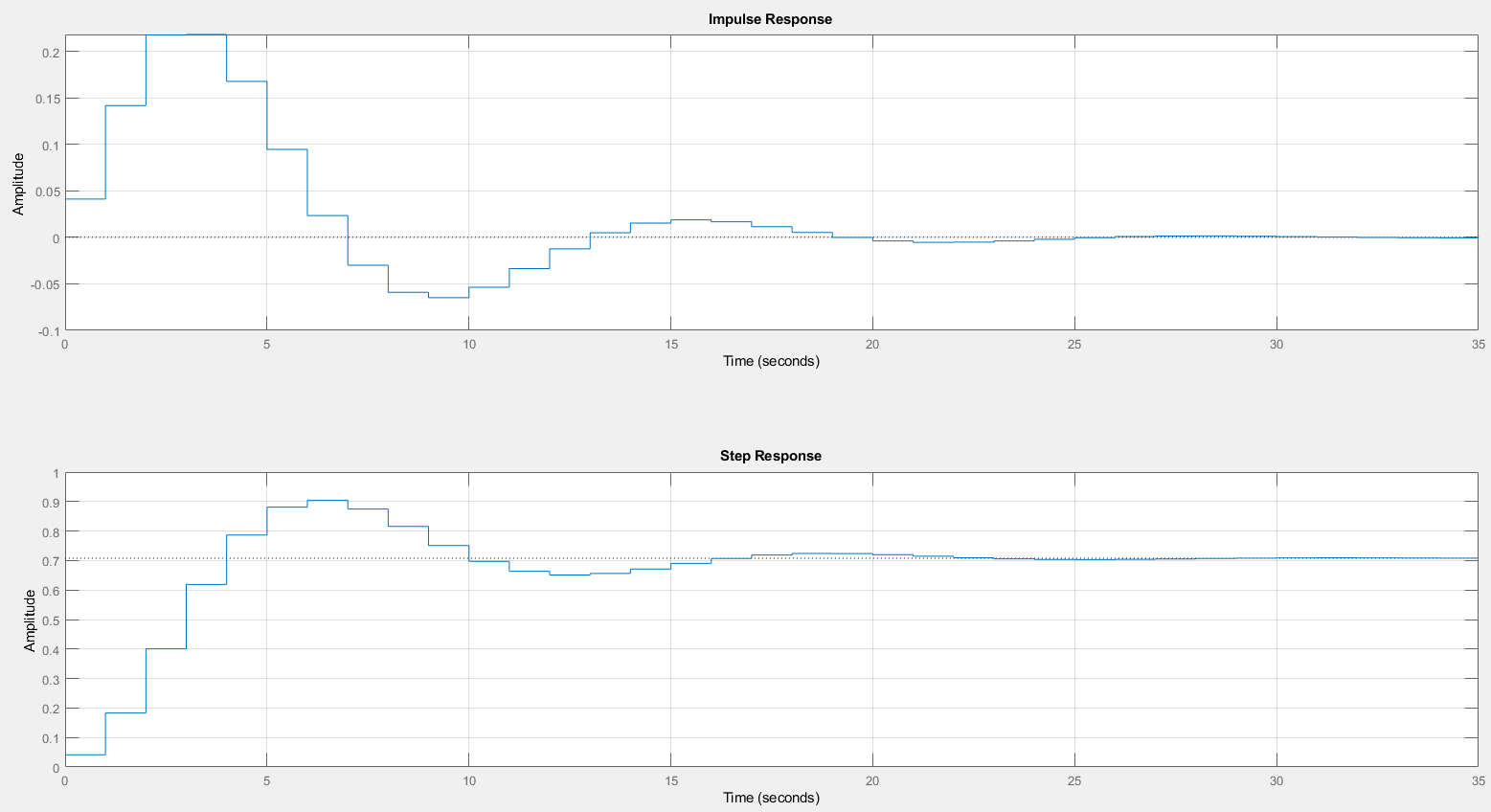


Rys. 6.4. Wykres położenia zer (x) i biegunów (o) układu nr 1.

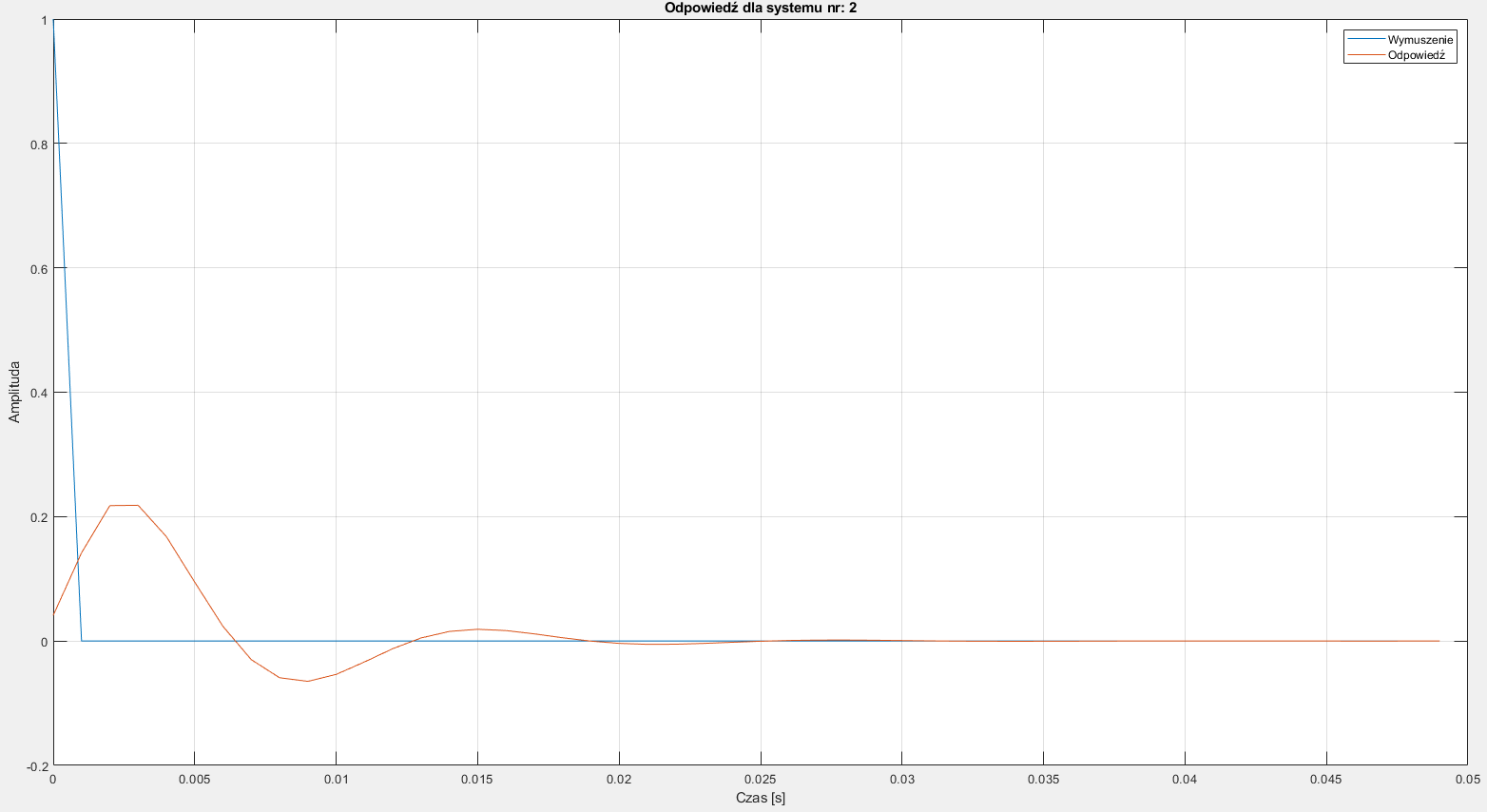
Transmitancja drugiego układu:



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

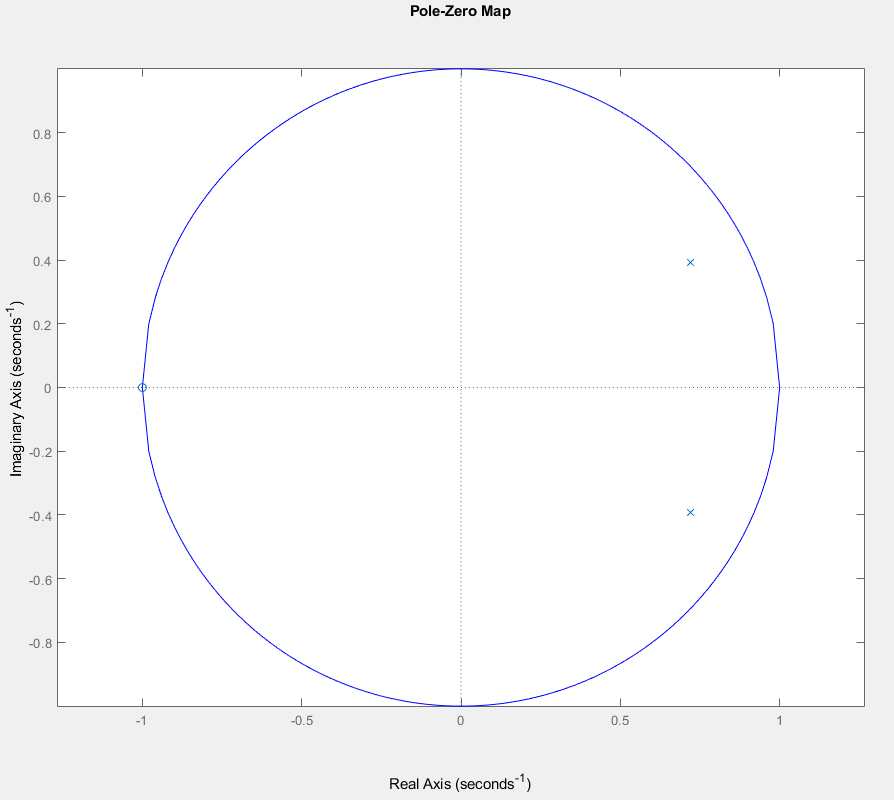


Rys. 6.6. Odpowiedź impulsowa i skokowa układu nr 2.



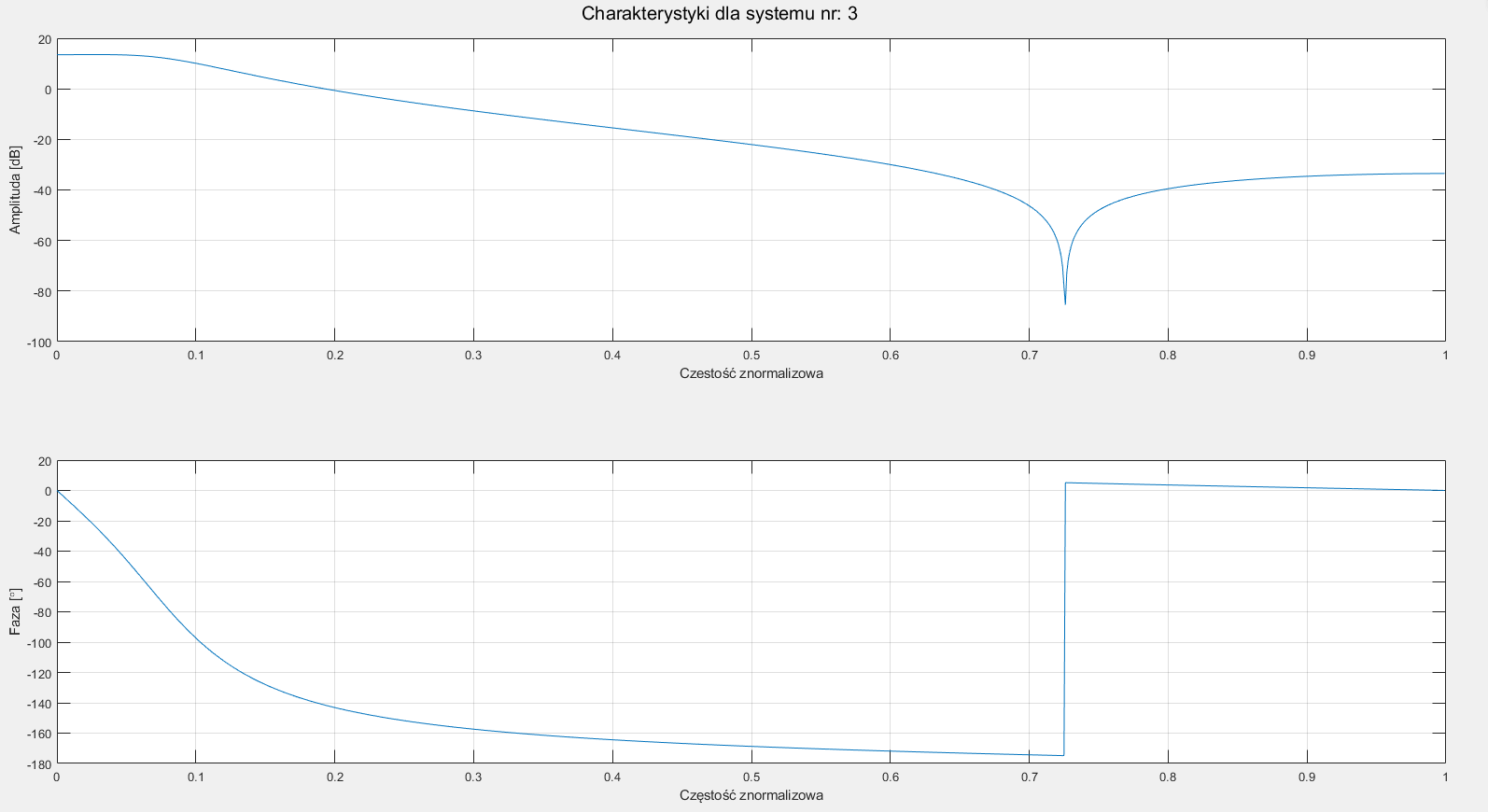
Rys. 6.7. Odpowiedź układu nr 2 na pobudzenie deltą Kroneckera.

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 górka). Z kolei na charakterystyce amplitudowej widoczne jest wzmocnienie składowych o częstości znormalizowanej około 0,15.

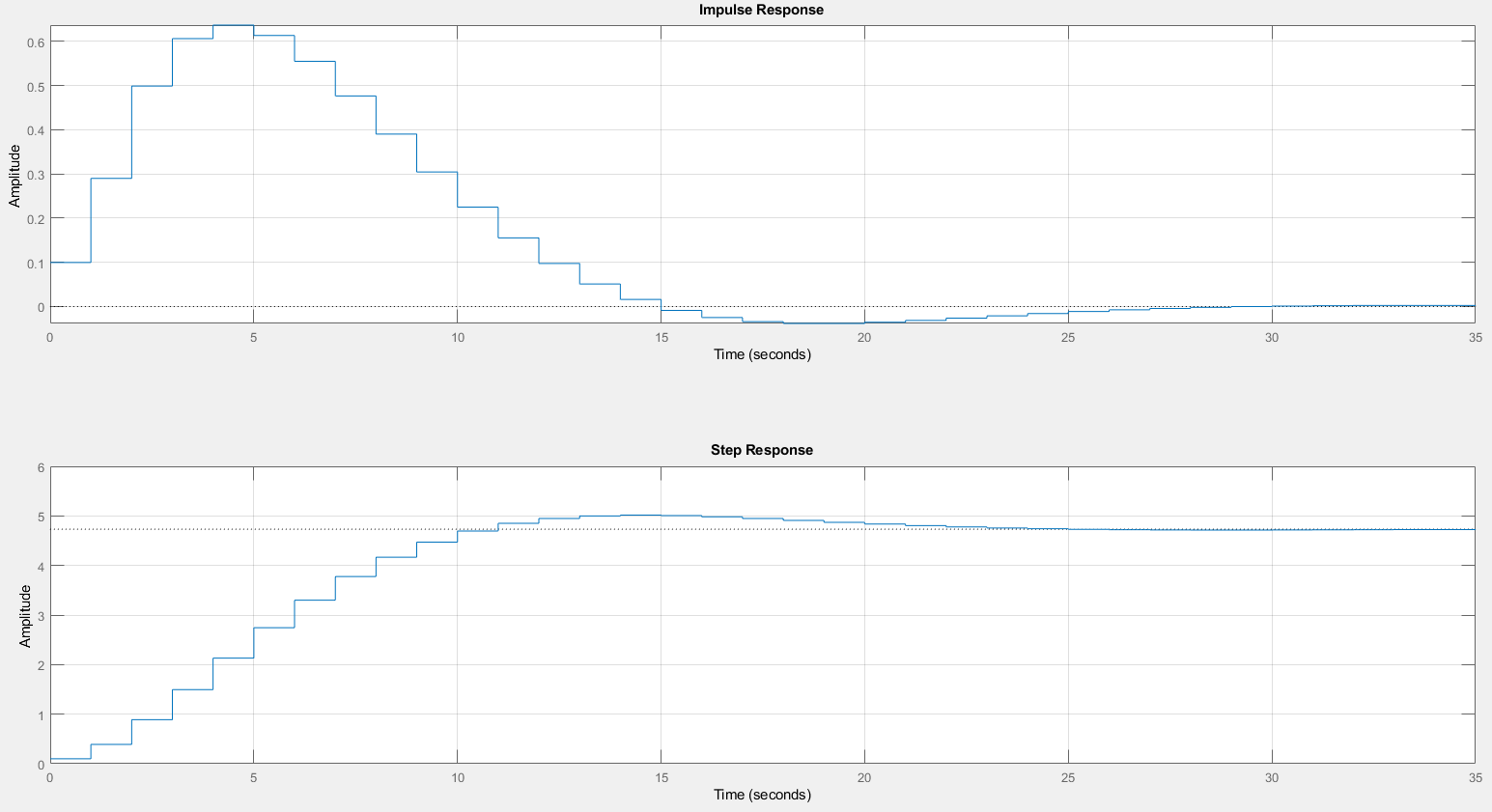


Rys. 6.8. Wykres położenia zer (x) i biegunów (o) układu nr 2.

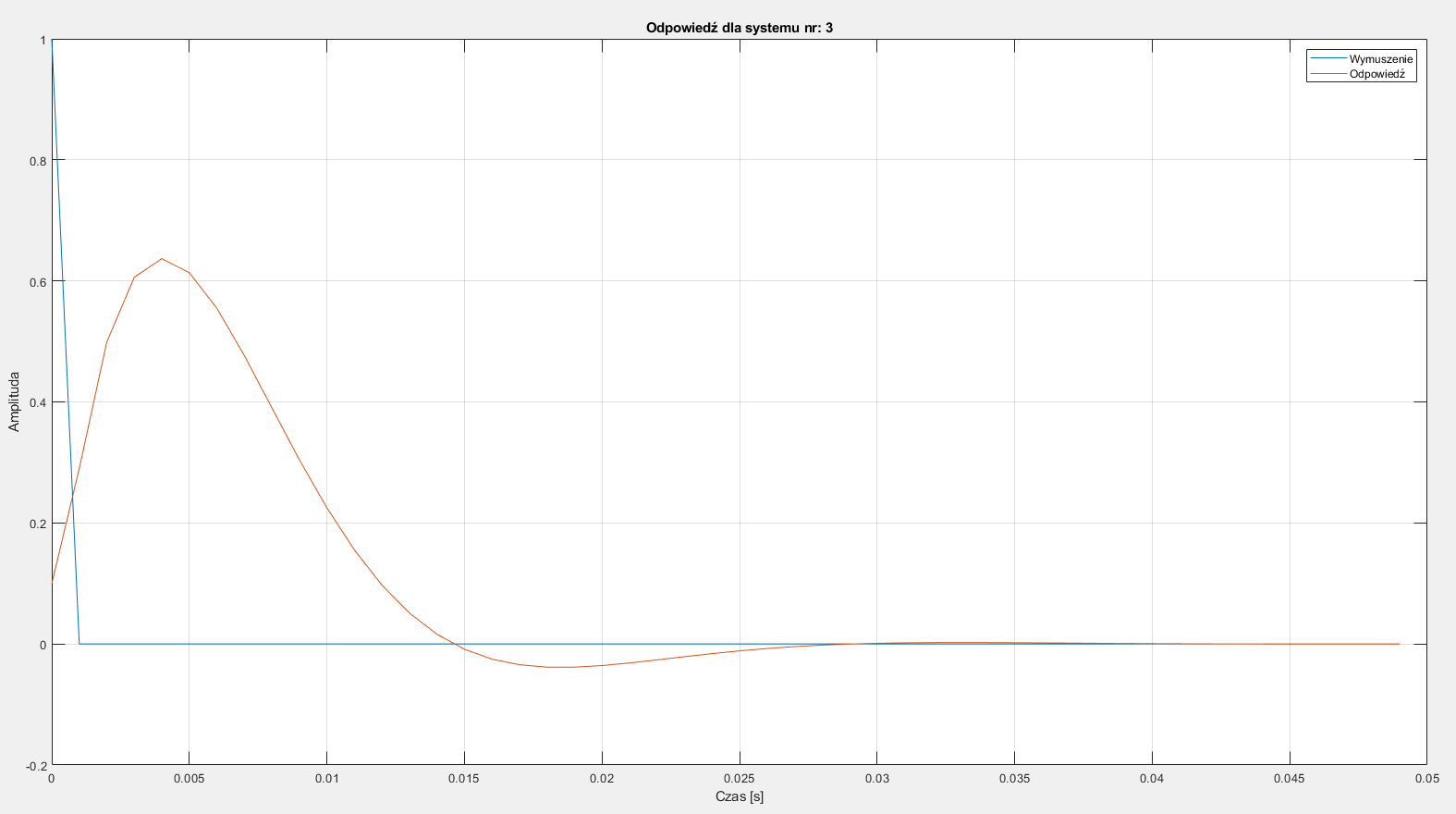
Transmitancja trzeciego układu:



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

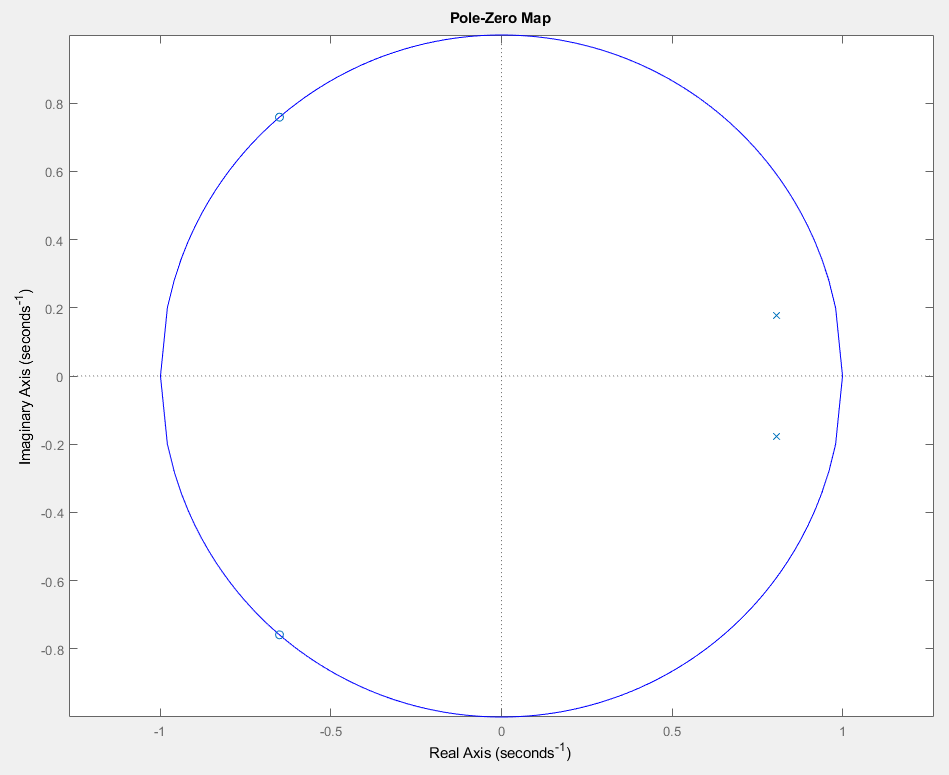


Rys. 6.10. Odpowiedź impulsowa i skokowa układu nr 3.



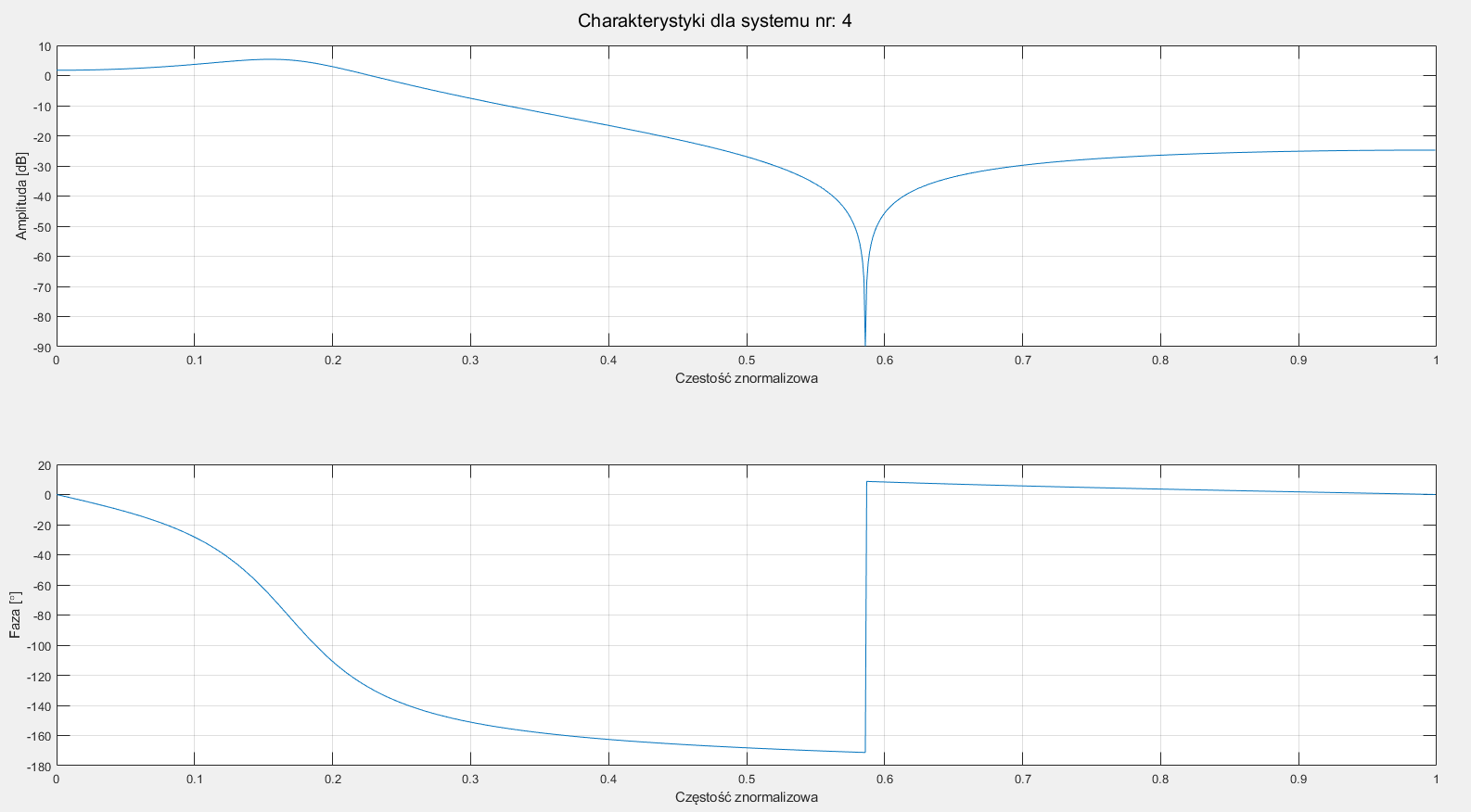
Rys. 6.11. Odpowiedź układu nr 3 na pobudzenie deltą Kroneckera.

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

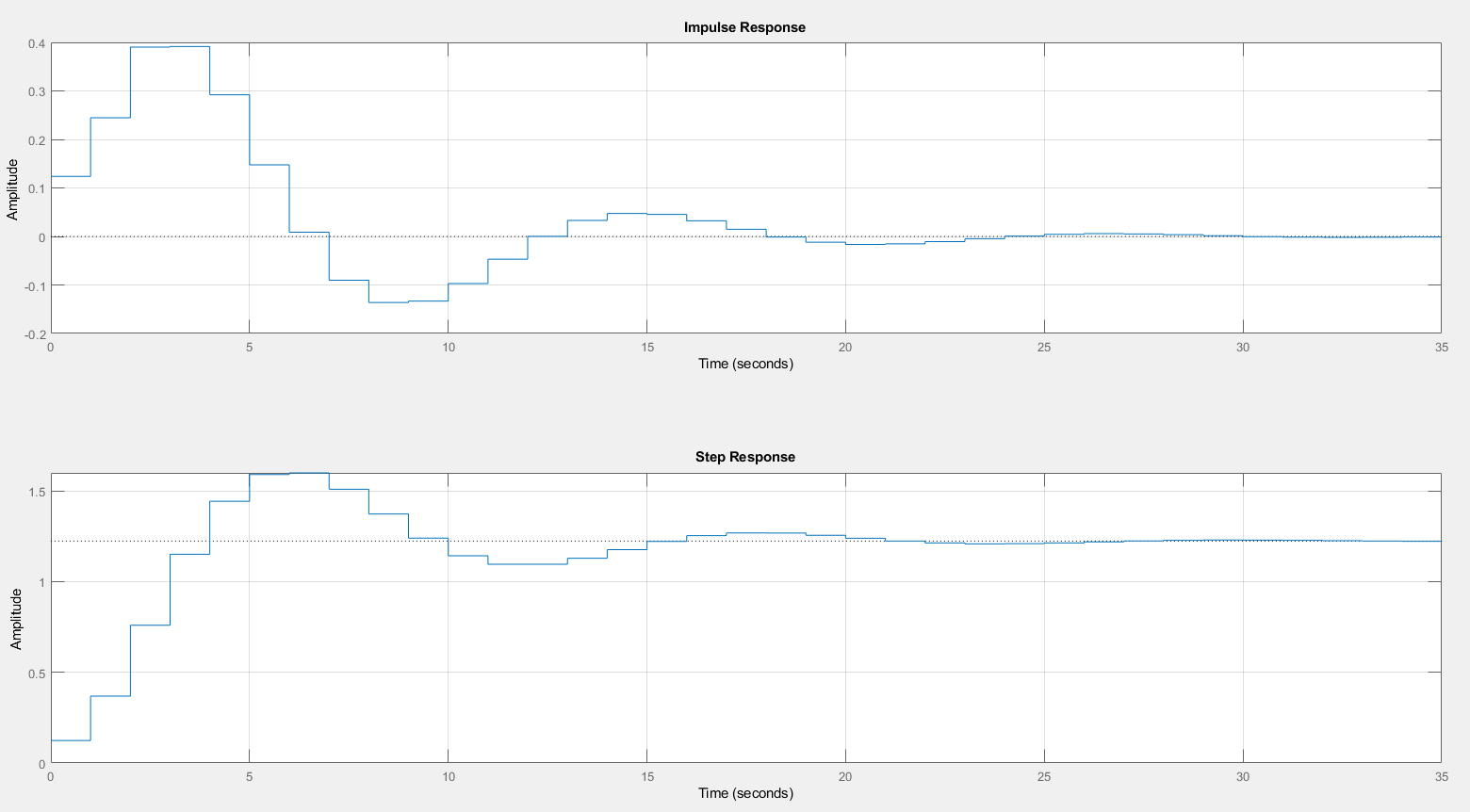


Rys. 6.12. Wykres położenia zer (x) i biegunów (o) układu nr 3.

Transmitancja ostatniego układu:



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

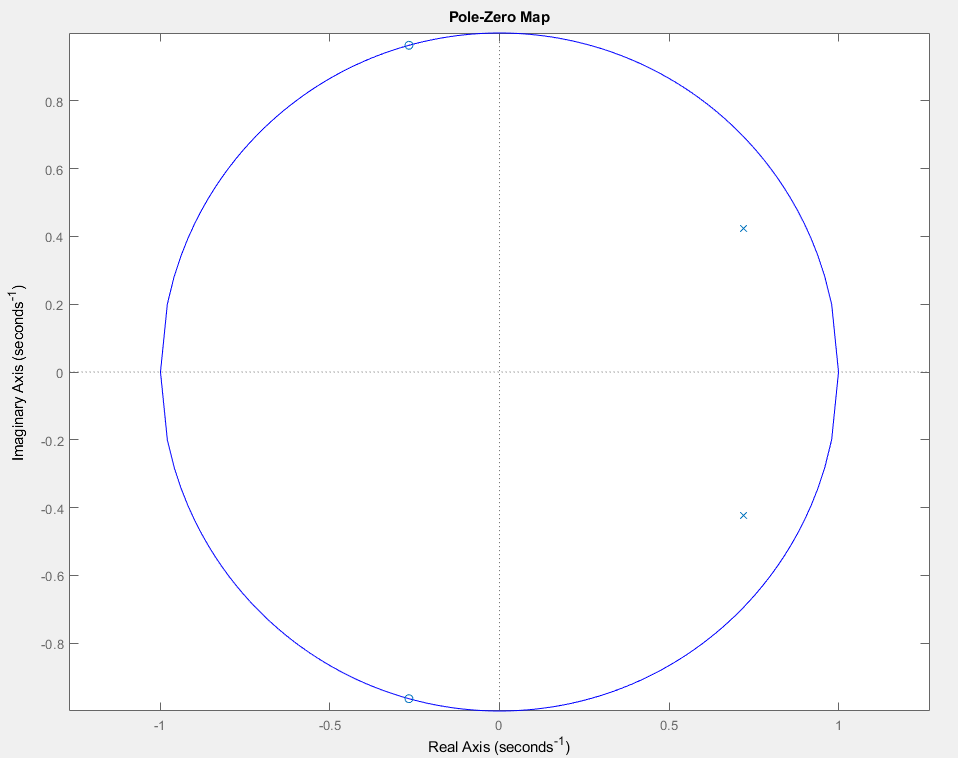


Rys. 6.14. Odpowiedź impulsowa i skokowa układu nr 4.



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.

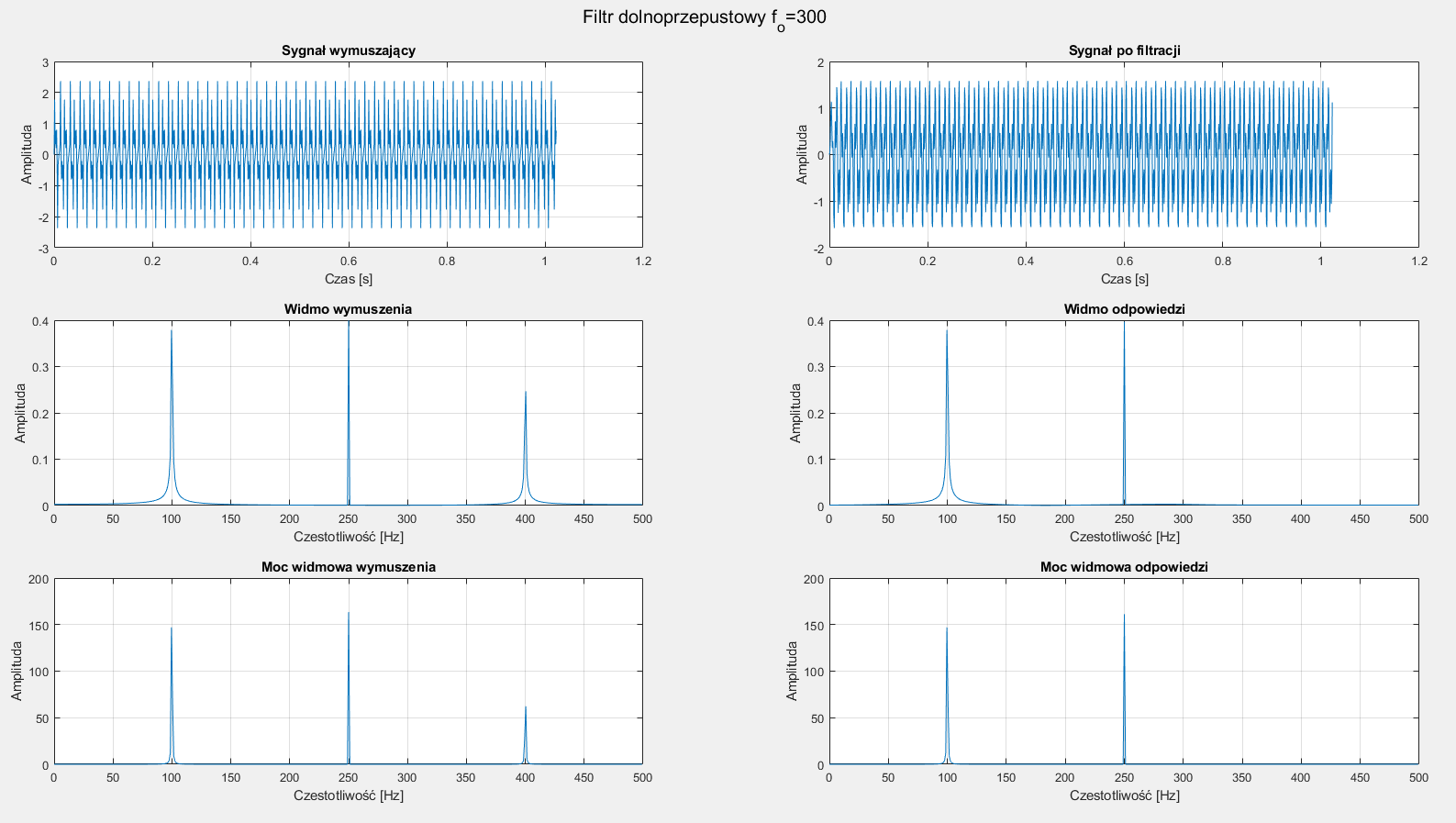


Rys. 6.16. Wykres położenia zer (x) i biegunów (o) układu nr 4.

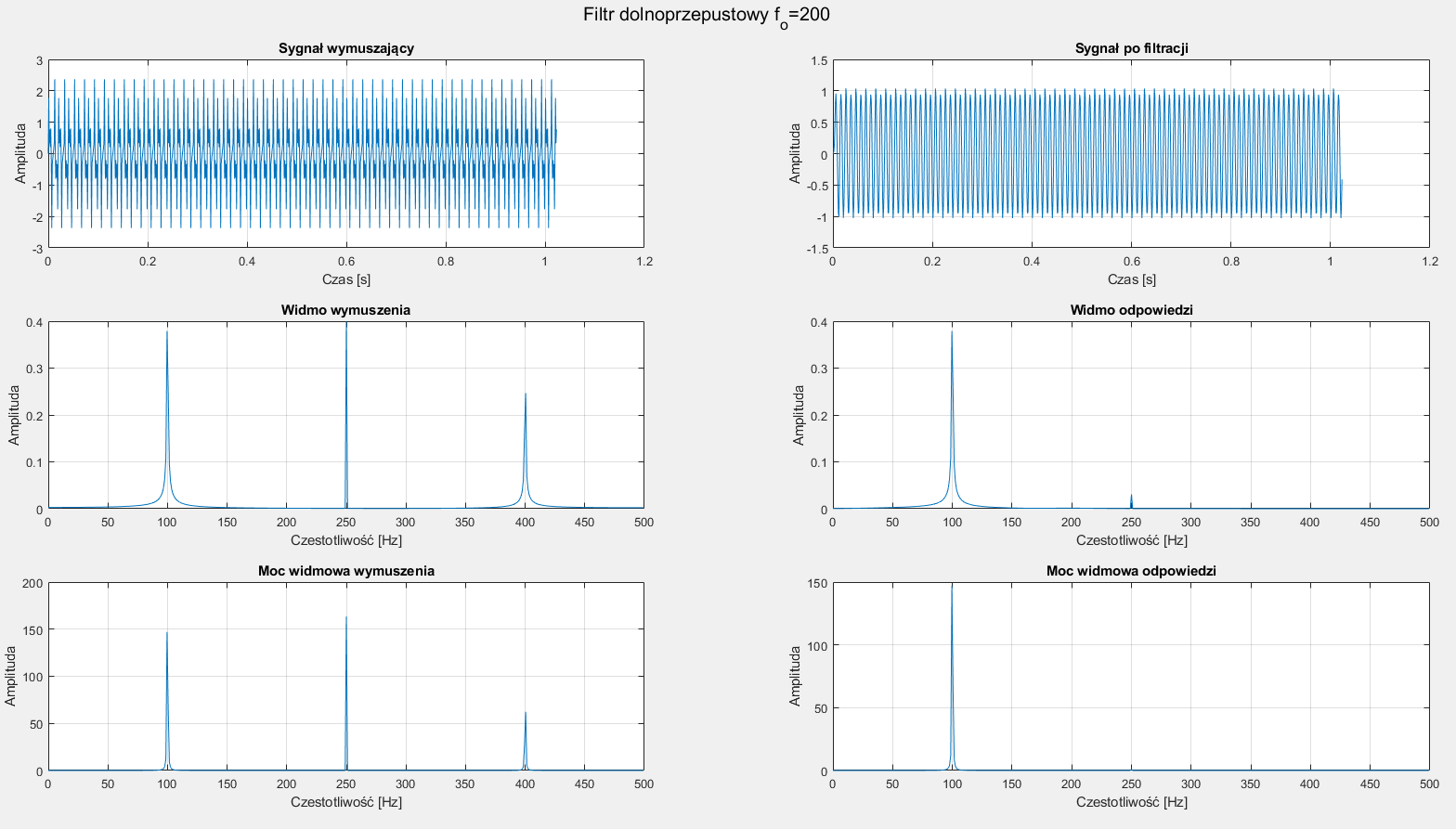
Następnie dla sygnału o składającego się z 3 sygnałów sinusoidalnych o różnych częstościach dokonano filtracji filtrem dolnoprzepustowym oraz środkowo przepustowym. W tym celu wykorzystano funkcję butter(rząd, częstość\_znomalizowa) do zaprojektowania filtra Butterwortha o zadanym rzędzie i znormalizowanej częstości 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\_Mateusz\_Lab\_6\_2.m".

|  |
| --- |
| % Lab 6.  % Dla filtra FIR sprawdzic jego działanie.  % Mateusz Krupnik  clear all; close all; clc;  %% Sprawdzenie działania filtrów cyfrowych  % Dane sygnałów  f1=100; f2=250; f3=400; fs=1000; % Czestotliwosci skladowych i Nyquista  A1=1; A2=0.8; A3=0.65; % Aplitudy skladowych  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; % Czestotliwosci ociecia  wn1 = fo1\*2/fs; wn2 = fo2\*2/fs; % Znormalizowane czestotliwosci  wn = [wn1, wn2]; fo = [fo1, fo2];  % Filtr Butter - dolnoprzepustowy  N = [2, 4 ,8]; rzad = 3; % wybór rzędu filtra    %% Filtr dolnoprzepustowy Butterwotha    for i=1:length(wn)  [l, m] = butter(N(1,rzad), wn(i)); % Rząd oraz czest. odciecia  y = filter(l, m, x); % Odpowiedz filtra  % Obliczenie transformaty Fouriera  [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(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('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Widmo wymuszenia');  subplot(324); plot(f\_w, Wid(2,:));  xlabel('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Widmo odpowiedzi');  subplot(325); plot(f\_w, Moc(1,:));  xlabel('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Moc widmowa wymuszenia');  subplot(326); plot(f\_w, Moc(2,:));  xlabel('Czestotliwość [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 sepctrum, 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.

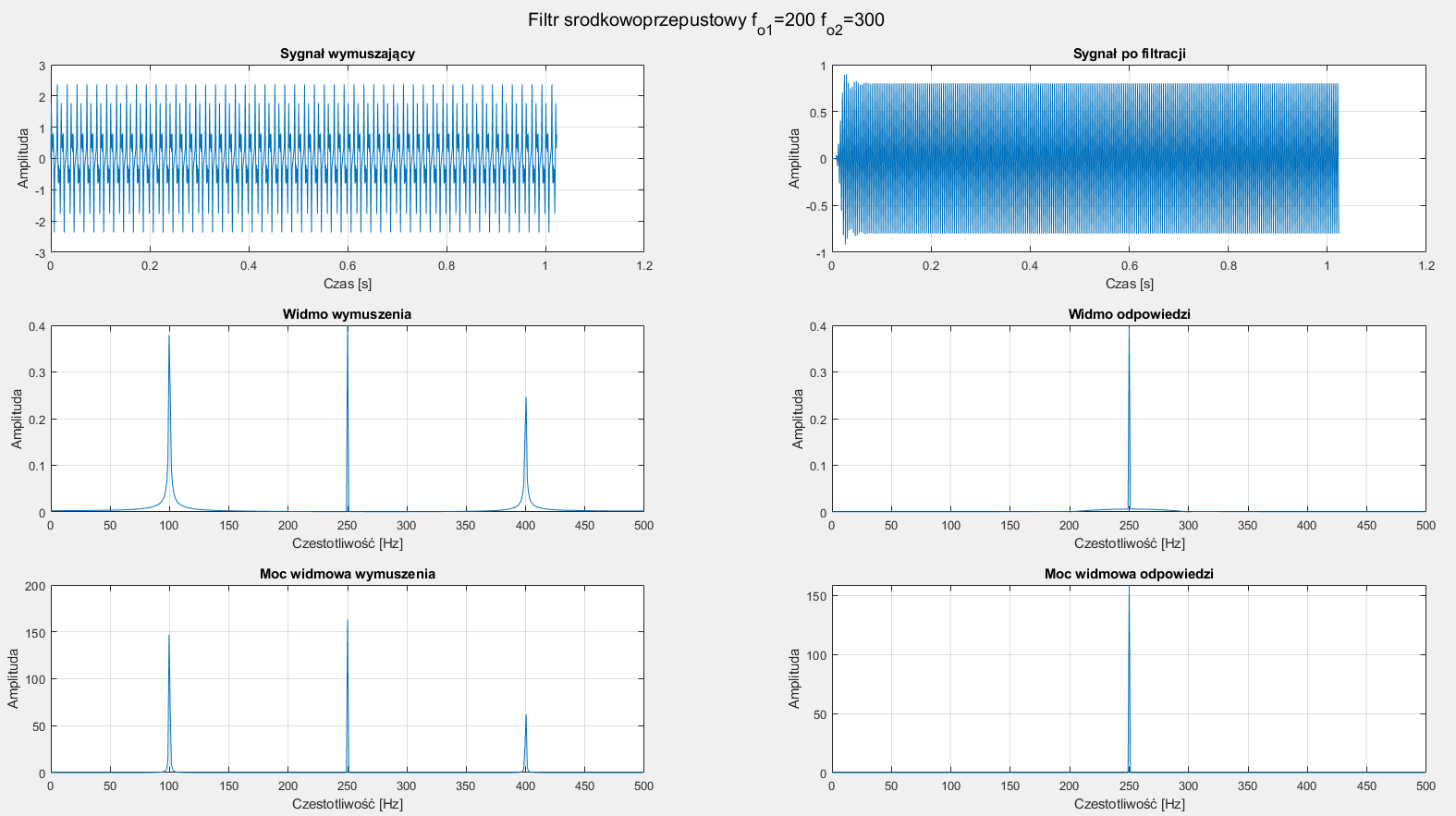


Rys. 6.18. Filtracja filtrem dolnoprzepustowym Butterwortha o częstotliwości odcięcia 200 Hz i rzędzie równym 8.

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 należałby zwiększyć rząd filtra lub przesunąć granice odcięcia. Następnie przetestowana została filtracja środkowoprzepustowa, kod programu i wynik znajdują się poniżej.

Tab. 6.3. Kod programu "Krupnik\_Mateusz\_Lab\_6\_3.m".

|  |
| --- |
| % Lab 6.  % Dla filtra FIR sprawdzic jego działanie.  % Mateusz Krupnik  clear all; close all; clc;  %% Sprawdzenie działania filtrów cyfrowych  % Dane sygnałów  f1=100; f2=250; f3=400; fs=1000; % Czestotliwosci skladowych i Nyquista  A1=1; A2=0.8; A3=0.65; % Aplitudy skladowych  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; % Czestotliwosci ociecia  wn1 = fo1\*2/fs; wn2 = fo2\*2/fs; % Znormalizowane czestotliwosci  wn = [wn1, wn2]; fo = [fo1, fo2];  % Filtr Butter - dolnoprzepustowy  N = [2, 4 ,8]; rzad = 3; % wybór rzędu filtra    %% Filtr srodkowoprzepustowy Butterwortha  % Parametry z poprzedniego filtru  [l, m] = butter(N(1,rzad), fliplr(wn)); % Odwrócenie kolejności cz. odc.  y = filter(l, m, x); % Odpowiedz 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('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Widmo wymuszenia');  subplot(324); plot(f\_w, Wid(2,:));  xlabel('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Widmo odpowiedzi');  subplot(325); plot(f\_w, Moc(1,:));  xlabel('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Moc widmowa wymuszenia');  subplot(326); plot(f\_w, Moc(2,:));  xlabel('Czestotliwość [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 sepctrum, 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 |

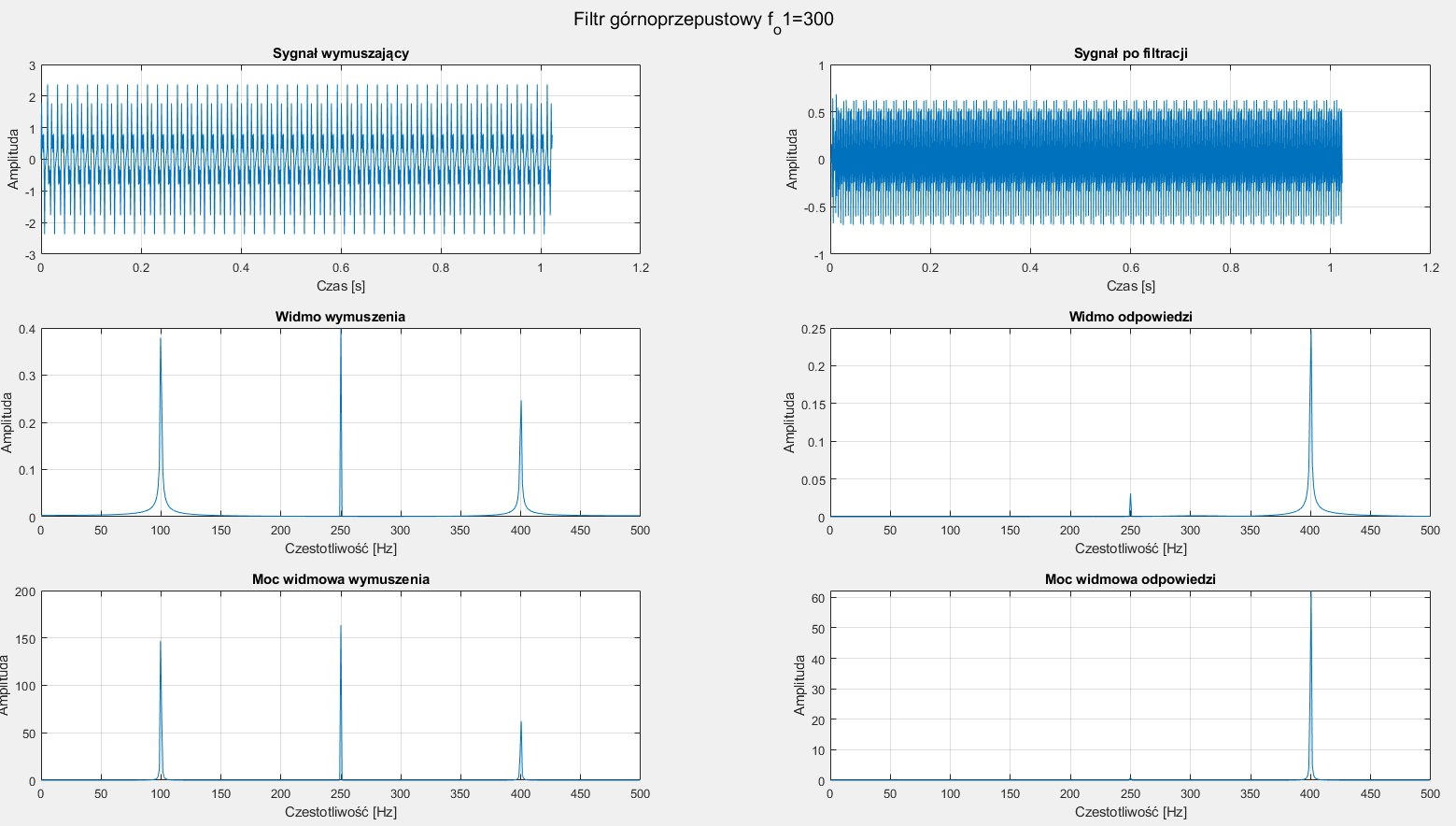


Rys. 6.19. Filtracja filtrem środkowoprzepustowym Butterwortha 8 rzędu o częstotliwościach granicznych 200 i 300 Hz.

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

Tab. 6.4. Kod programu "Krupnik\_Mateusz\_Lab\_6\_4.m".

|  |
| --- |
| % Lab 6.  % Dla filtra FIR sprawdzic jego działanie.  % Mateusz Krupnik  clear all; close all; clc;  %% Sprawdzenie działania filtrów cyfrowych  % Dane sygnałów  f1=100; f2=250; f3=400; fs=1000; % Czestotliwosci skladowych i Nyquista  A1=1; A2=0.8; A3=0.65; % Aplitudy skladowych  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; % Czestotliwosci ociecia  wn1 = fo1\*2/fs; wn2 = fo2\*2/fs; % Znormalizowane czestotliwosci  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); % Odpowiedz 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(20); sgtitle(['Filtr górnoprzepustowy f\_o1=' 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('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Widmo wymuszenia');  subplot(324); plot(f\_w, Wid(2,:));  xlabel('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Widmo odpowiedzi');  subplot(325); plot(f\_w, Moc(1,:));  xlabel('Czestotliwość [Hz]'); ylabel('Amplituda'); grid;  title('Moc widmowa wymuszenia');  subplot(326); plot(f\_w, Moc(2,:));  xlabel('Czestotliwość [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 sepctrum, 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 |



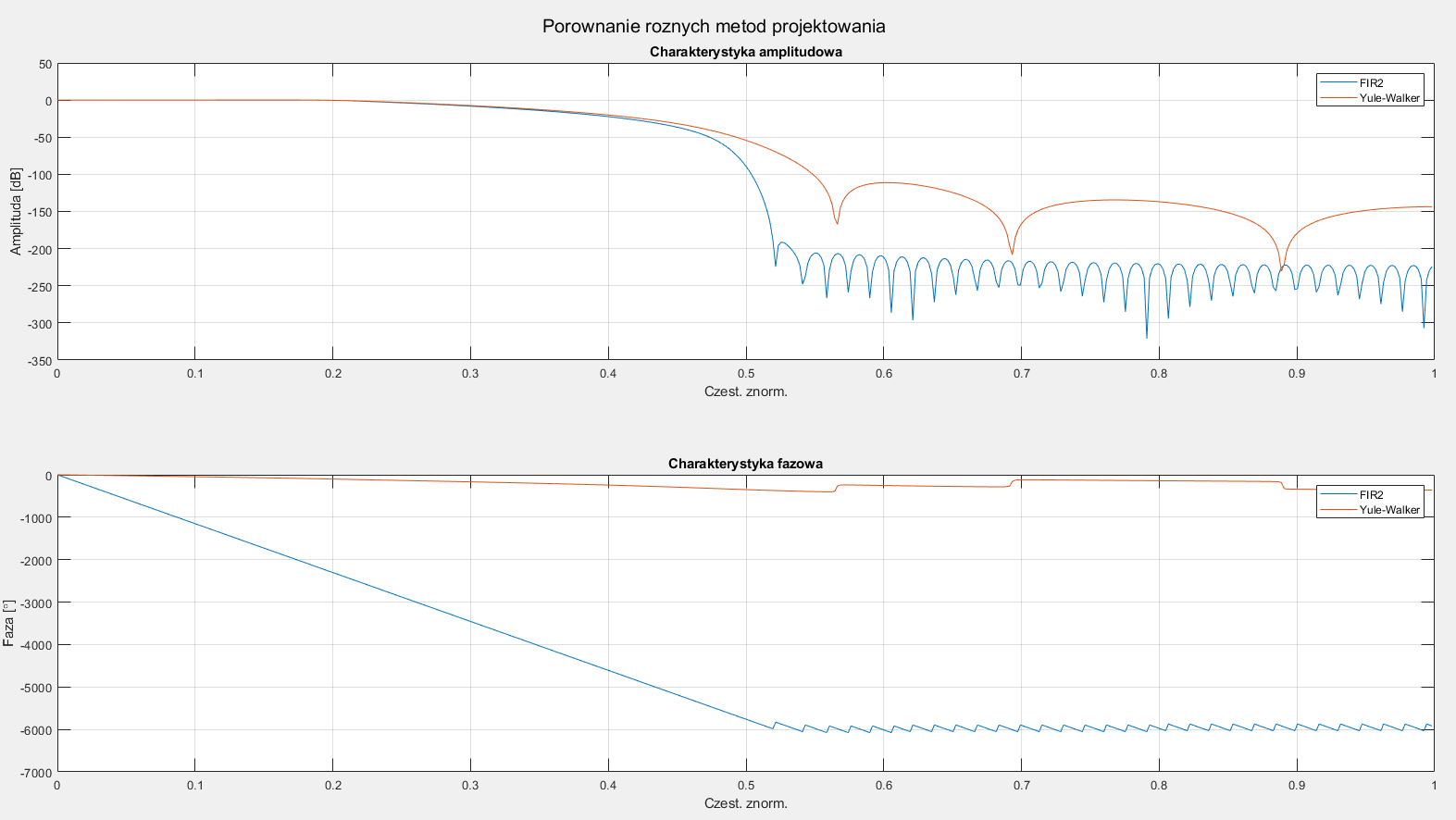
Rys. 6.20. Filtracja filtrem górnoprzepustowym Butterwortha 8 rzędu o częstotliwości odcięcia 300 Hz.

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

Kolejno dokonane zostało porównanie metod projektowania filtrów cyfrowych. Porównano metodę Yula-Walkera polegająca 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\_Mateusz\_Lab\_6\_5.m".

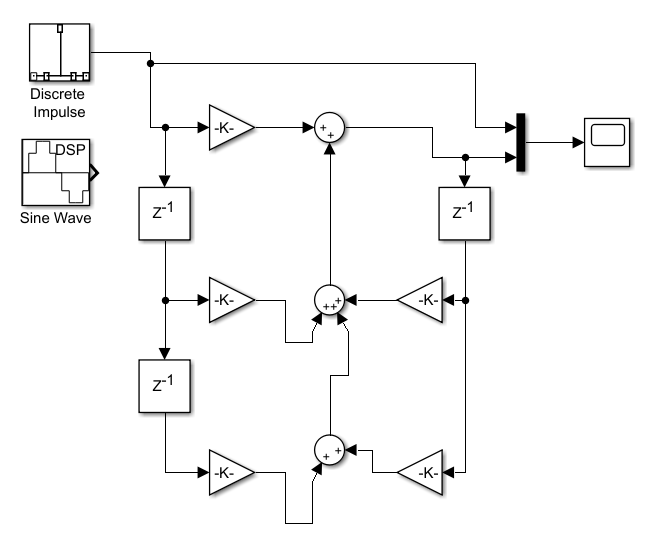
|  |
| --- |
| % Lab 6.  % Dla filtra FIR sprawdzic jego działanie.  % Mateusz Krupnik  clear all; close all; clc;  %% Sprawdzenie działania filtrów cyfrowych  % Dane sygnałów  f1=100; f2=250; f3=400; fs=1000; % Czestotliwosci skladowych i Nyquista  A1=1; A2=0.8; A3=0.65; % Aplitudy skladowych  t=0:(1/fs):1.023; % 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; % Czestotliwosci ociecia  wn1 = fo1\*2/fs; wn2 = fo2\*2/fs; % Znormalizowane czestotliwosci  wn = [wn1, wn2]; fo = [fo1, fo2];  % Filtr Butter - dolnoprzepustowy  N = [2, 4 ,8]; rzad = 3; % wybór rzędu filtra    %% Projekt filtra cyfrowego i jego dzialanie  % Parametry, niektóre z poprzednich filtrów  format long e; % Typ formatowania zmiennych, 16 miejsc, wykladniczo  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,rzad), F\_, M\_); % Metoda Yule-Walkera  b = fir2(128, F\_, M\_); fir2(128, F\_, M\_); % Metoda próbkowania w d. czest.      [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('Porownanie roznych 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.

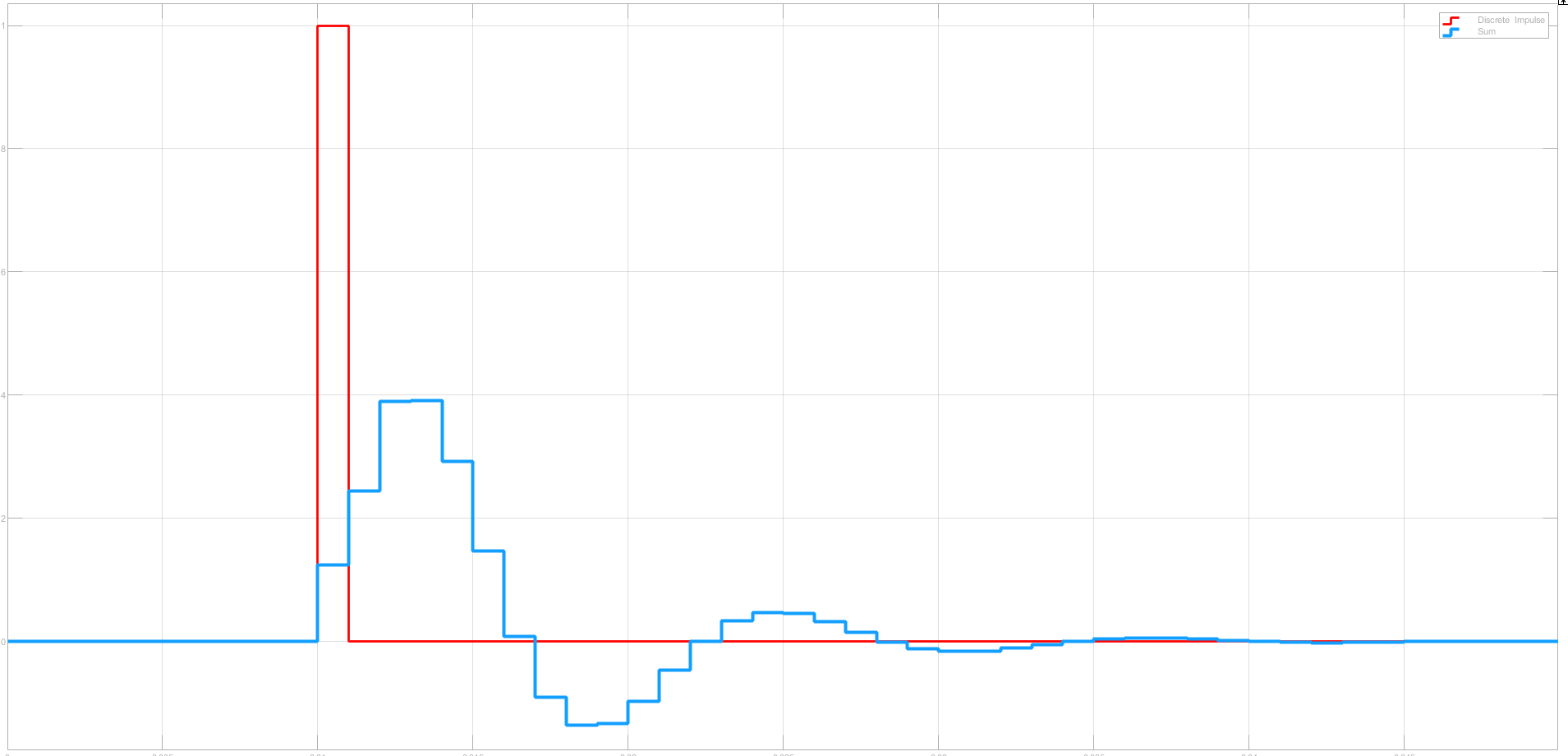
Jak można zauważyć filtry te posiadają o znaczące różnice w budowie. Filtr uzyskany metodą próbkowania w dziedzinie częstotliwości posiada o wiele większy rozmiar, ale jego faza jest liniowa w paśmie przepustowym. W paśmie zaporowym wykazuje się pulsacjami o stałym poziomie wysokości listków. Filtr Yule-Walkera 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.

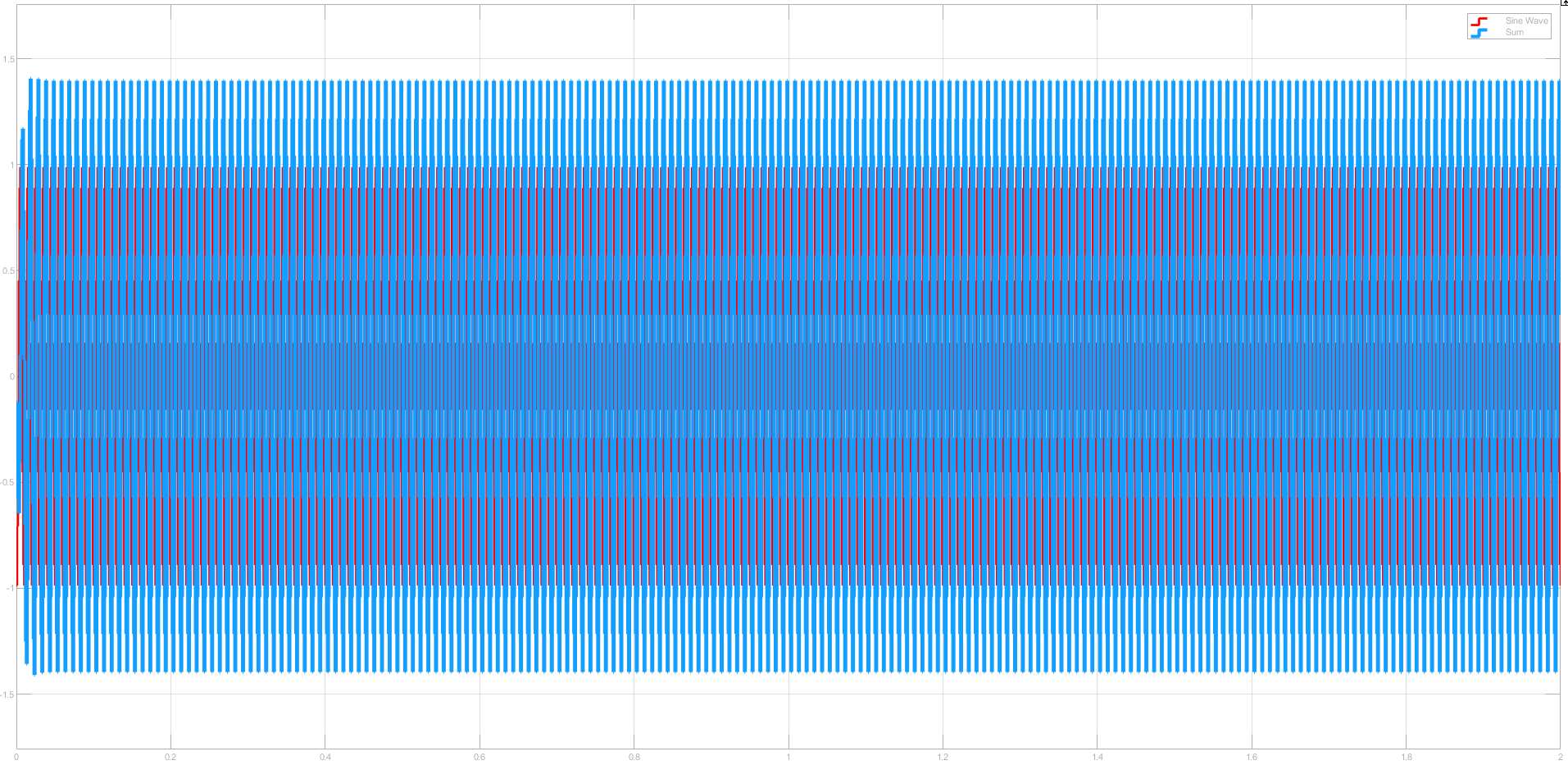


Rys. 6.22. Schemat blokowy filtra IIR. Plik „Krupnik\_Mateusz\_Lab\_6\_6\_SIMULINK.slx”.

Następnie wyznaczono odpowiedź impulsową i filtrację cyfrowego sygnału sinusoidalnego.

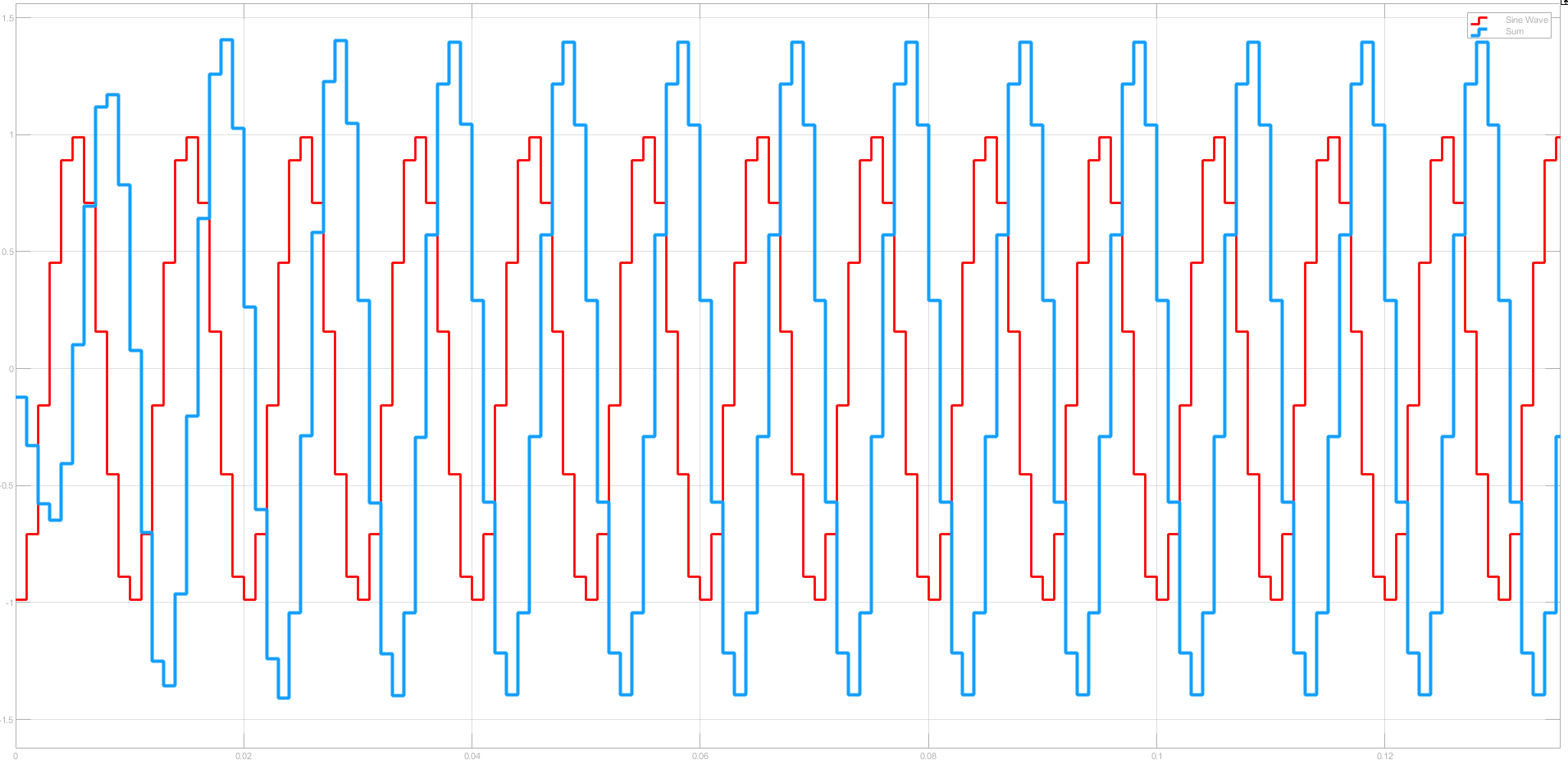


Rys. 6.23. Odpowiedź na impuls dyskretny.



Rys. 6.24. Filtracja sygnału sinusoidalnego.

Jak widać wyniki filtra 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.



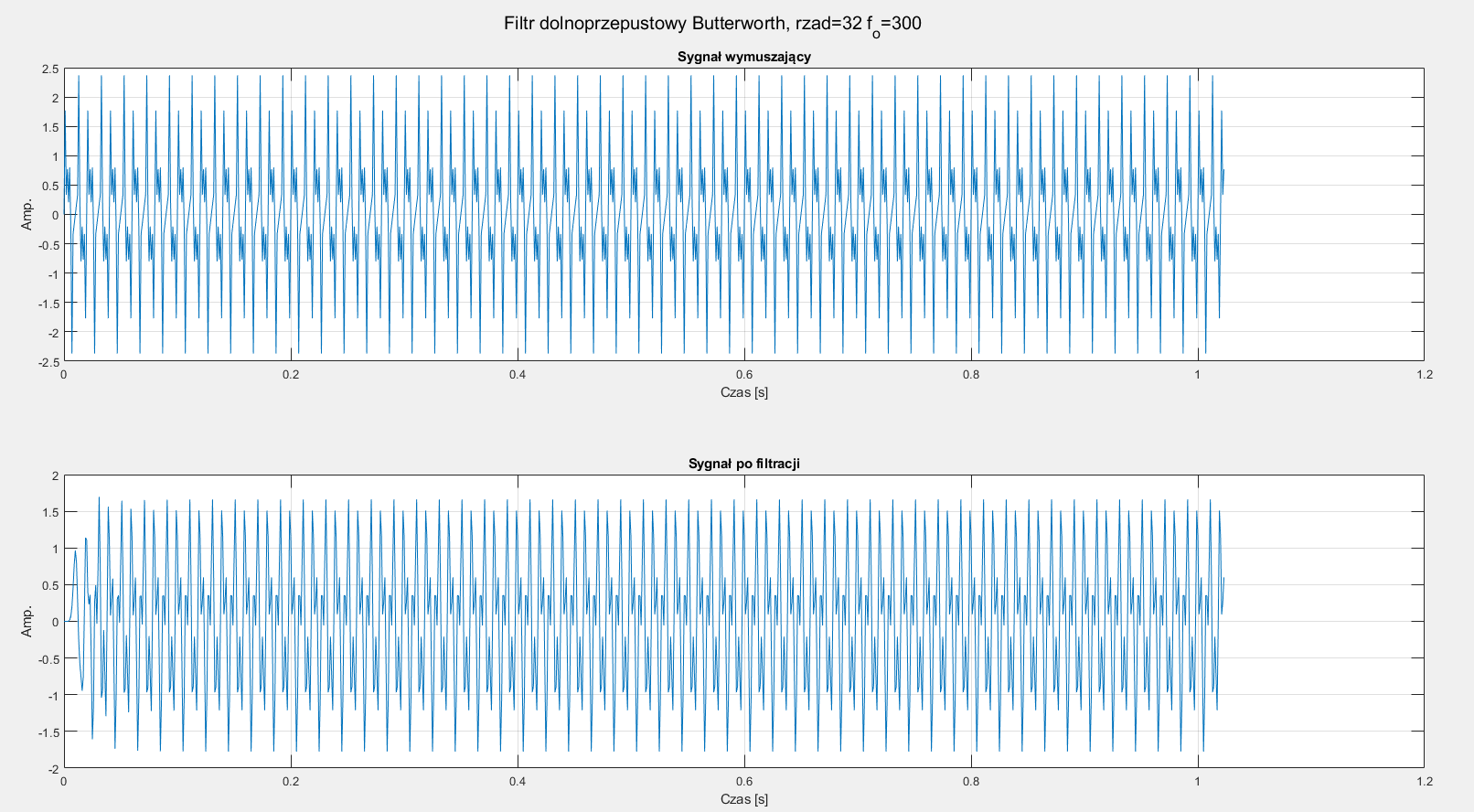
Rys. 6.25. Przybliżenie na początkowy fragment filtracji.

# Laboratorium 7

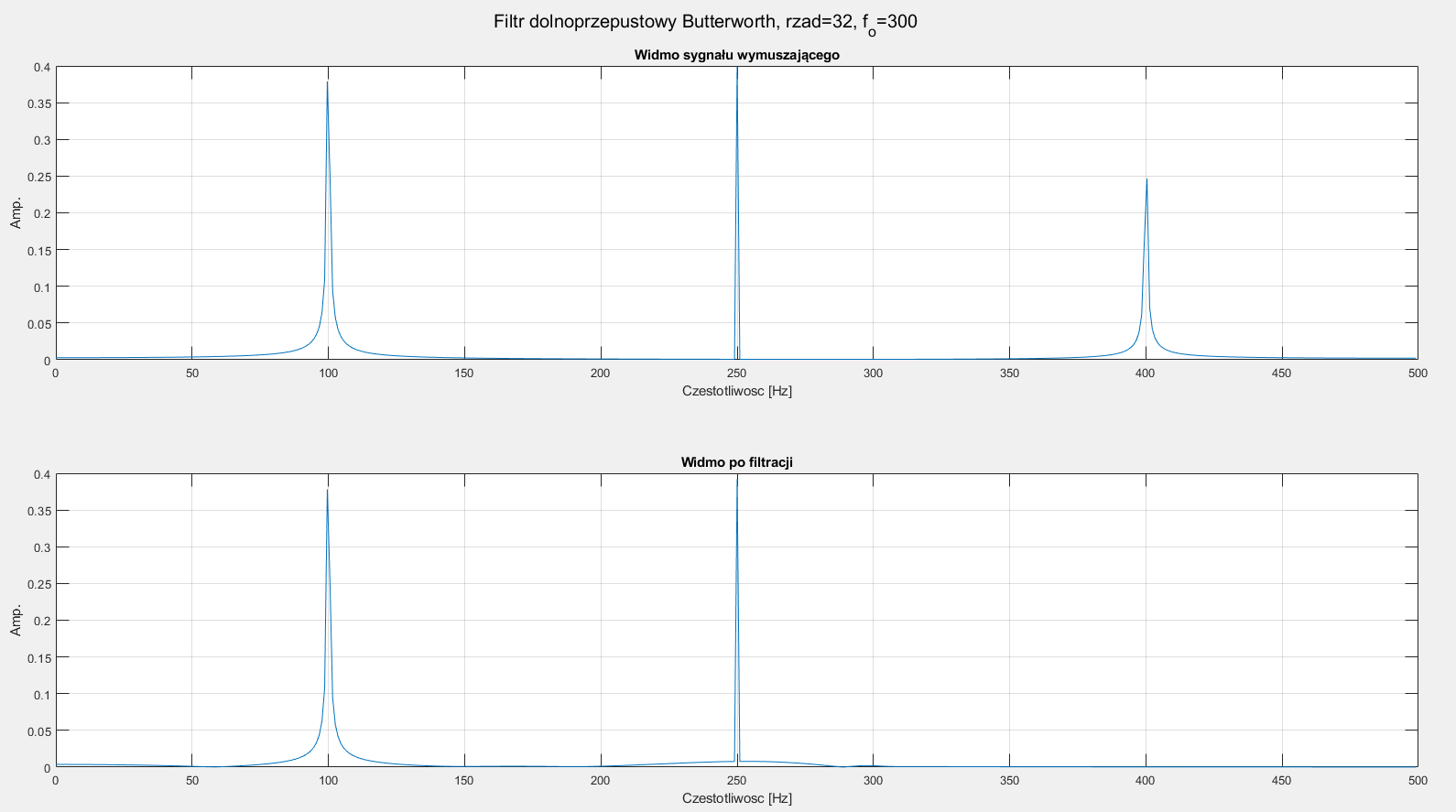
Celem tych ćwiczeń było porównanie działania filtrów FIR oraz IIR. Wykorzystane zostały metody butter(rząd, częstość\_odcięcia), która służy do projektowania filtrów rekursywnych analogowych oraz cyfrowych a także metodę fir1(rząd, częstość\_odcięcia) która z kolei służy do projektowania cyfrowych filtrów 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\_Mateusz\_Lab\_7.m".

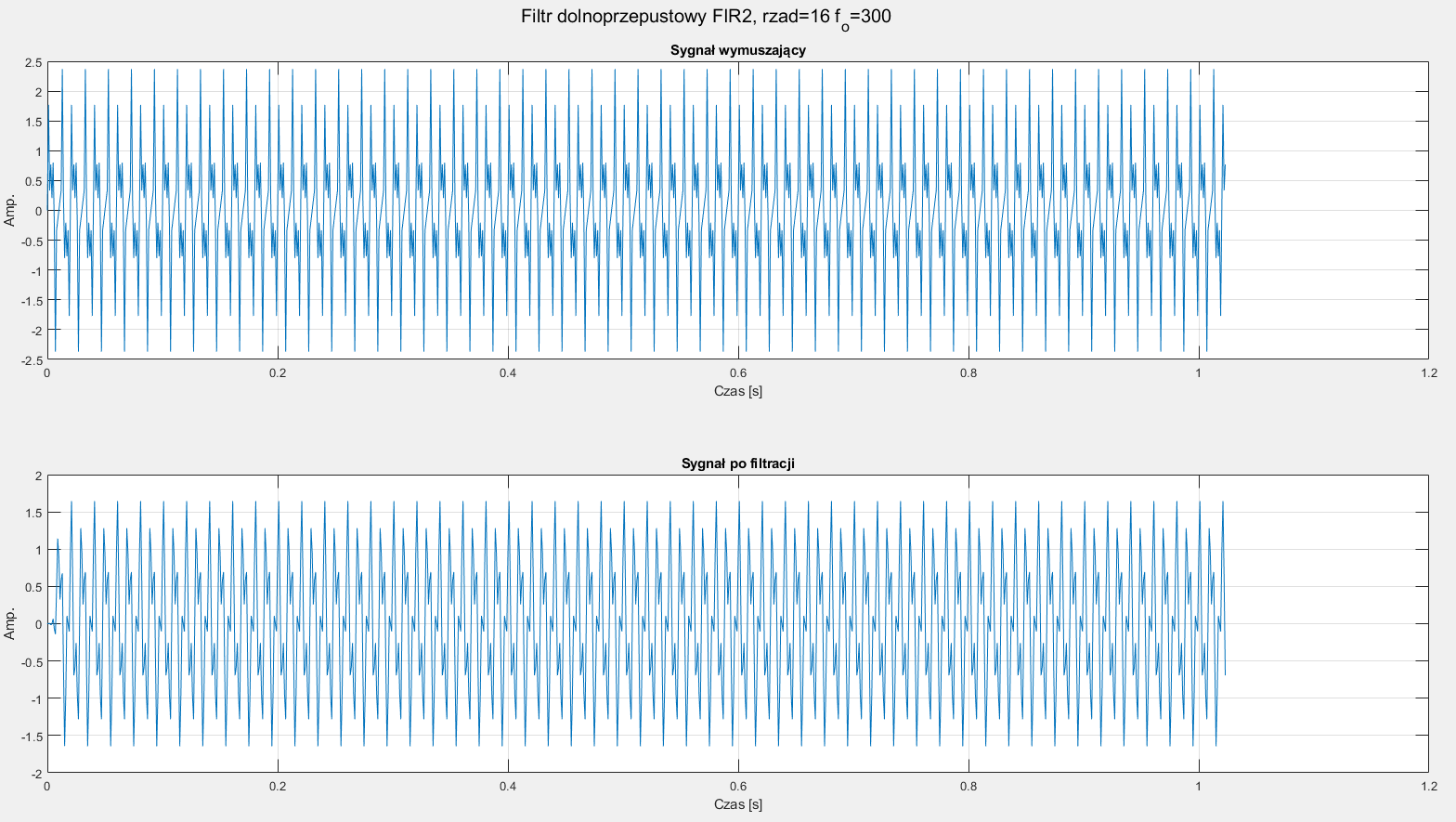
|  |
| --- |
| % Lab 7. Porównanie działania filtra rekursywnego i nierekursywnego.  % Metody fir1 i butter.  % Mateusz Krupnik'  clc; clear all; close all;    % Dane sygnałów z lab. 6  f1=100; f2=250; f3=400; fs=1000; % Czest. skladowych i Nyquista  A1=1; A2=0.8; A3=0.65; % Aplitudy skladowych  t=0:(1/fs):1.023; % 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 czestosci odciecia 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(fo1)]);  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(fo1)]);  subplot(211);  plot(f\_w, Wid(1,:)); title('Widmo sygnału wymuszającego');  xlabel('Czestotliwosc [Hz]'); ylabel('Amp.'); grid;  subplot(212);  plot(f\_w, Wid(2,:)); title('Widmo po filtracji');  xlabel('Czestotliwosc [Hz]'); ylabel('Amp.'); grid;    % Filtr metoda probkowania w dziedzinie czestotlwiosci  % 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(fo1)]);  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(fo1)]);  subplot(211);  plot(f\_w, Wid(1,:)); title('Widmo sygnału wymuszającego');  xlabel('Czestotliwosc [Hz]'); ylabel('Amp.'); grid;  subplot(212);  plot(f\_w, Wid(2,:)); title('Widmo po filtracji');  xlabel('Czestotliwosc [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 czestotliwości  T = 0:(1/fs):1.023;  % chirp(wektor czasu, czestestotliwosc start, koniec narastania,  % czestotliwoscdocelowa)  X = chirp(T,50,1.023,450); % funkcj 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 sygnalu 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 fitlracji FIR');    figure(11)  spectrogram(X\_iir, 256, 250, 256, 1E3, 'yaxis');  title('Sygnał po fitlracji IIR');      %% Filtr środkowoprzepustowy  % Filtr Butterwortha  rzad1 = 8; % rząd filtra  [B, A] = butter(rzad1, [wn2 wn1]); % dla czestosci odciecia nr 1  x\_filtered = filter(B, A, x); % filtracja syg. przez układ    % Filtr metoda probkowania w dziedzinie czestotlwiosci  % 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 srodkowoprzepustowy f\_o=[' num2str(fo1)...  ', ' num2str(fo2) ']']);  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(fo1)...  ', ' 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 czestotliwości  T = 0:(1/fs):1.023;  % chirp(wektor czasu, czestestotliwosc start, koniec narastania,  % czestotliwoscdocelowa)  X = chirp(T,50,1.023,450); % funkcj 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 sygnalu 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 fitlracji FIR');    figure(21)  spectrogram(X\_iir, 256, 250, 256, 1E3, 'yaxis');  title('Sygnał po fitlracji IIR'); |



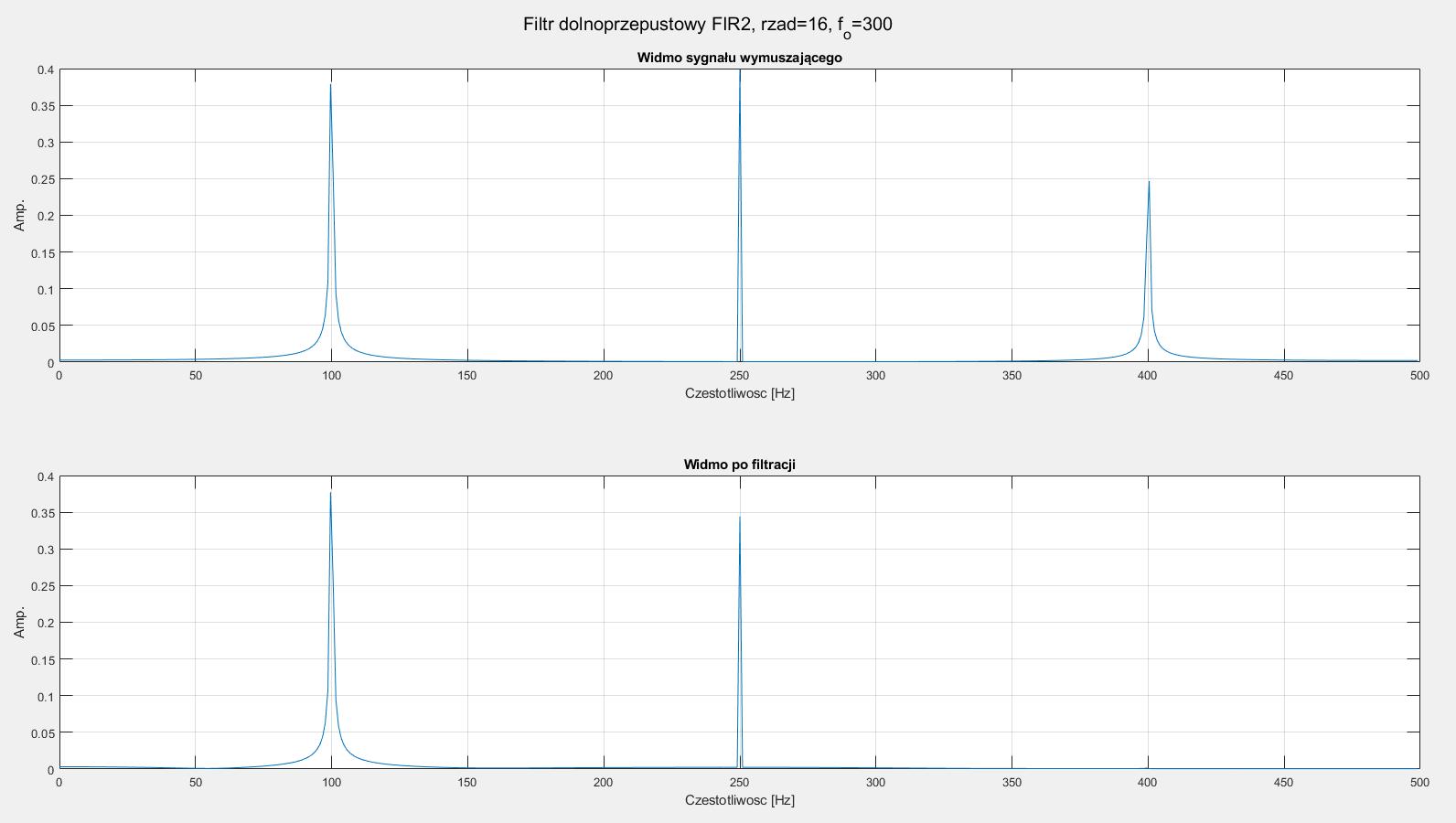
Rysunek 7.1. Sygnał wymuszenia oraz po filtracji. Zastosowano filtr Butterwortha 32 rzędu, częstotliwość odcięcia 300Hz.



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 nierekursywnym.

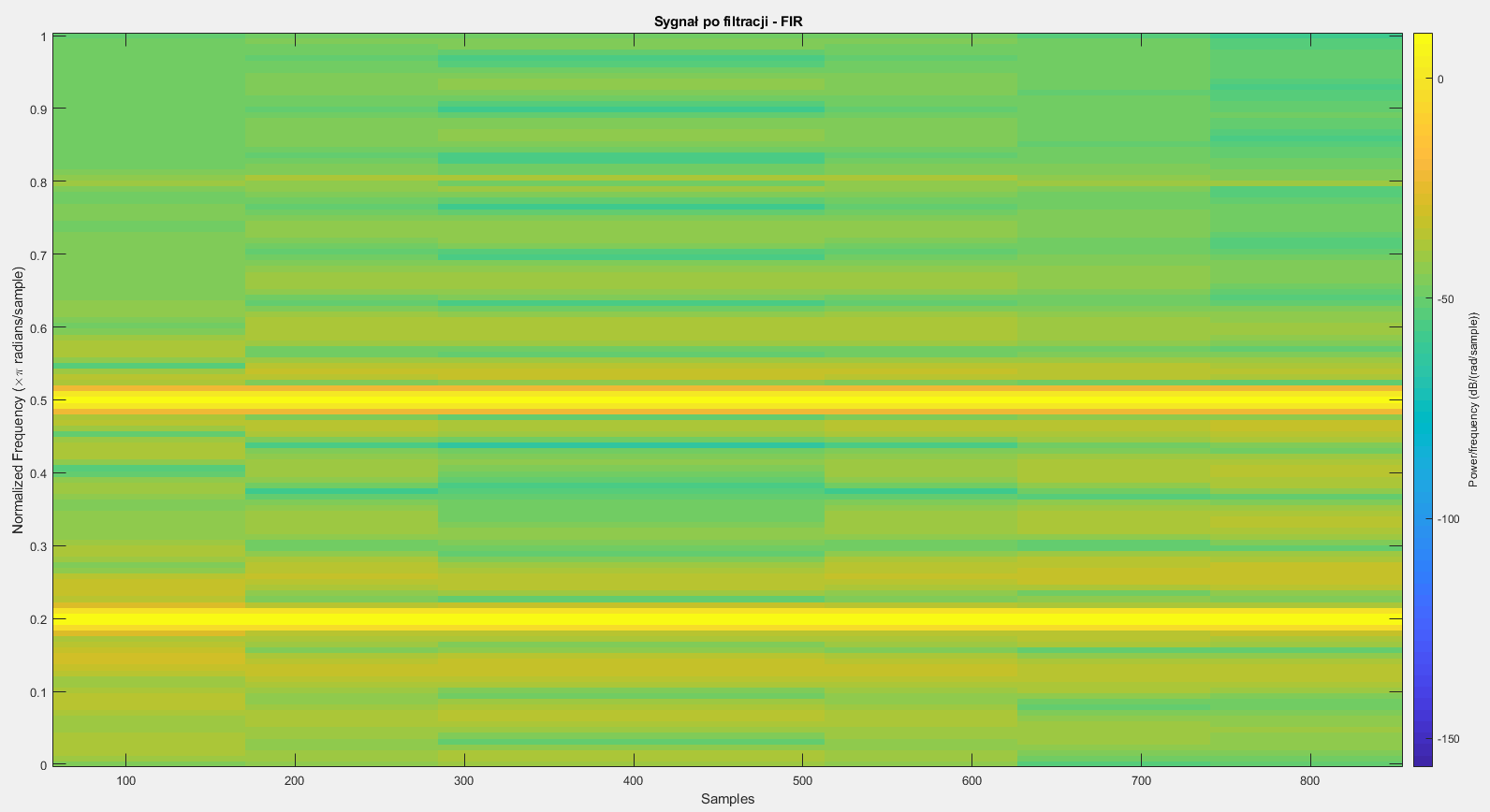
W przypadku analizy częstotliwościowej widoczne jest usunięcie wszystkich składowych powyżej częstotliwości odcięcia. Przeprowadzona jednak została analiza krótko-czasową za pomocą polecenia spectrogram, które zwraca charakterystykę czasowo-częstotliwościową.



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

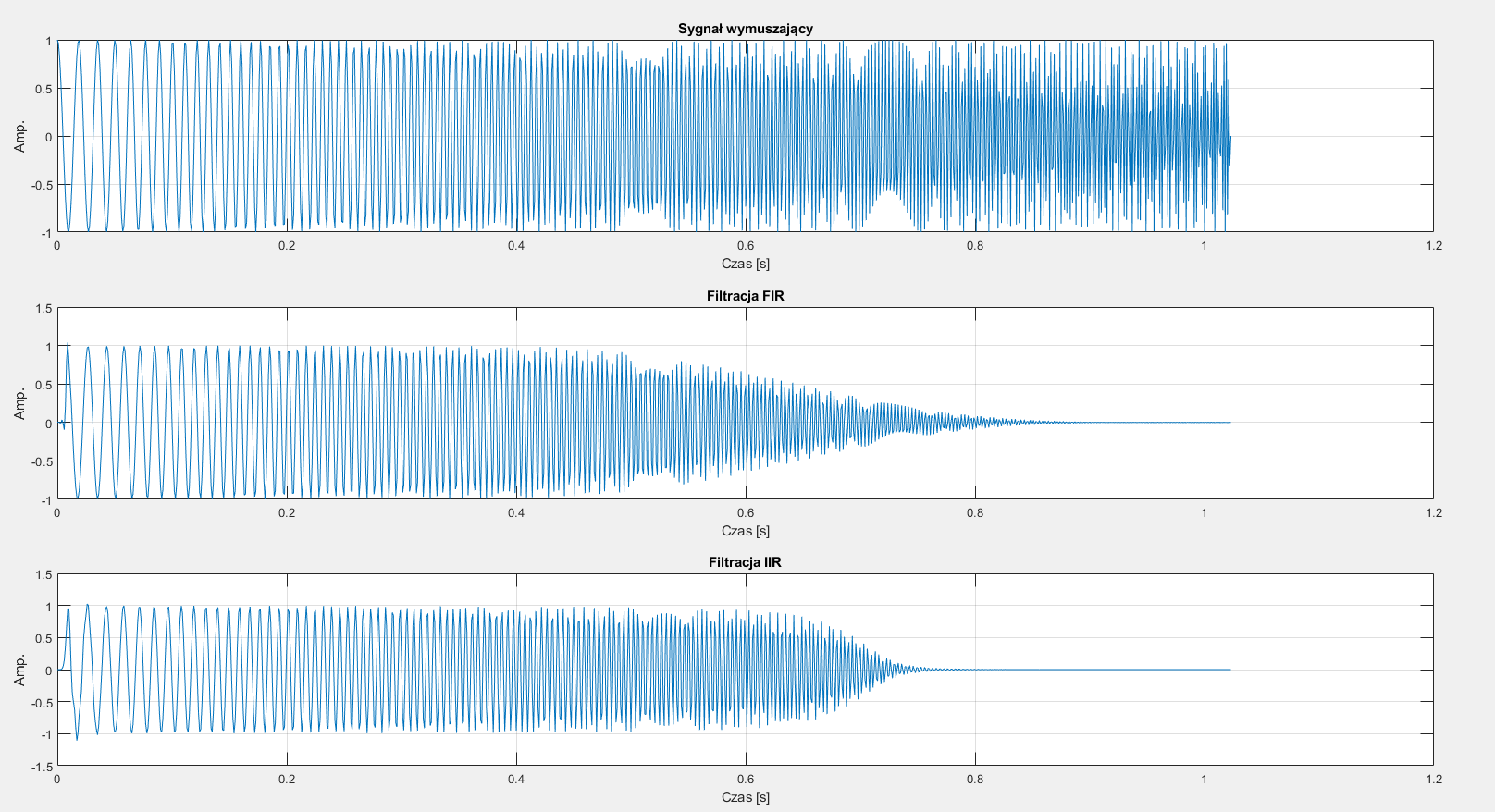


Rysunek 7.6. Spektrogram po filtracji filtrem IIR (Butterwortha).



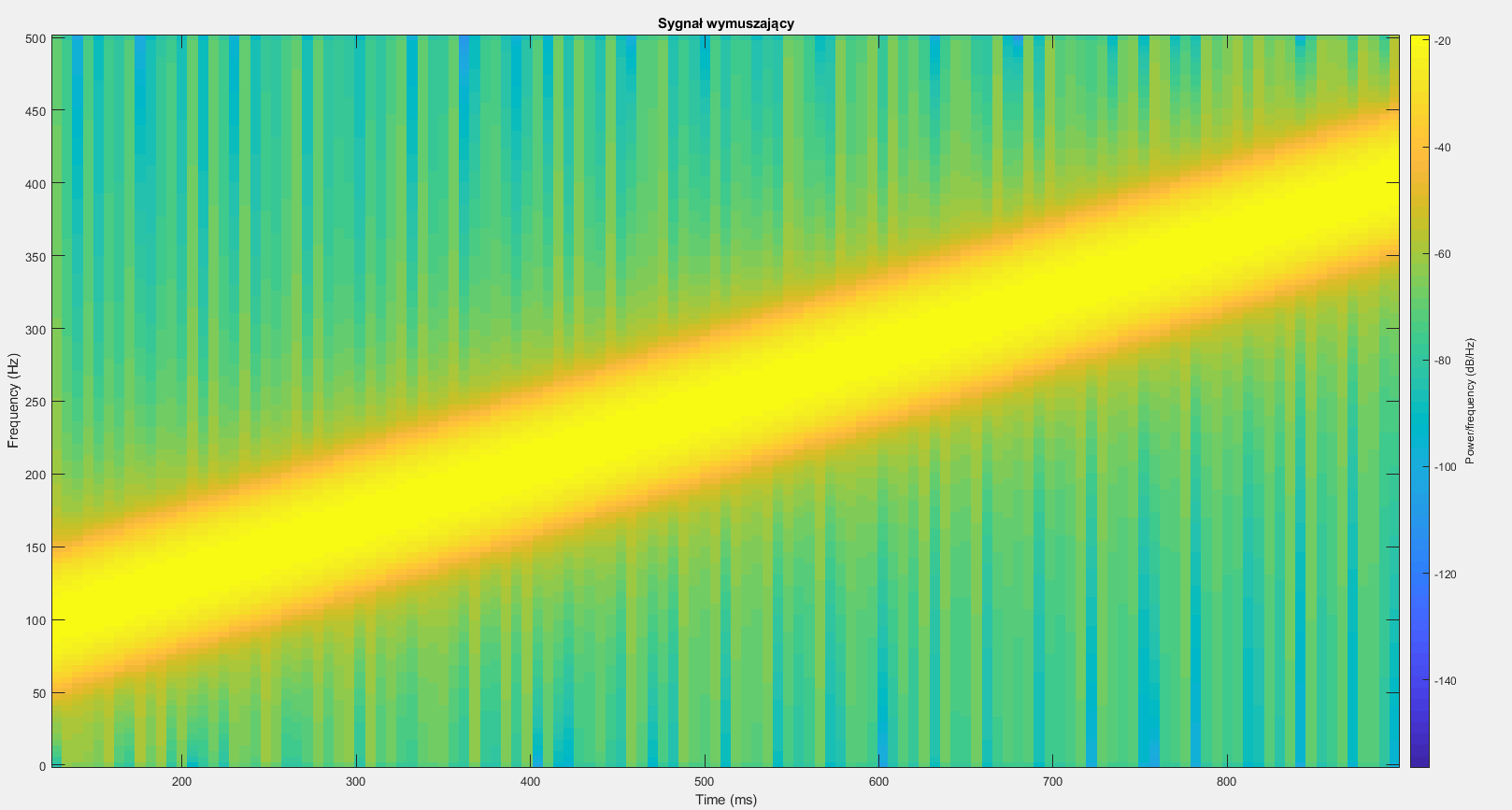
Rysunek 7.7. Spektrogram po filtracji filtrem FIR (fir1).

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.

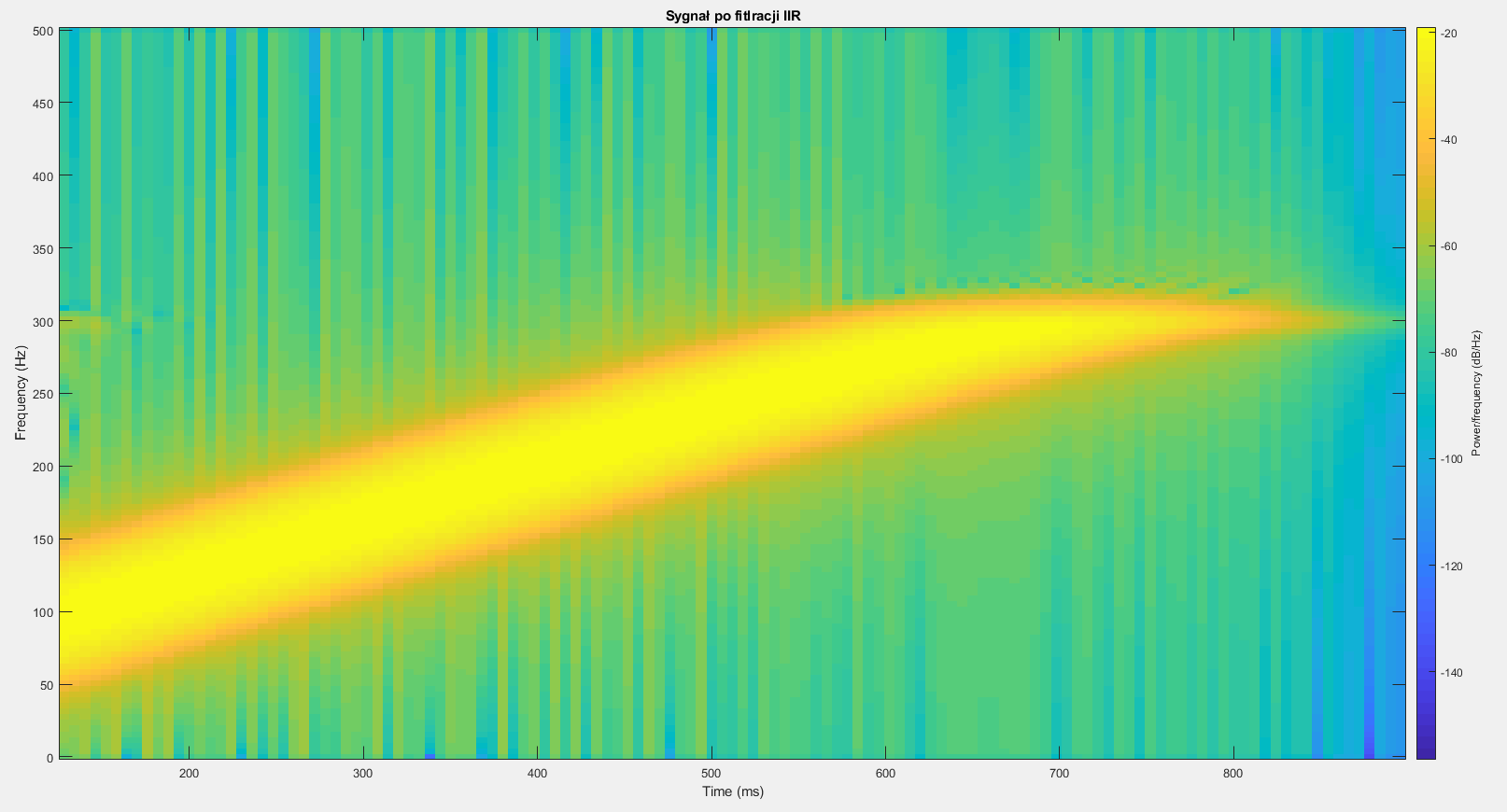


Rysunek 7.9. Spektrogram sygnału wymuszającego.

Widoczna jest linowa zmiana częstotliwości sygnału z czasem.



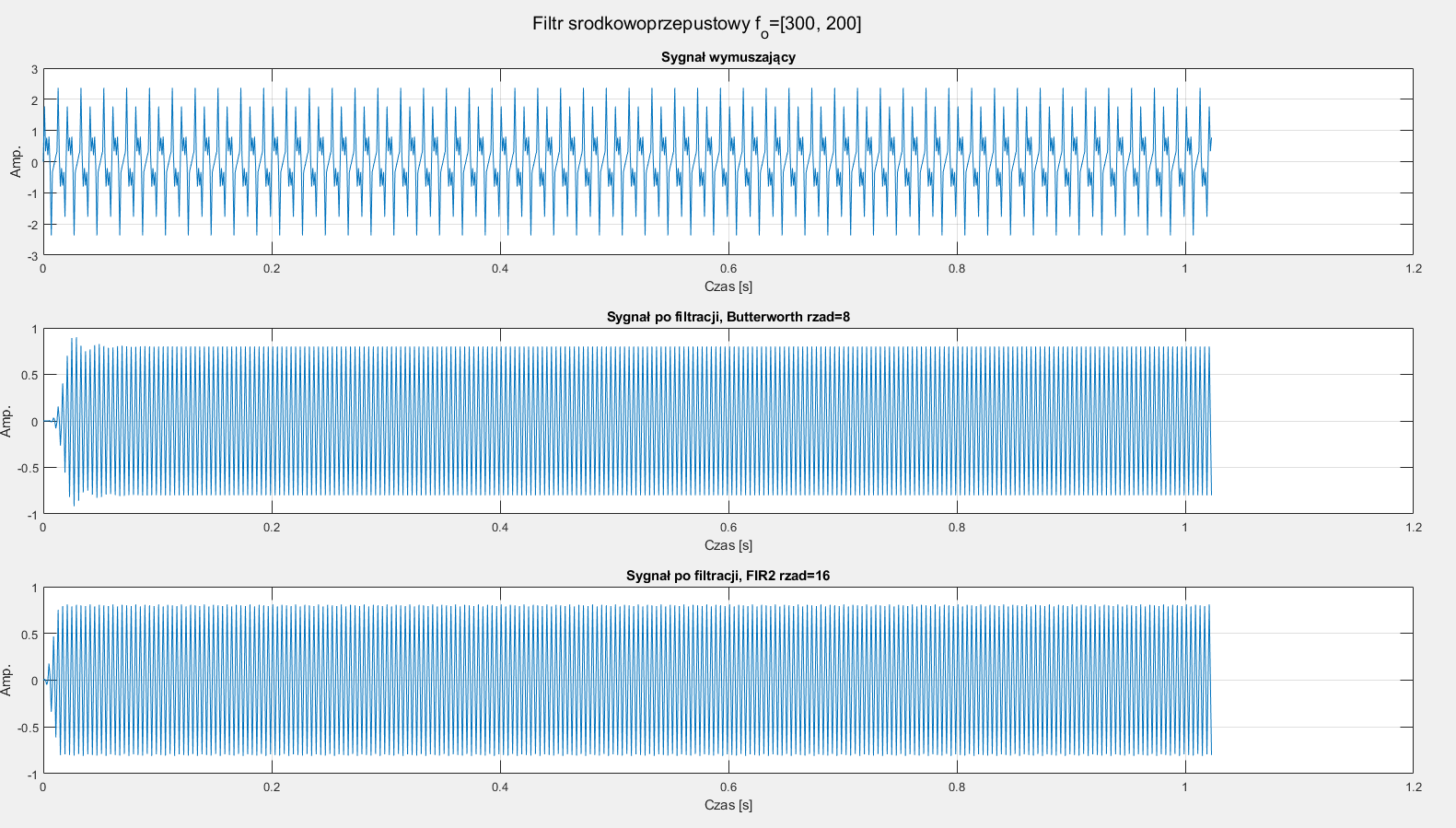
Rysunek 7.10. Spektrogram sygnału po filtracji filtrem nierekursywnym.



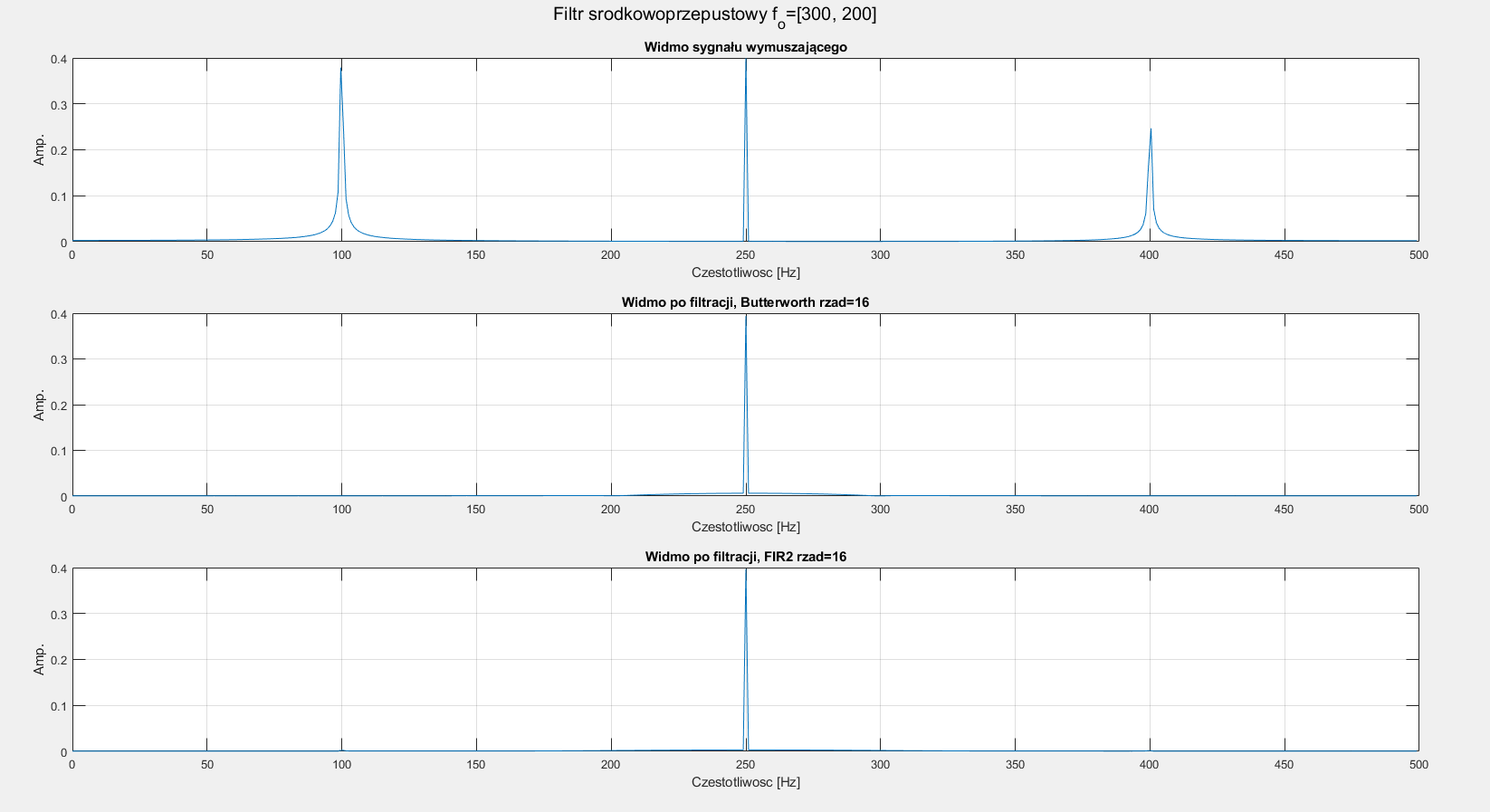
Rysunek 7.11. Spektrogram sygnału po filtracji filtrem rekursywnym.

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 środkowoprzepustowego o częstotliwościach odcięcia 200 i 300 Hz. W przypadku filtra rekursywnego zastosowany zastał filtr 8 rzędu a dla filtra nierekursywnego 16 rzędu.

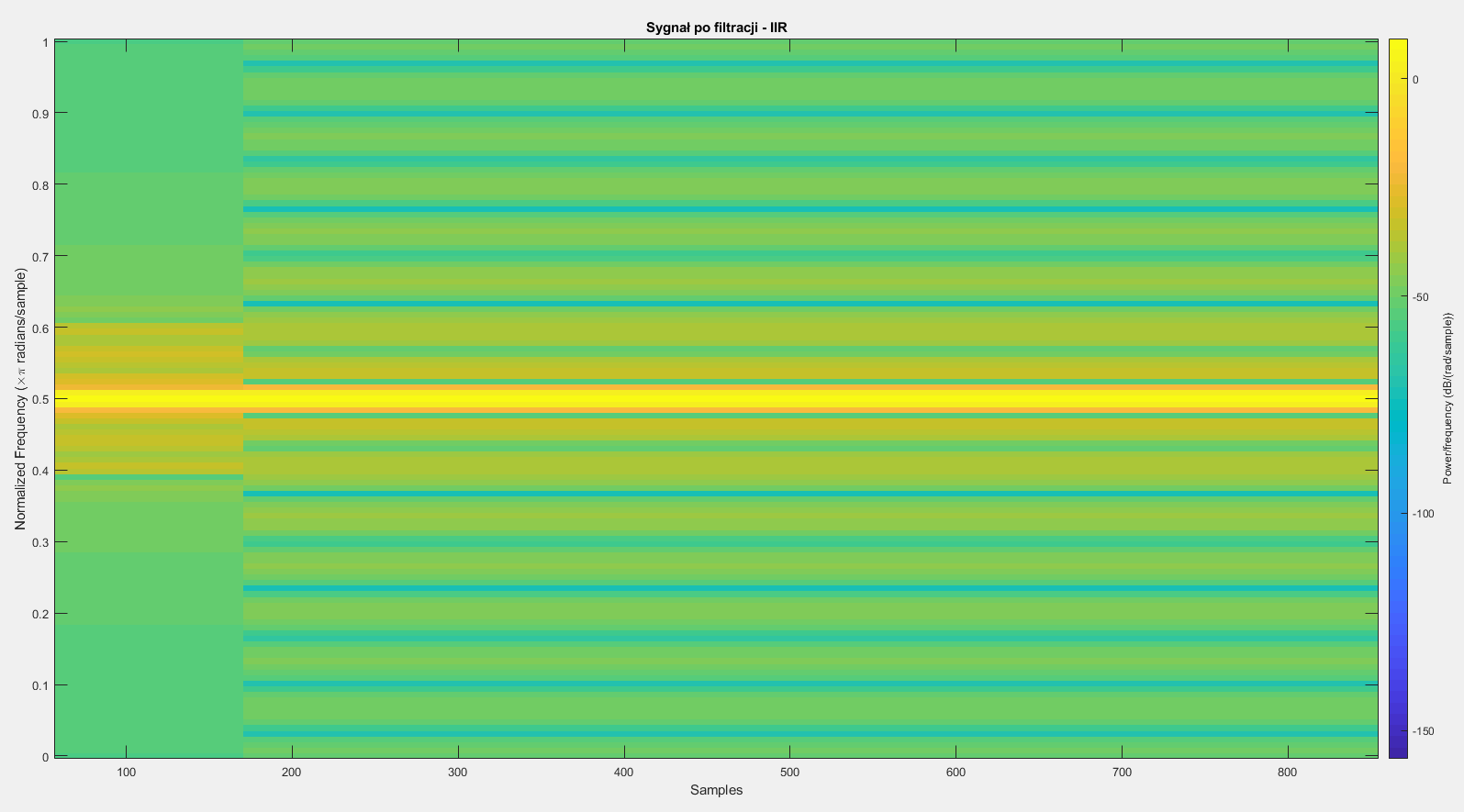


Rysunek 7.12. Przebiegi czasowe sygnałów: wymuszający, po filtracji filtrem rekursywnym, po filtracji filtrem nierekursywnym.

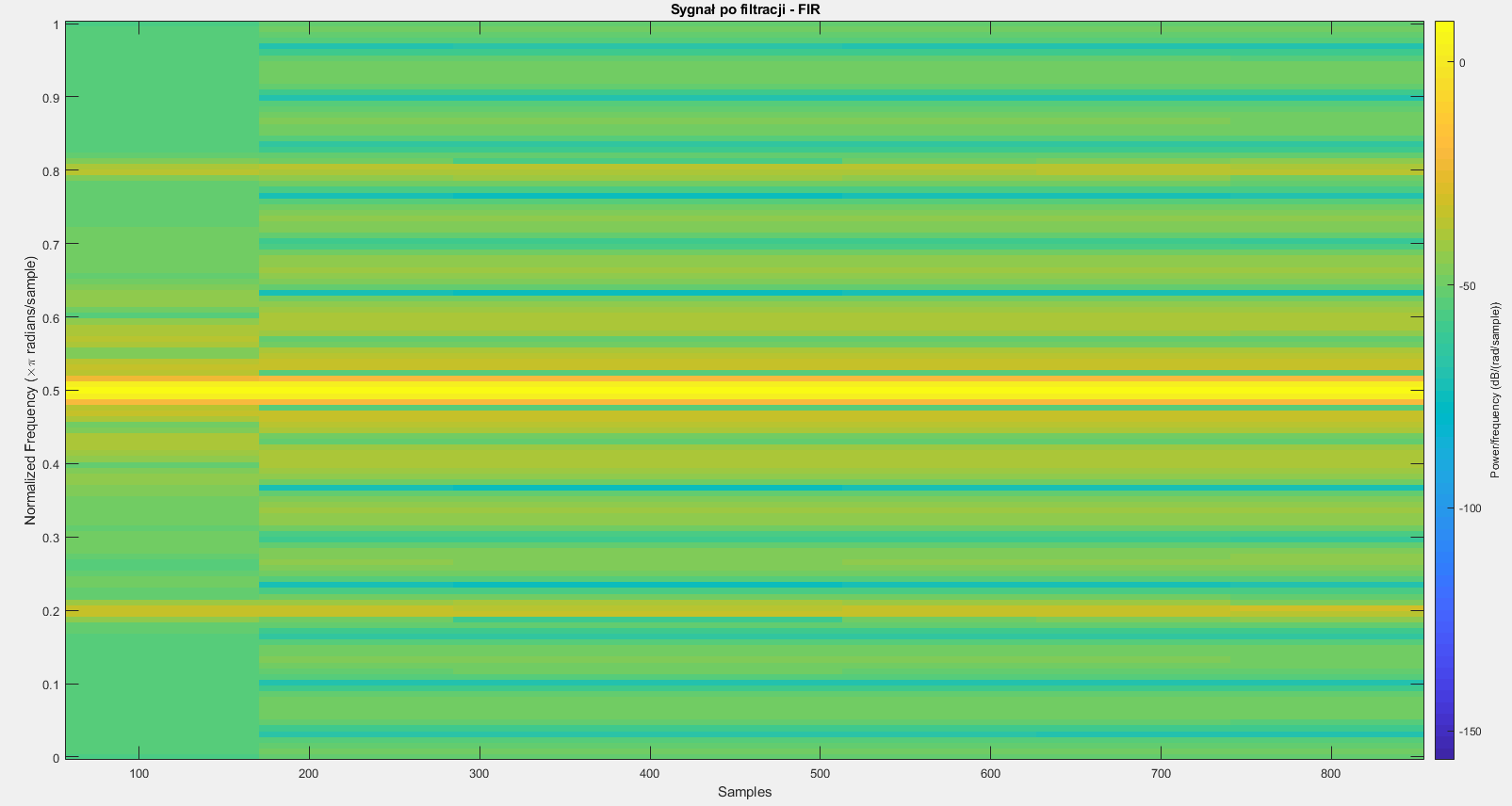


Rysunek 7.13. Widma sygnałów w kolejności jak na rysunku powyżej.

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.

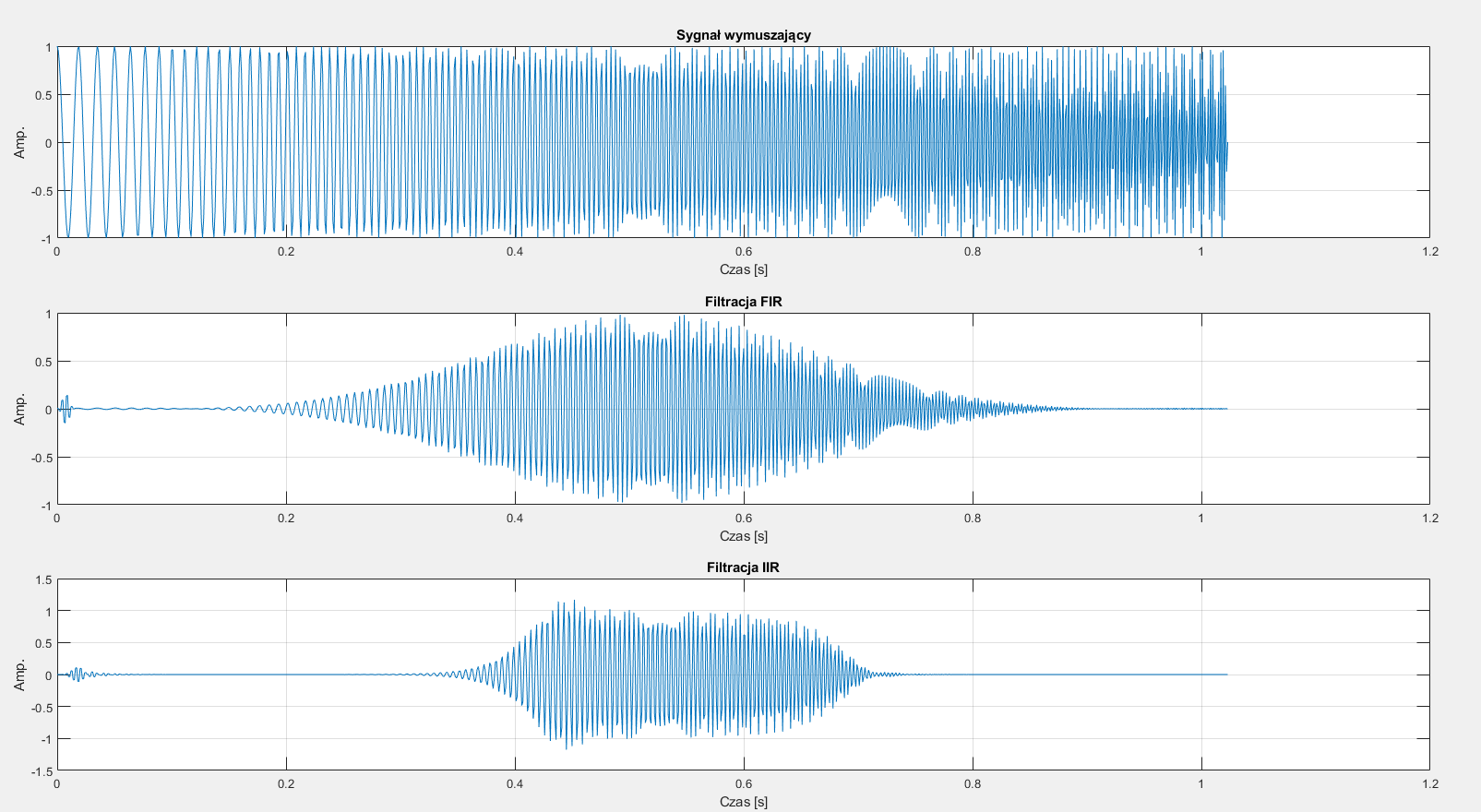


Rysunek 7.14. Spektrogram po filtracji filtrem rekursywnym.

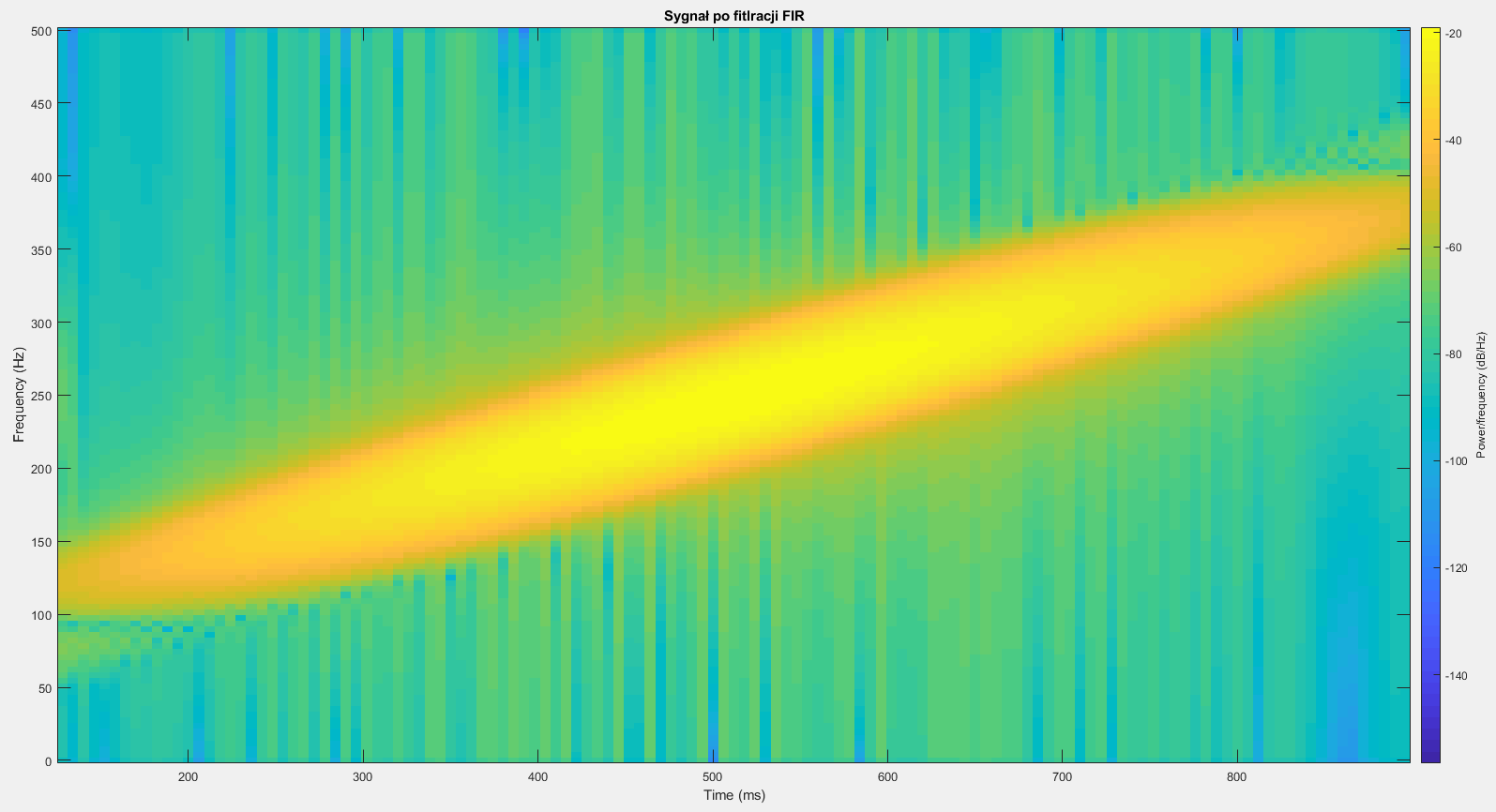


Rysunek 7.15. Spektrogram po filtracji filtrem nierekursywnym.

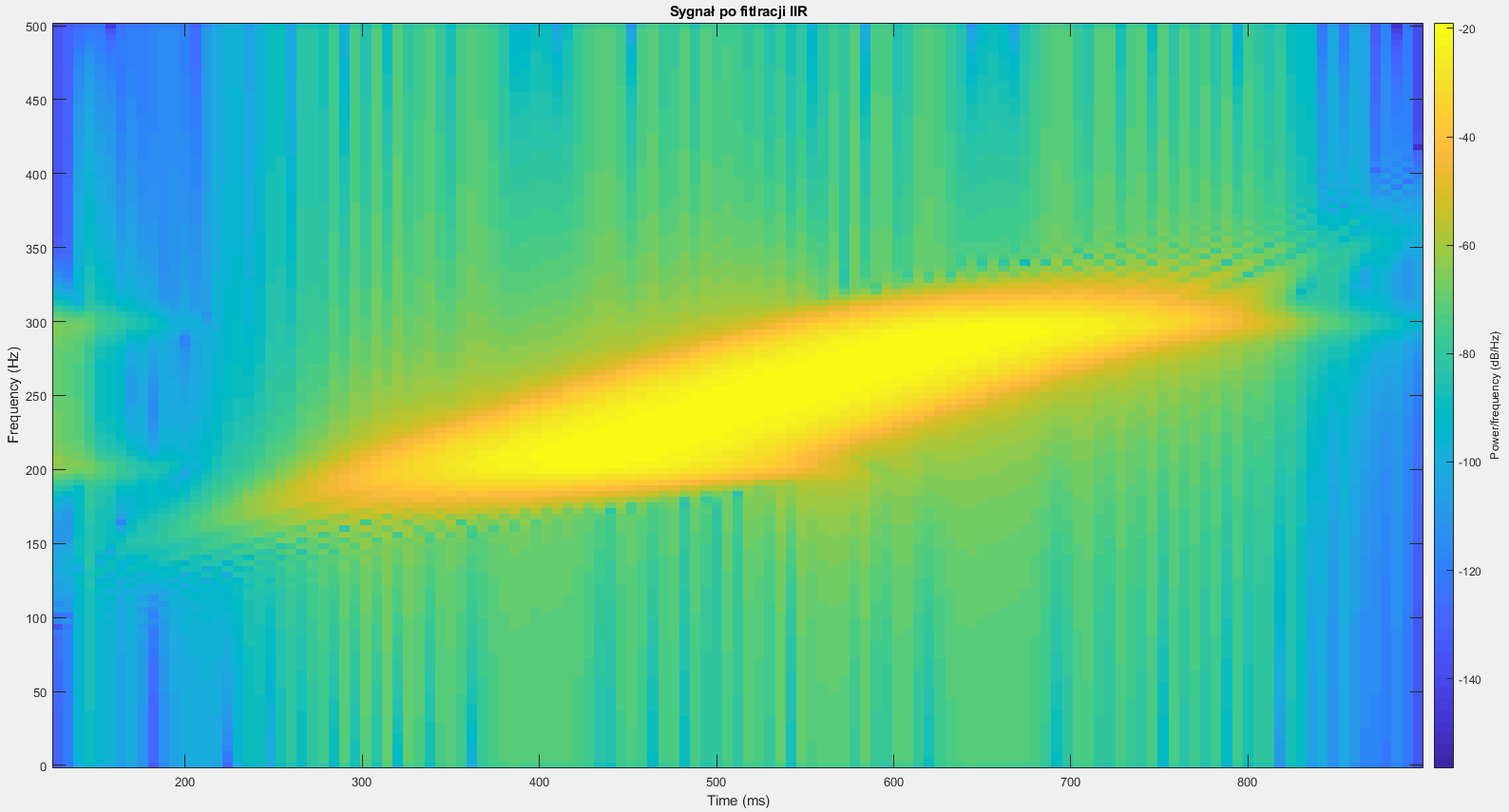
W przypadku filtra FIR widoczne są pozostałości składowych poza pasmem przepustowym. W dalszej części rozdziału widoczne są charakterystyki czasowo-częstotliwościowe sygnału zmodulowanego po filtracji. Ponownie filtr rekursywny wykazuje się słabszym i wolniejszym działaniem od filtra nierekursywnego, 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 środkowo-przepustowej odpowiednio filtrem nierekursywnym i rekursywnym.



Rysunek 7.17. Spektrogram sygnału po filtracji środkowo-przepustowej filtrem nierekursywnym.



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