

**Wydział Inżynierii Mechanicznej i Robotyki**

**Informatyka w inżynierii mechanicznej**

Sprawozdanie z przedmiotu Cyfrowe Przetwarzanie Sygnałów

285608, Mateusz Krupnik

# 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 i 2

W ramach tych ćwiczeń zrealizowane zostały programy mające zaznajomić z operacjami wejścia i wyjścia, czyli zapisywania i odczytywania danych z plików. Plik realizujący oba ćwiczenia nazwany jest „Laboratoria\_nr\_oraz\_2.m”.

Podstawowe polecenia to funkcje:

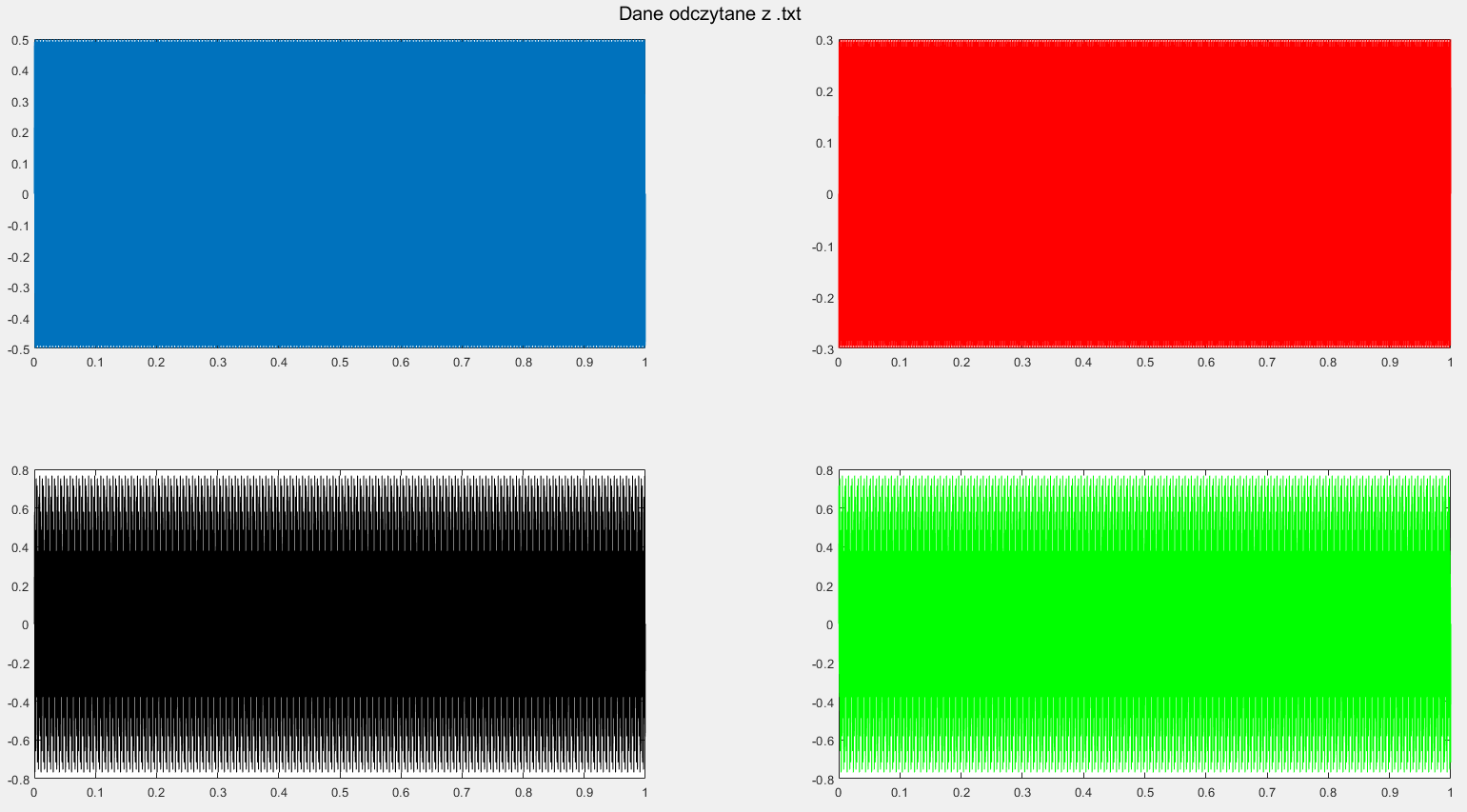
* fopen – funkcja otwierająca plik o podanej nazwie, z podanym trybem (zapis, odczyt),
* fprint – funkcja zapisywania danych do otwartego pliku, podać należy: format, zmienną z przestrzeni roboczej,
* fclose – funkcja zamykająca otwarte pliki,
* fscanf – funkcja odczytywania danych z pliku o zadanym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

W drugiej sekcji (laboratoria nr 2) 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. .. Kod pierwszego programu.

|  |
| --- |
| %% 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,:))    %% 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 = [signal\_min(y1) signal\_min(y2) signal\_min(y3) signal\_min(y4)]  b = [signal\_max(y1) signal\_max(y2) signal\_max(y3) signal\_max(y4)]  % Wyznaczenie wartości średniej za pomocą funkcji signal\_mean - mean()  y\_mean = [signal\_mean(y1) signal\_mean(y3) ...  signal\_mean(y3) signal\_mean(y4)]  % Energia sygnalu - za pomocą funkcji signal\_energy()  e = [signal\_energy(y1) signal\_energy(y2) ...  signal\_energy(y3) signal\_energy(y4)] |



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

Wyznaczone parametry sygnałów przedstawione są w tabeli 2.2.

Tab. .. Parametry sygnałów sinusoidalnych.

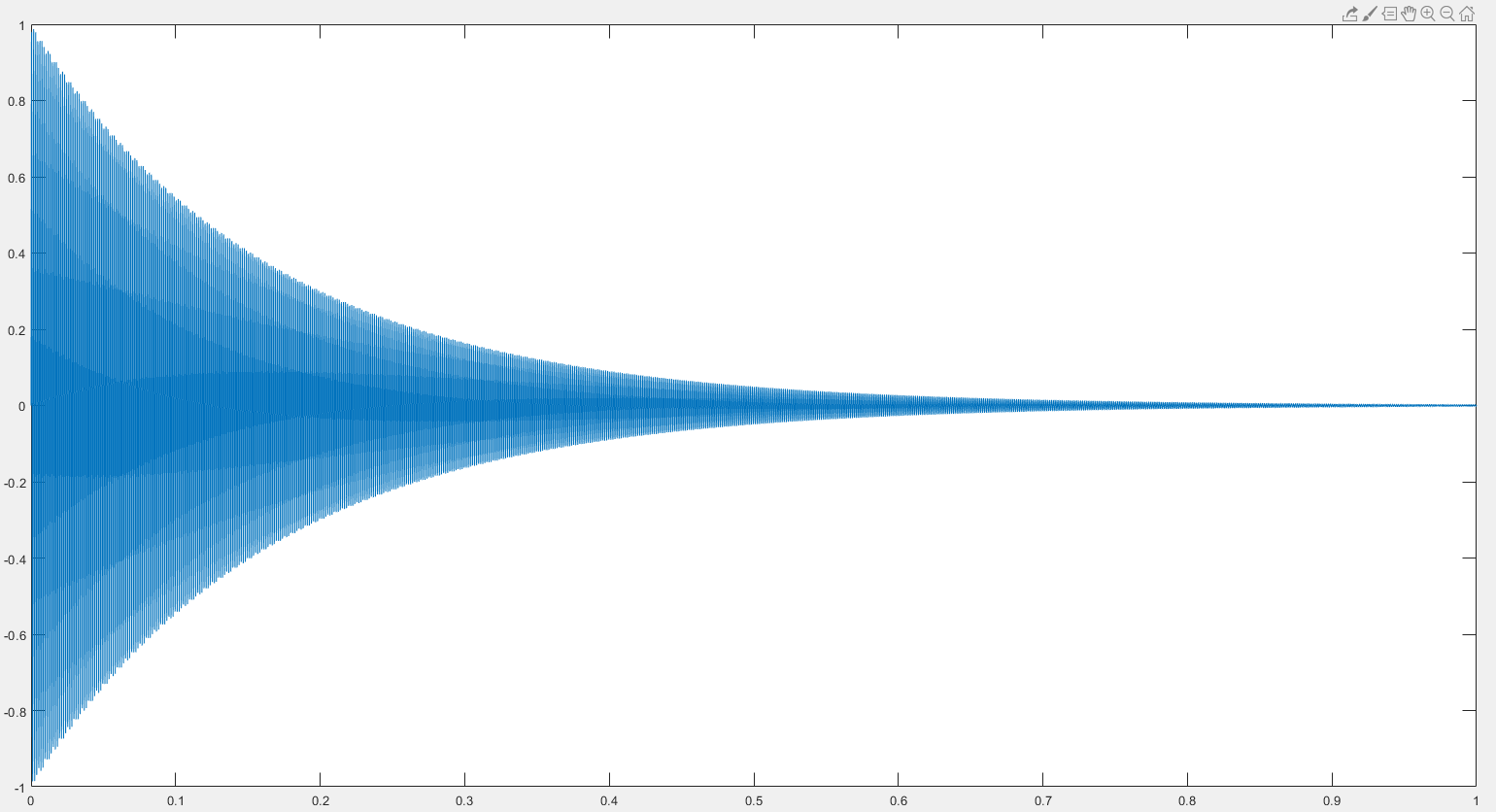
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sygnał  Parametr | Niebieski (1) | Czerwony (2) | Czarny (3) | Zielony (4) |
| Minimum | -0.5000 | -0.2994 | -0.7675 | -0.7675 |
| Maksimum | 0.5000 | 0.2994 | 0.7675 | 0.7675 |
| Średnia (x10-14) | -0.1417 | -0.1480 | -0.1480 | -0.1354 |
| Energia |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

# Laboratorium 3

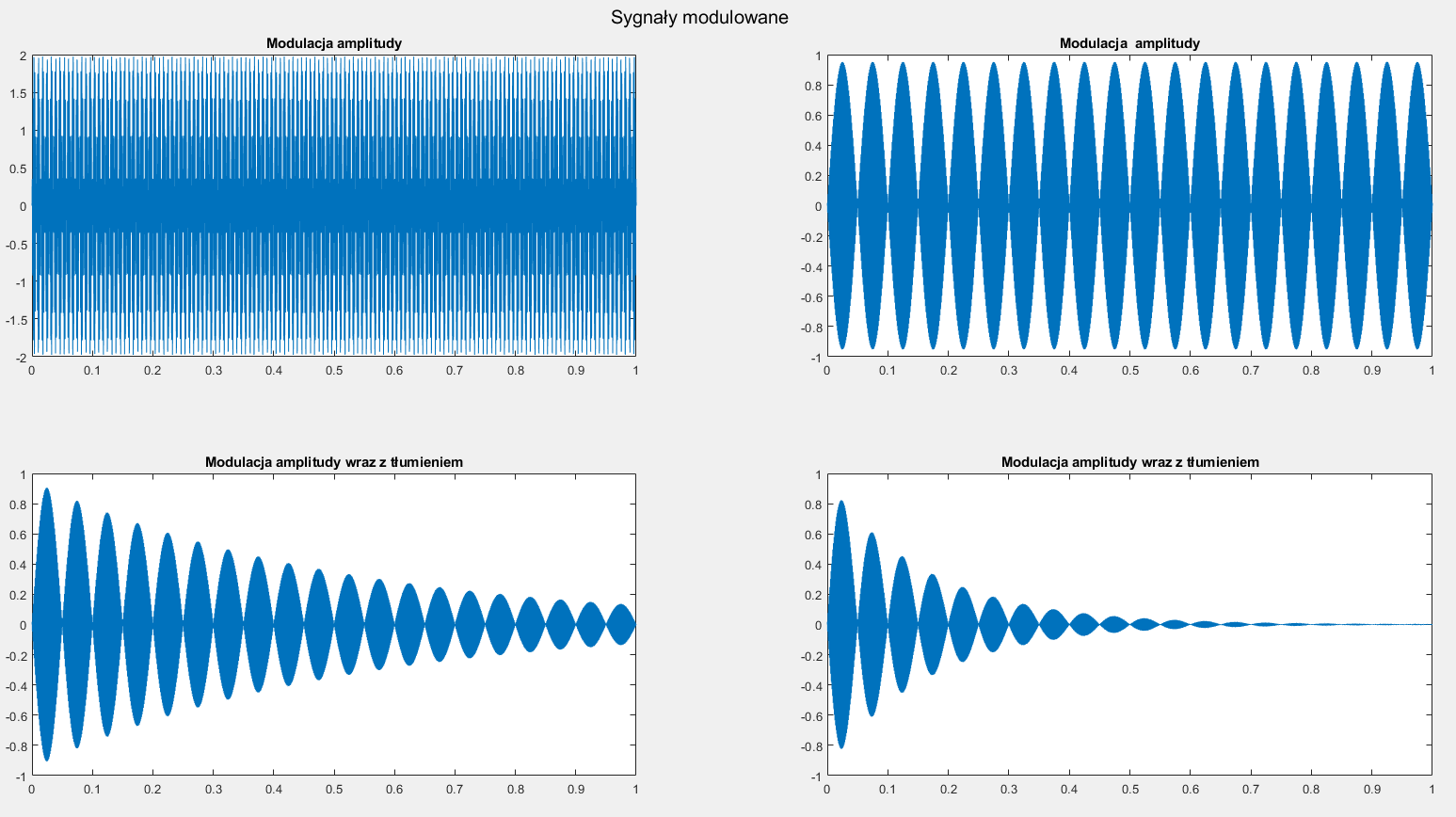
W ramach tych ćwiczeń realizowane zostały dodatkowo operacje na plikach dźwiękowych. Wykorzystywane funkcje to m. in. audiowrite służąca do zapisu sygnału w postaci pliku dźwiękowego np. o rozszerzeniu .wav. Ponownie funkcja sound pozwala na odtworzenie dźwięku. Generowane zostały sygnały sinusoidalne,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

Tab. .. Kod programu drugiego.

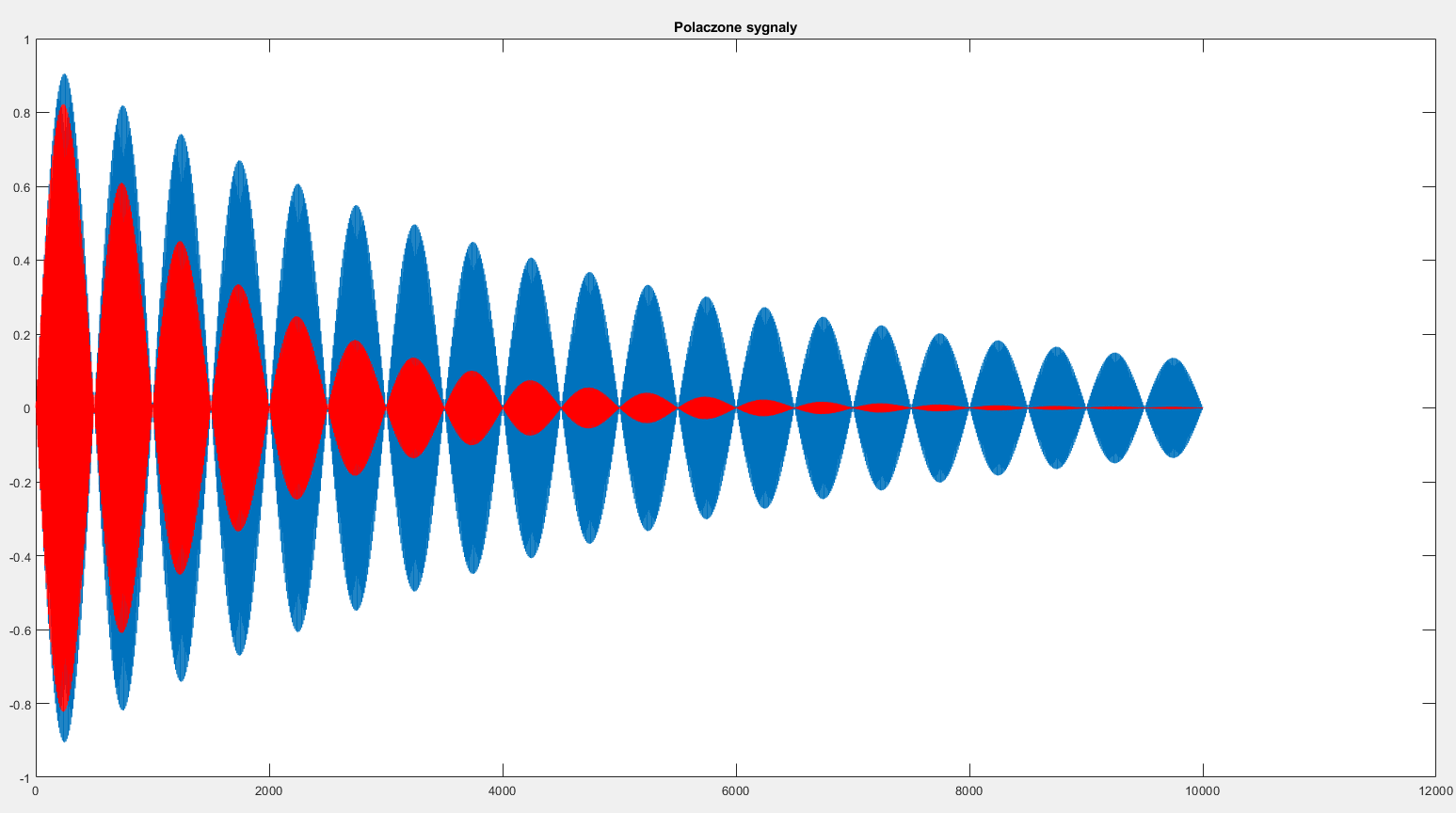
|  |
| --- |
| %% Lab 3 - praca z plikiem dziwiękowym - Mateusz Krupnik  clc; clear all; close all;  % Generowanie przebiegu, zapis i odczyt w postaci pliku wav  A=0.5;  B=-0.3;  f1=700;  fs=10000;  t=0:(1/fs):1;  y1=A\*sin(2\*pi\*f1\*t);  audiowrite('plik.wav', y1, fs);  clear all; % usuniecie danych  %% 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)); |



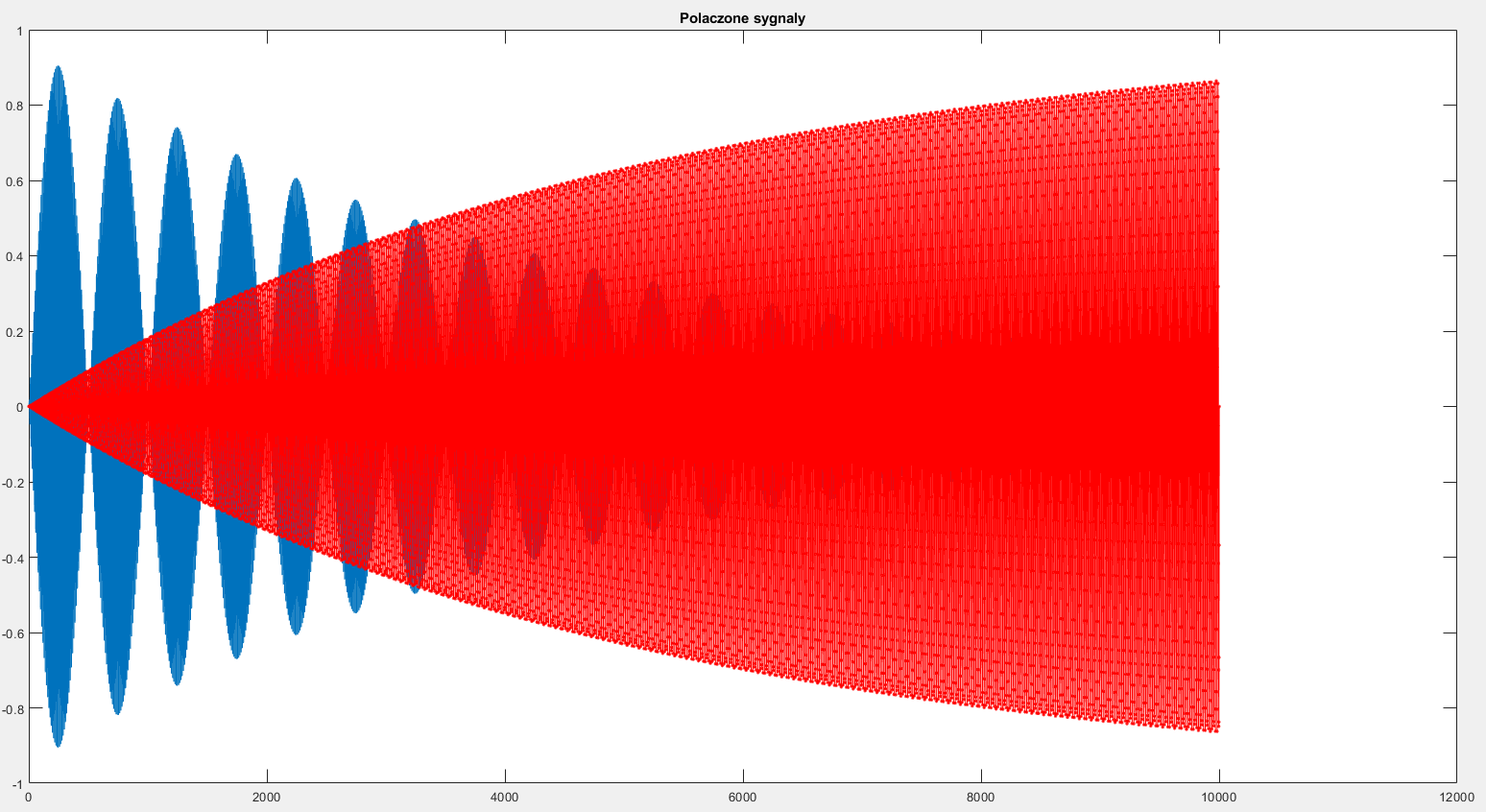
Rys. .. Przykładowy przebieg sygnału sinusoidalnego z tłumieniem wykładniczym.



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



Rys. .. Połącznie dwóch sygnałów o różnym tłumieniu.



Rys. .. 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:

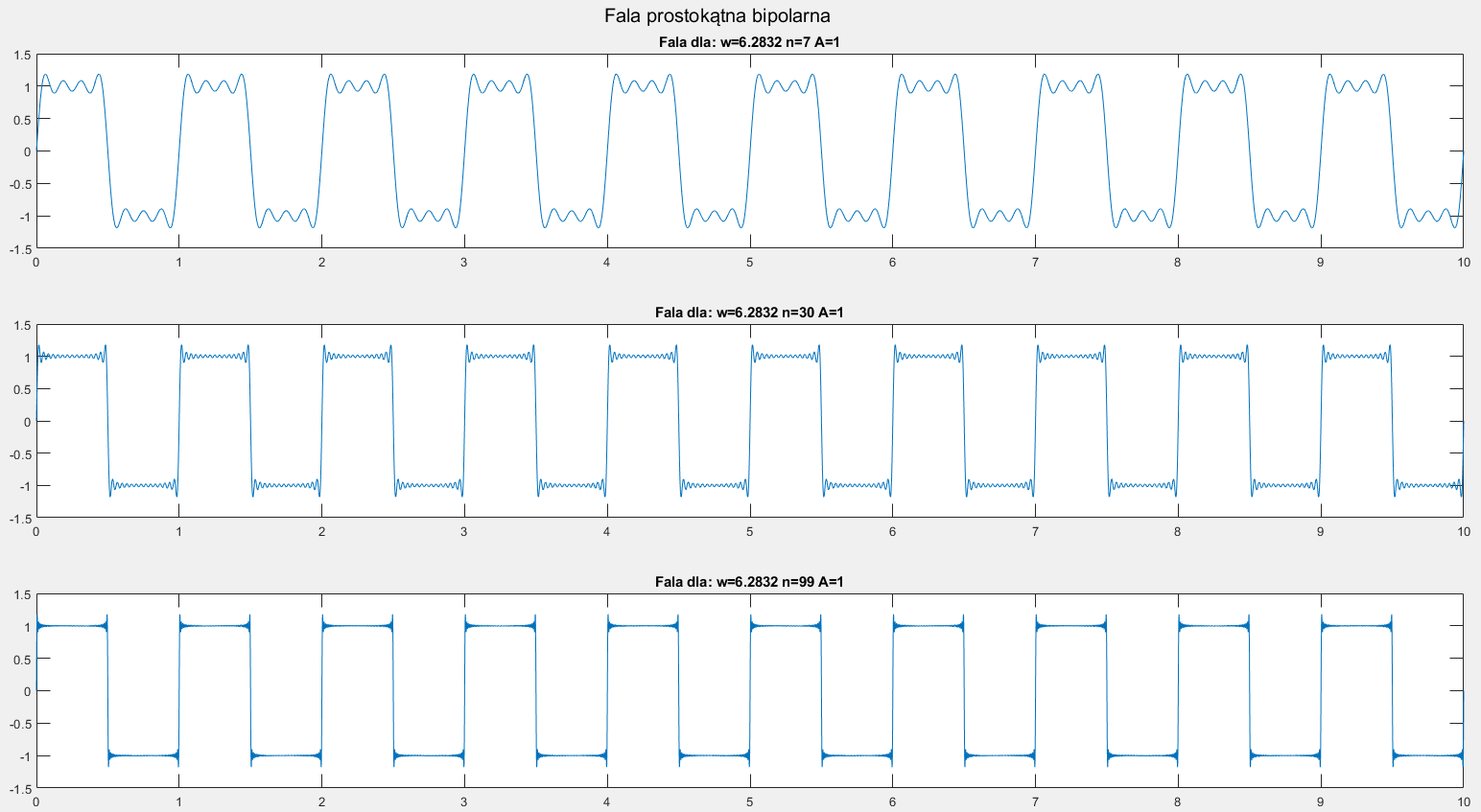
w – częstość [rad./s], A – amplituda [-], t - wektor czasu [s], n – rząd ciągu [-], f – częstotliwość [Hz], tau – wypełnienie [-] (od 0 do 1).

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.

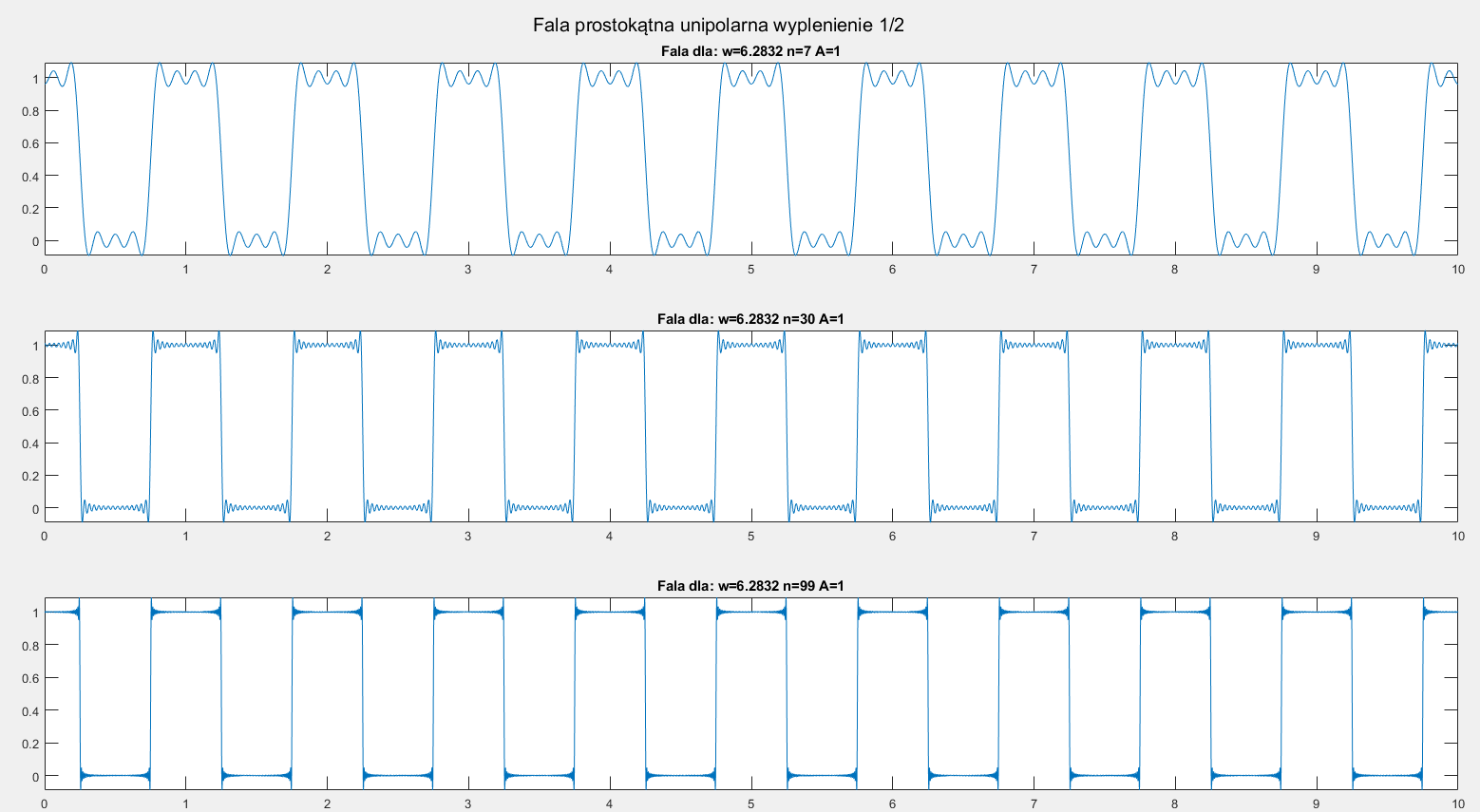
Tab. .. Kod programu trzeciego.

|  |
| --- |
| % 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(7)  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: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:n(i)  % x(i,:) = x(i,:) + ((1/j)\*cos(j\*w\*t));  % end  % for j=3:4: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 4.2 przedstawione są wzoru szeregów aproksymujących opisane funkcje. Dodatkowo przedstawione zostaną wykresy funkcji dla różnych rzędów ciągów aproksymujących.

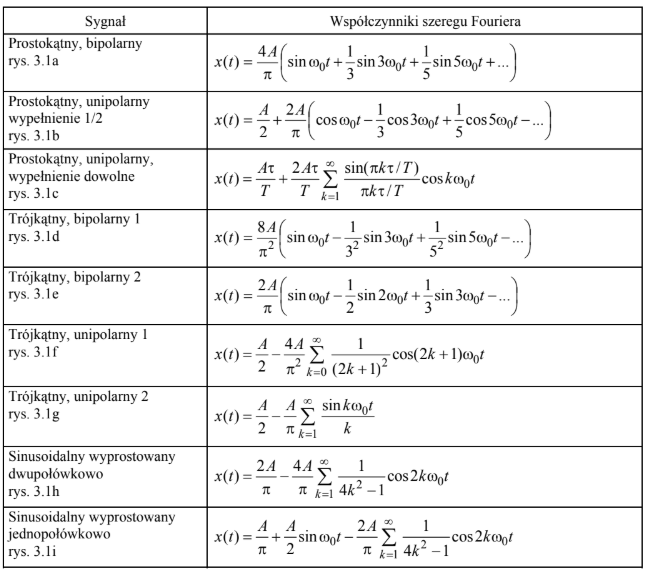


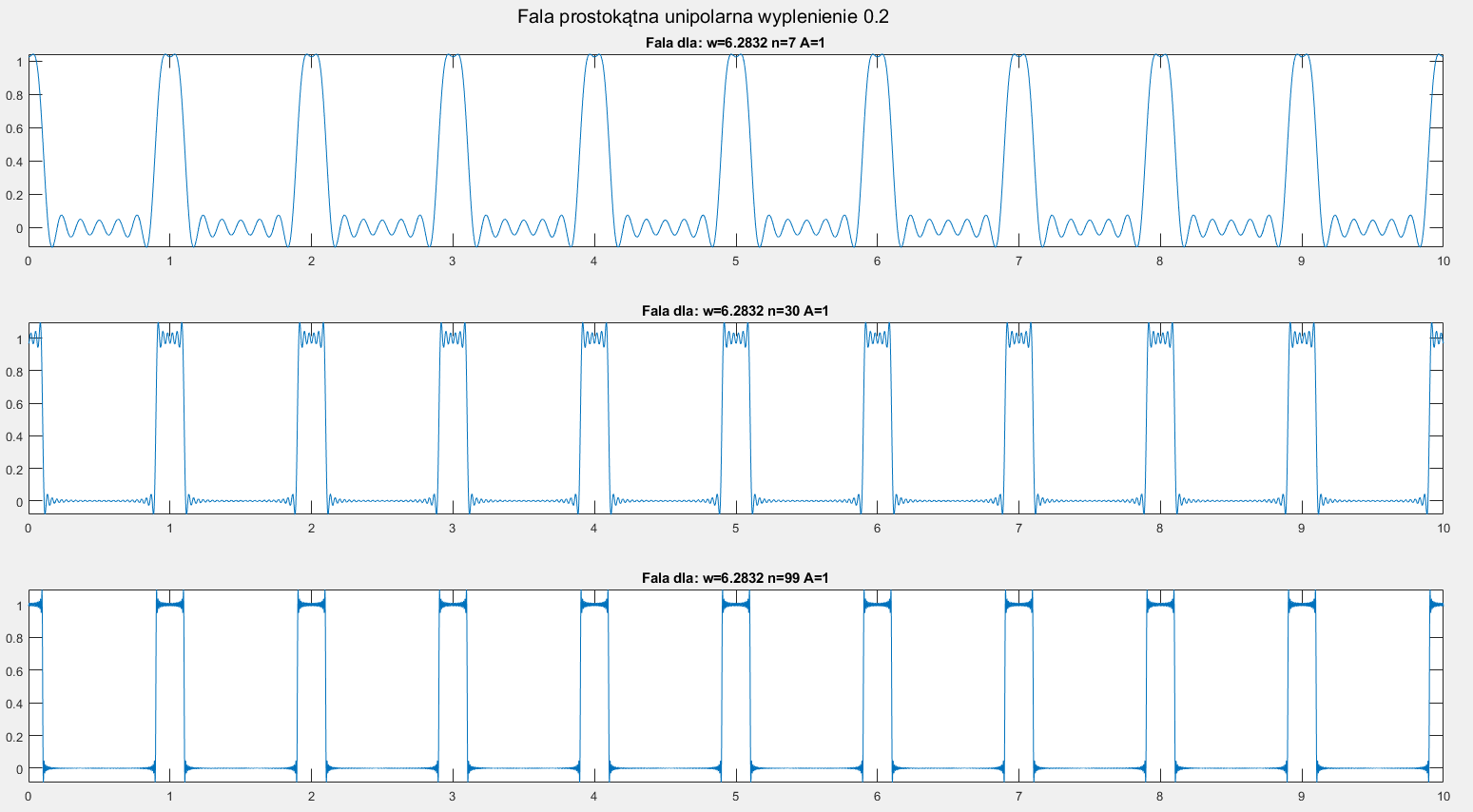
Rysunek .. Fala prostokątna bipolarna.



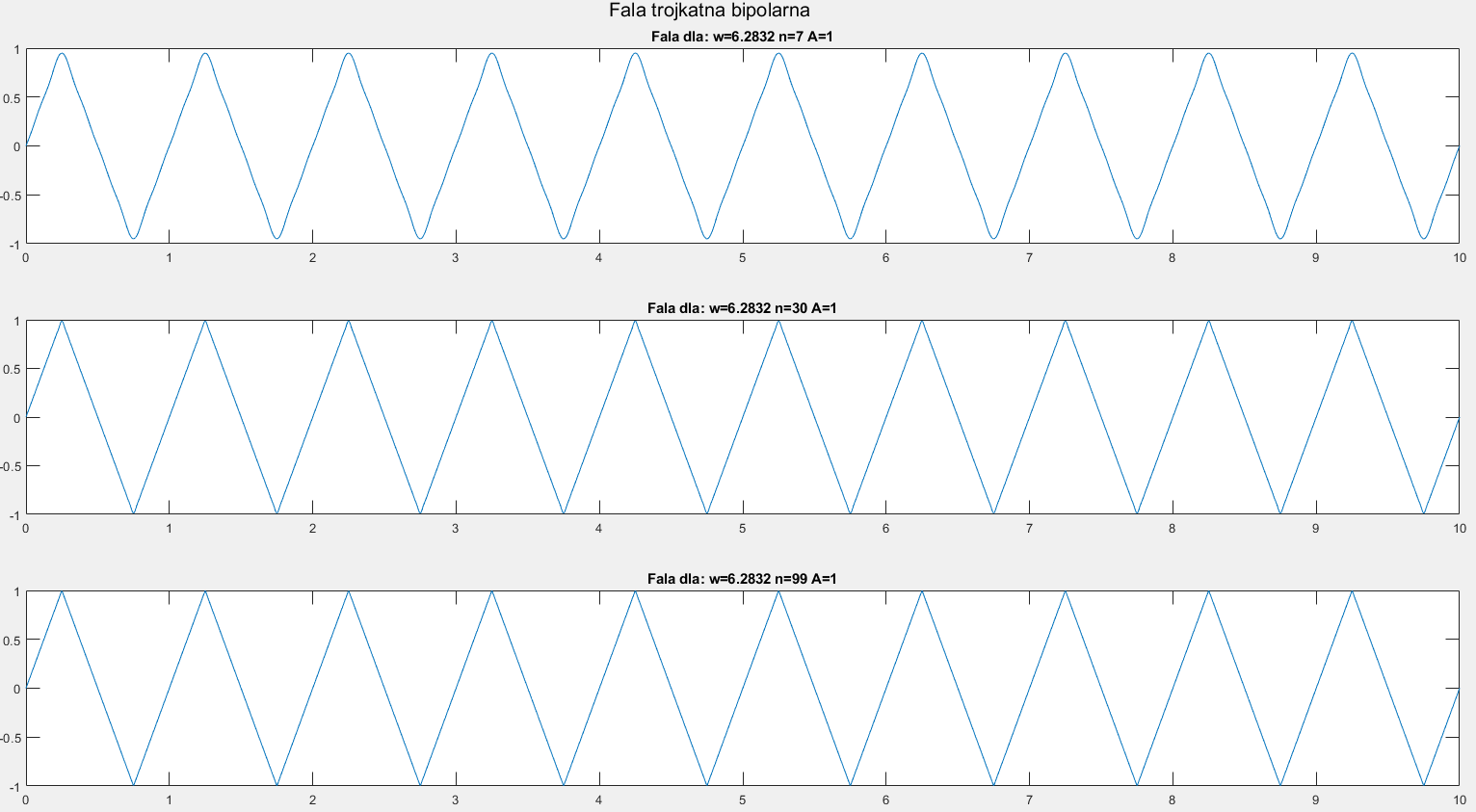
Rysunek .. Fala unipolarna o wypełnieniu 1/2.

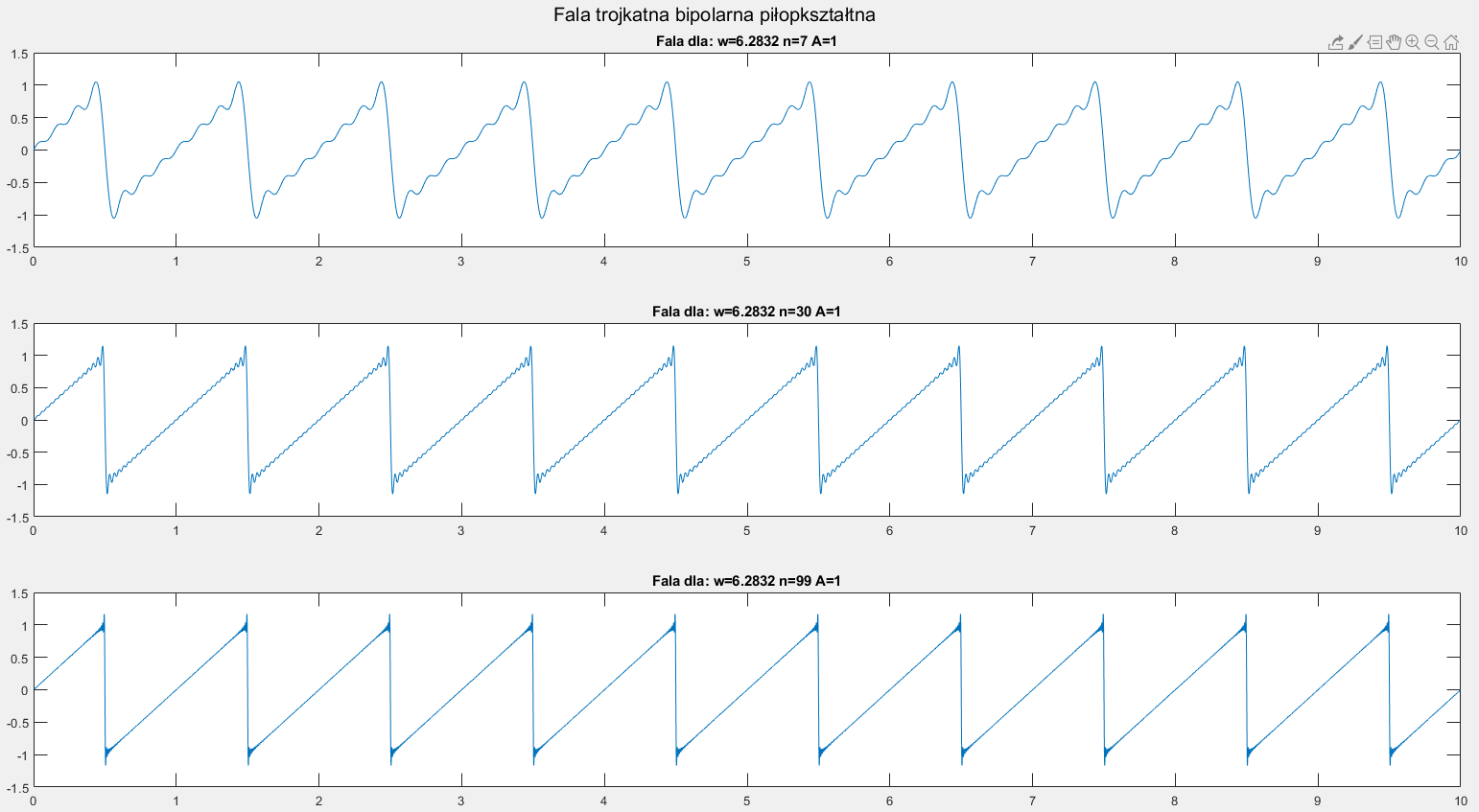
Tab. .. Wzory sygnałów używanych w programie.





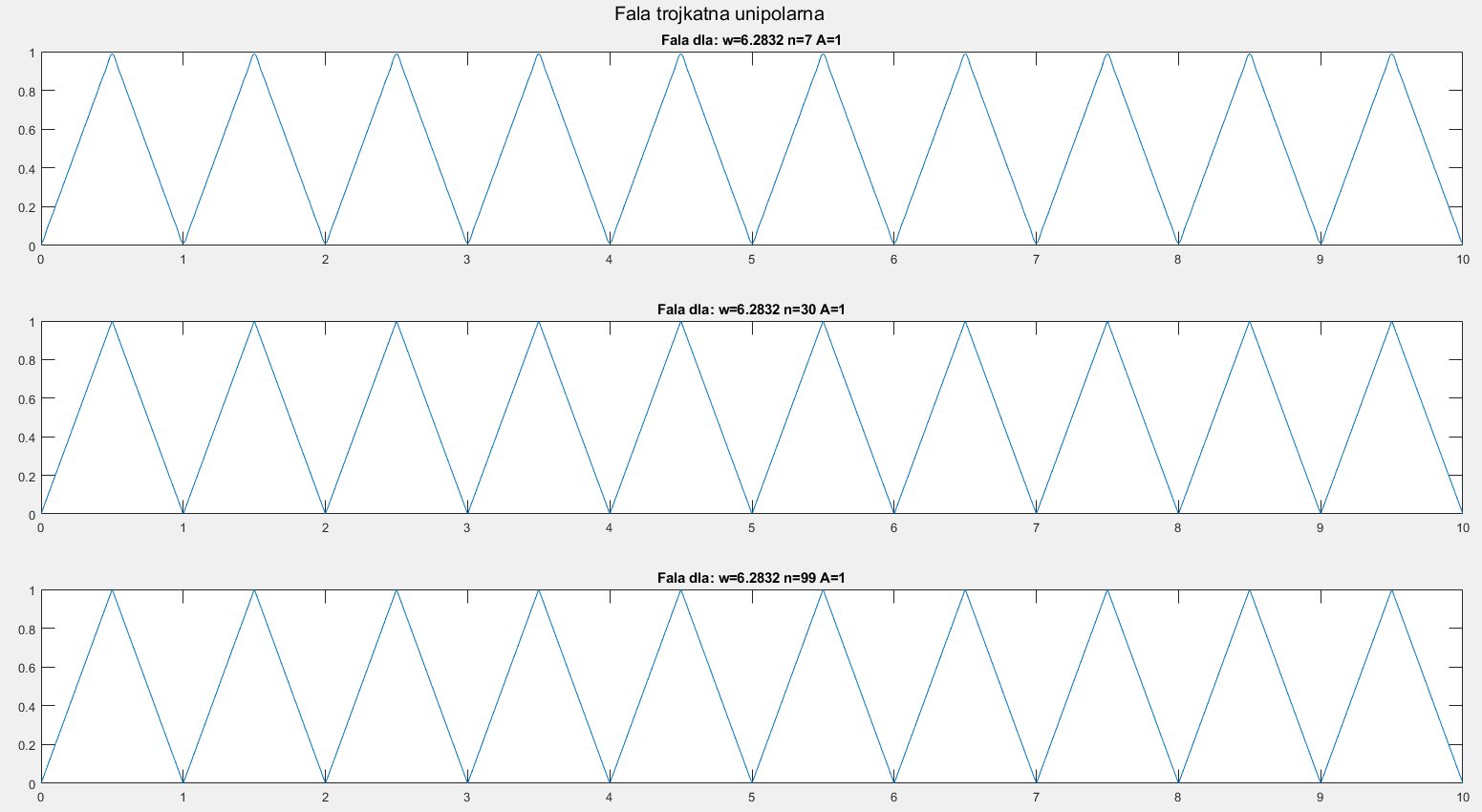
Rysunek .. Fala unipolarna o wypełnieniu dowolnym.

Rysunek .. Fala trójkątna bipolarna.

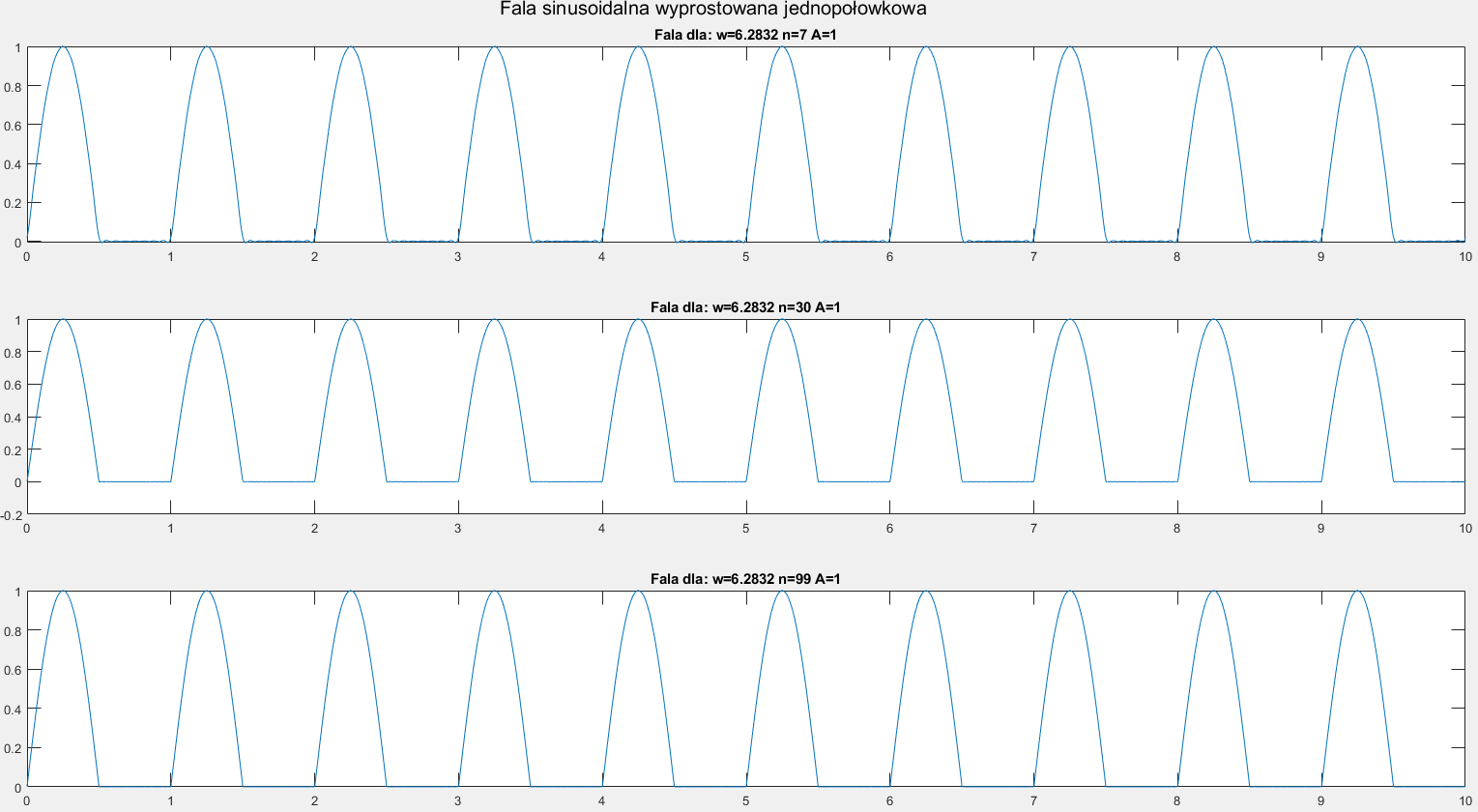


Rysunek .. Fala piłokształtna bipolarna.

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.

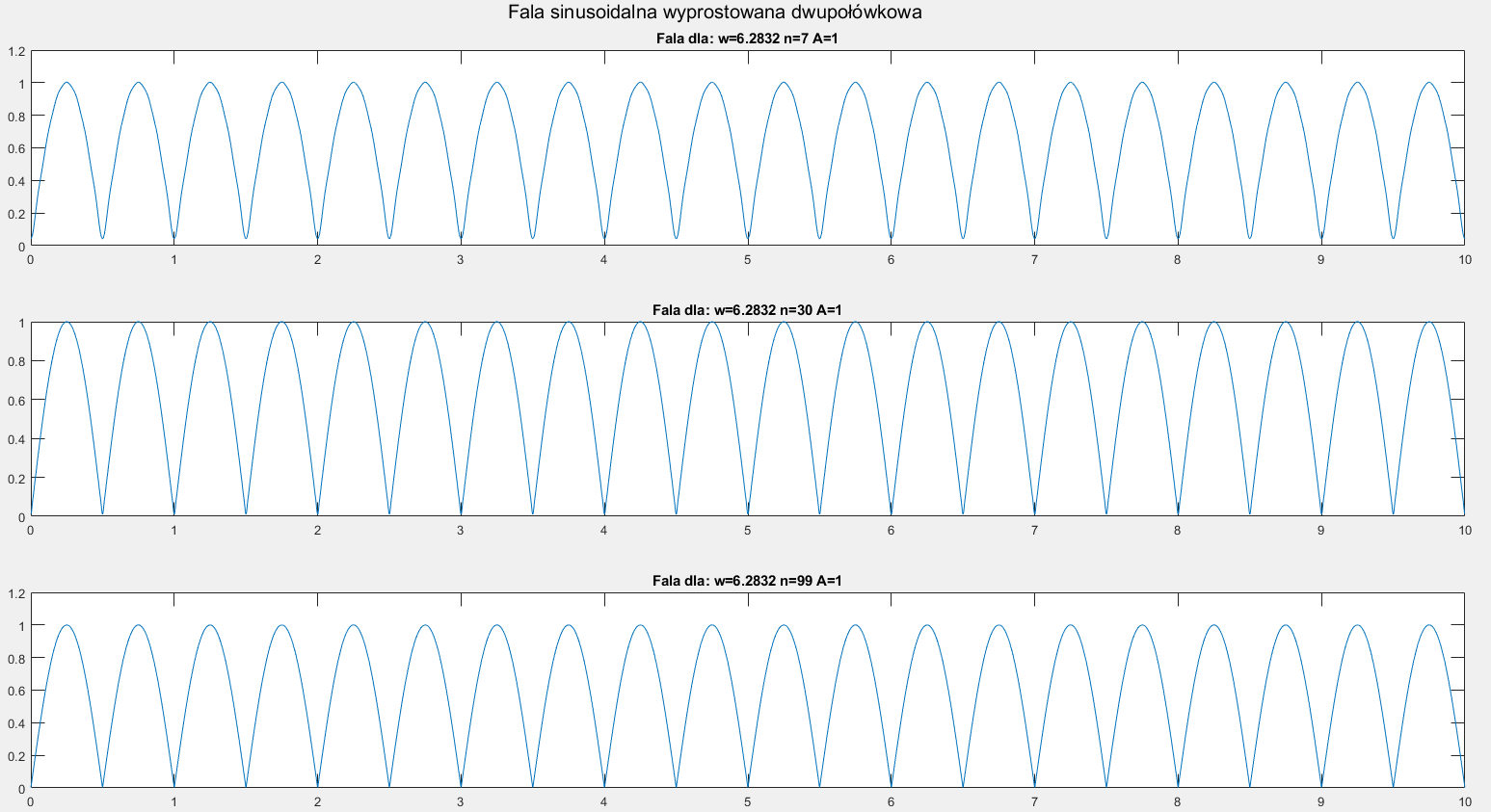


Rysunek .. Fala trójkątna unipolarna.

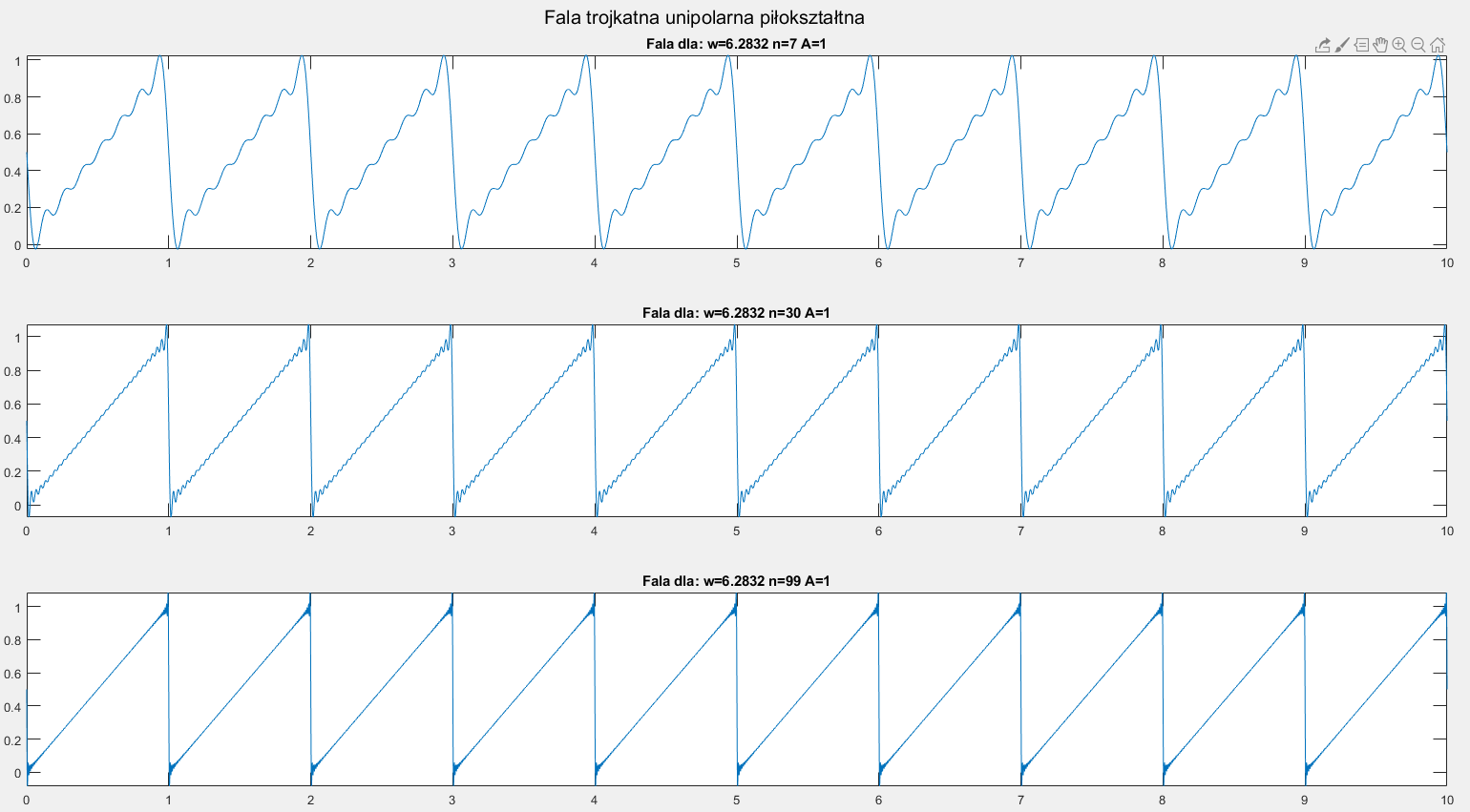


Rysunek .. Fala sinusoidalna jednopołówkowa, wyprostowana.

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



Rysunek .. Fala sinusoidalna dwupołówkowa, wyprostowana.



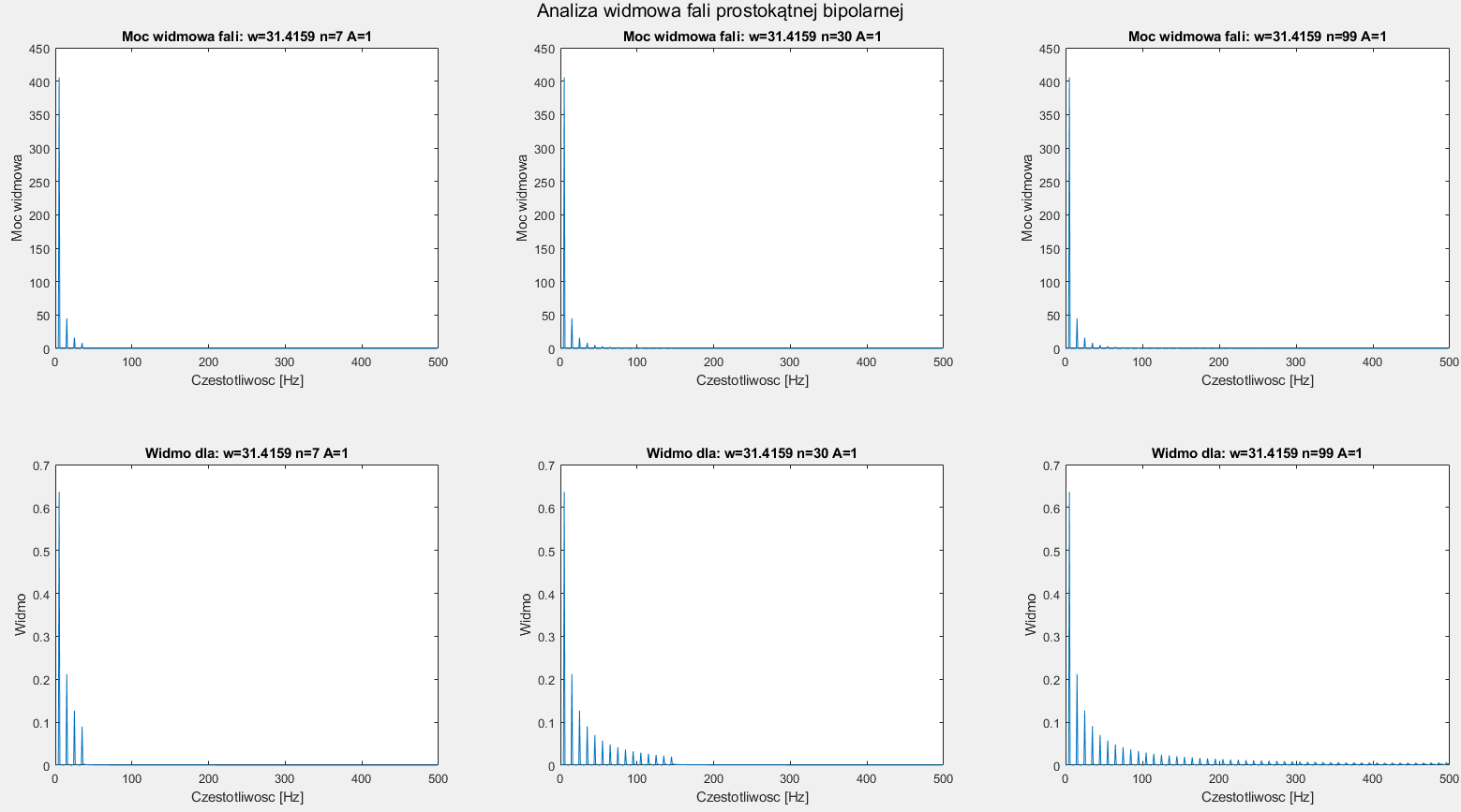
Rysunek .. Fala piłokształtna unipolarna.

# Laboratorium 5

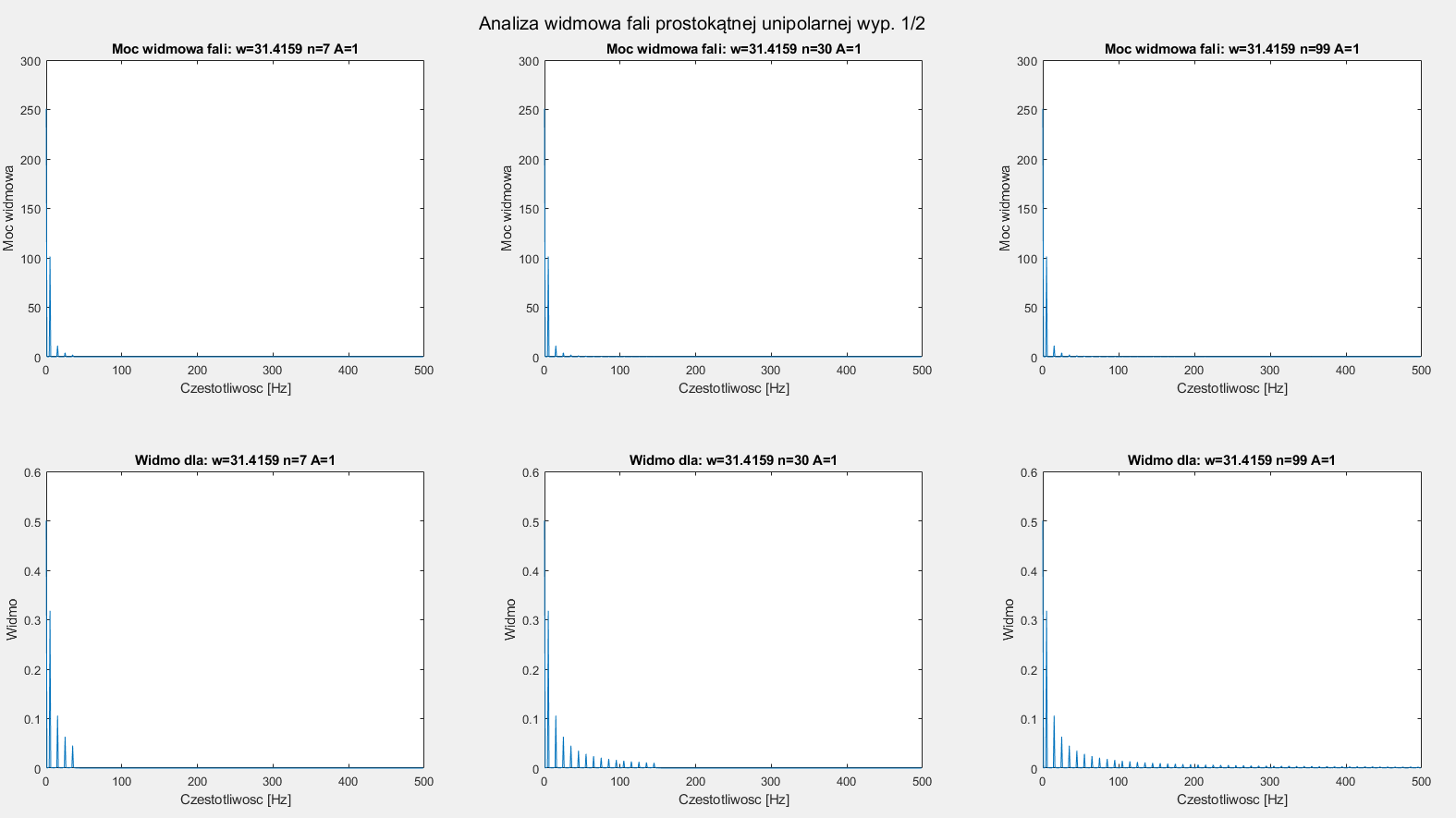
W ramach tego ćwiczenia zrealizowana została analiza częstotliwościowa fal obliczony i pokazanych w poprzednim rozdziale. Dodatkowo dokonana zostanie analiza układów RLC i RC. Stworzona została funkcja [f, M, W]=fft\_from\_signal(x, fs). Funkcja ta znajduje się na końcu pliku oraz w osobnym pliku tak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

|  |
| --- |
| % Lab 5. Procedura szybiej transfomraty Fouriera  % oraz analiza układów elektrycznych RC i RLC  % 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    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Czesc 2,  %% 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(19)  freqs(L,M) % analiza amplitudowo i fazowo - czestotliwosciowa  figure(20)  impulse(L,M) % odpowiedz impulsowa ukladu  figure(21)  step(L,M) % odpowiedz skokowa ukladu  figure(22)  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(23)  freqs(L,M) % analiza amplitudowo i fazowo - czestotliwosciowa  figure(24)  impulse(L,M) % odpowiedz impulsowa ukladu  figure(25)  step(L,M) % odpowiedz skokowa ukladu  figure(26)  iopzplot(sys) % wykres zer i biegunow dla ukladow wej/ wyj  [z,p,k]=tf2zp(L,M) % konwersja Transfer Function na zera i bieguny      %% 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 |

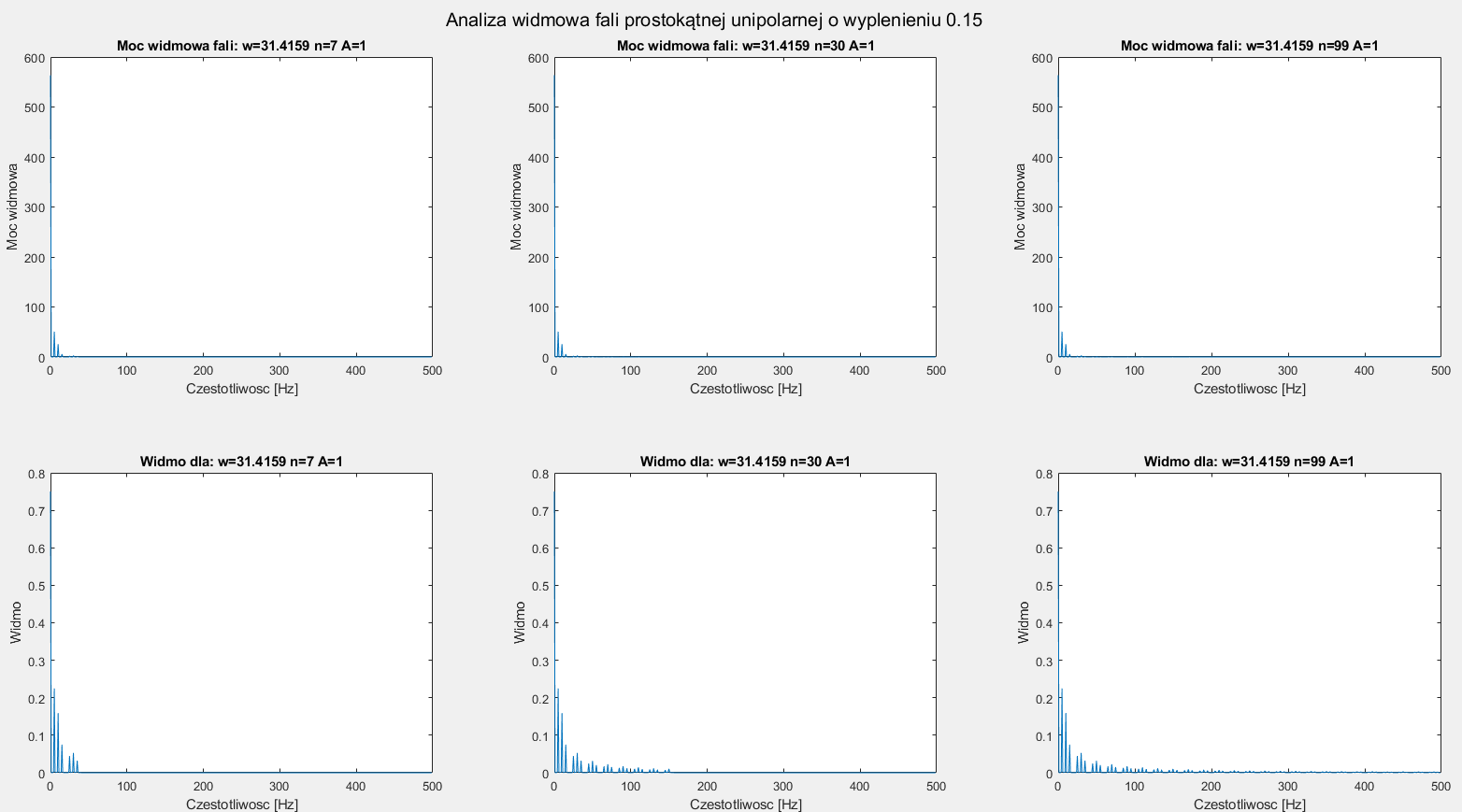


Rys. .. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej bipolarnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

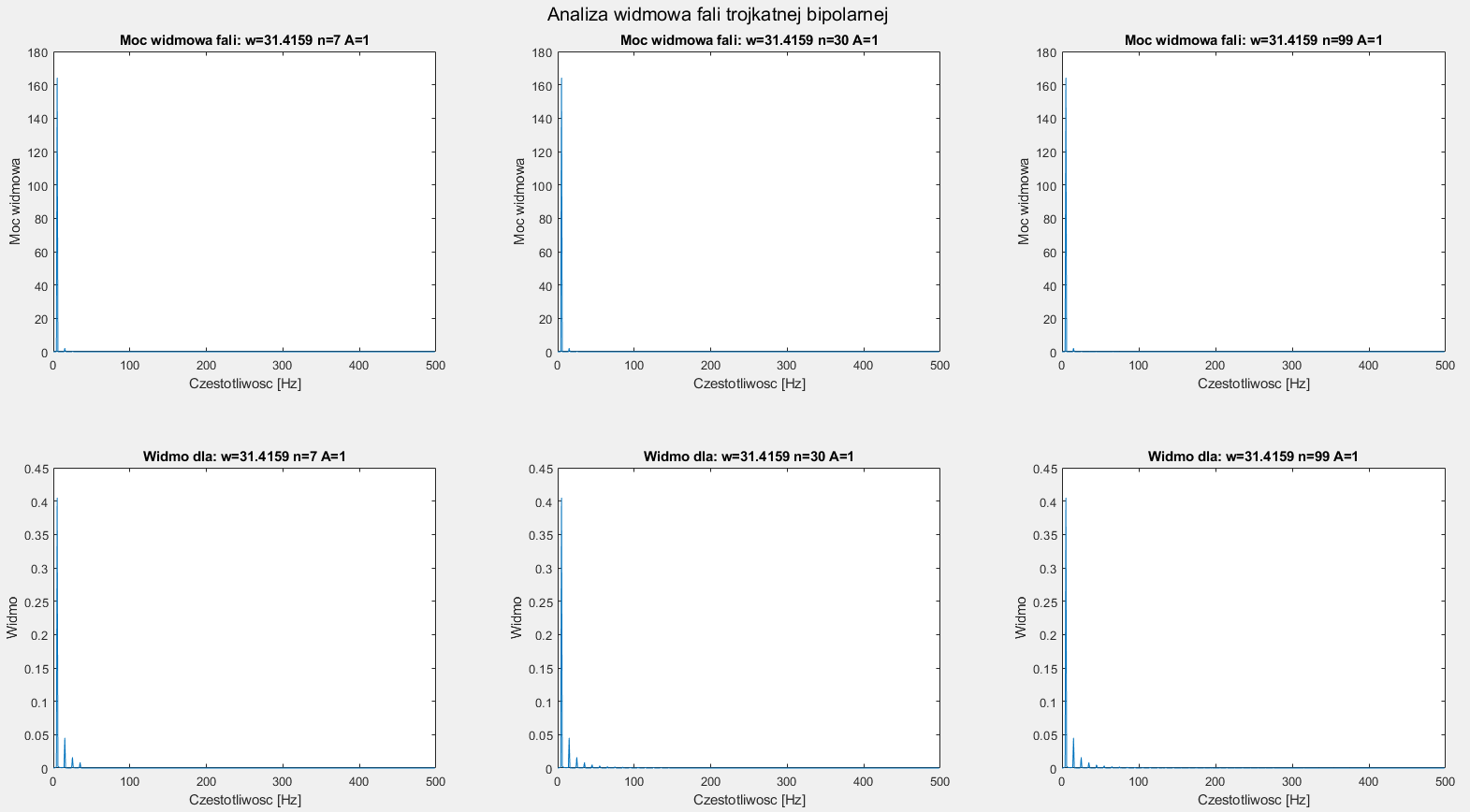


Rys. .. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej unipolarnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

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.

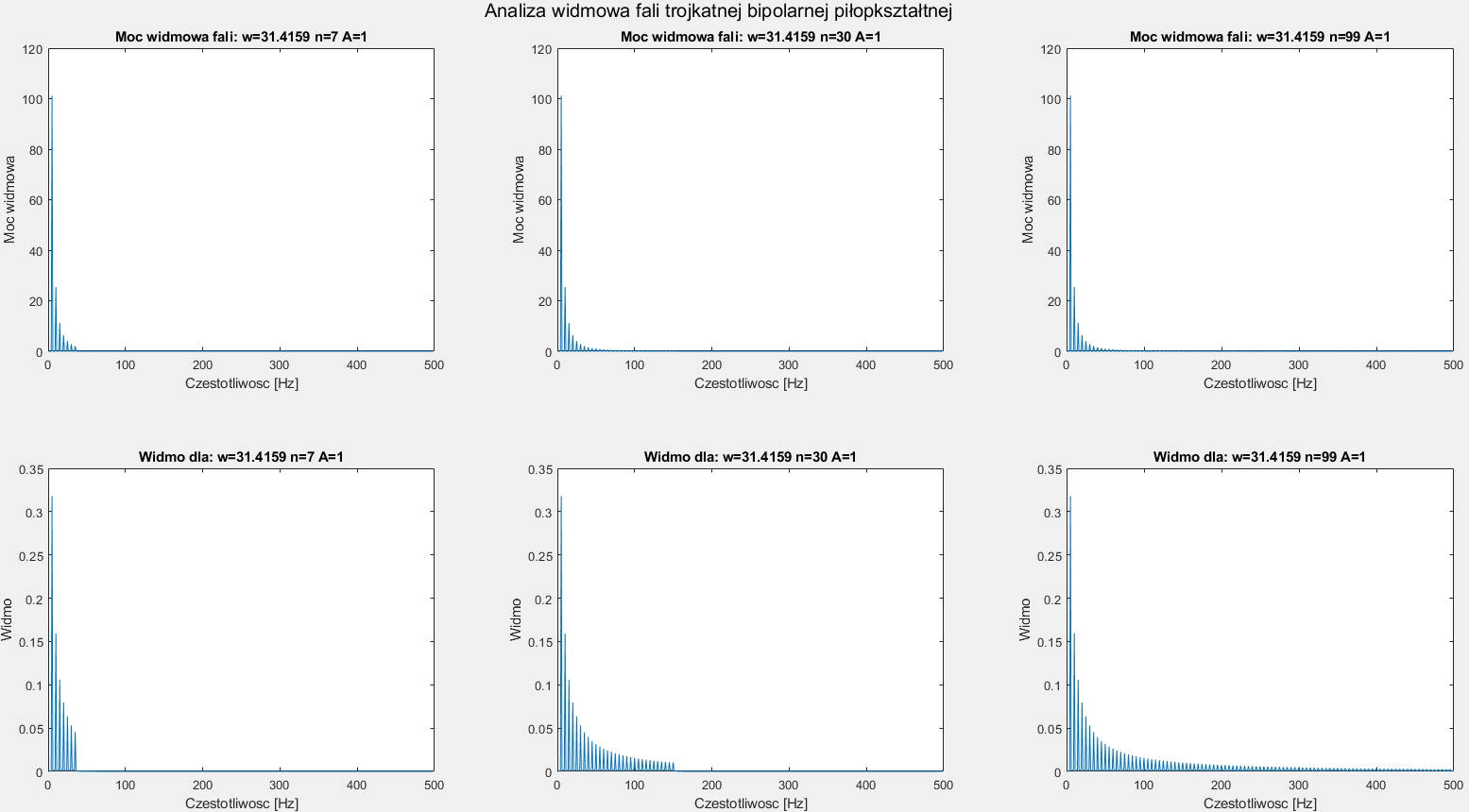


Rys. .. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali prostokątnej unipolarnej o wypełnieniu 0.15. Każda kolumna odpowiada większęmu rzędowi ciągu.

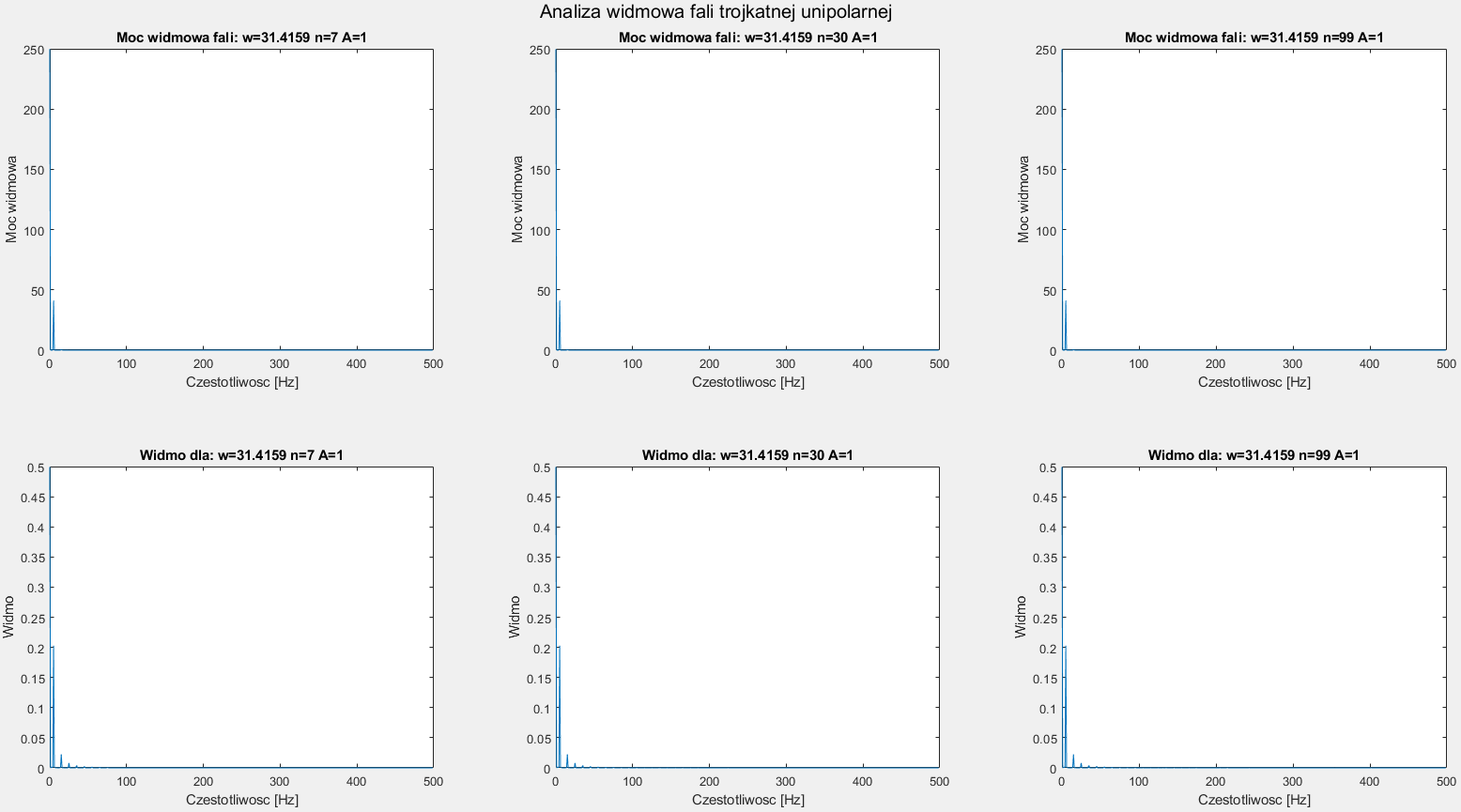


Rys. .. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej bipolarnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

Na wykresie 5.4. można zauważyć ze dla fali trójkątnej już pierwsze 2 składowe odpowiadają, za aproksymacje kształtu. Kolejne składowe stanowią kosmetykę załamań fali. W przypadku rysunku 5.3. widoczne jest, że kolejne prążki widma nie maleją wykładniczo jak w przypadku rysunku. 5.2. Pojawiają się pewne oscylacje.

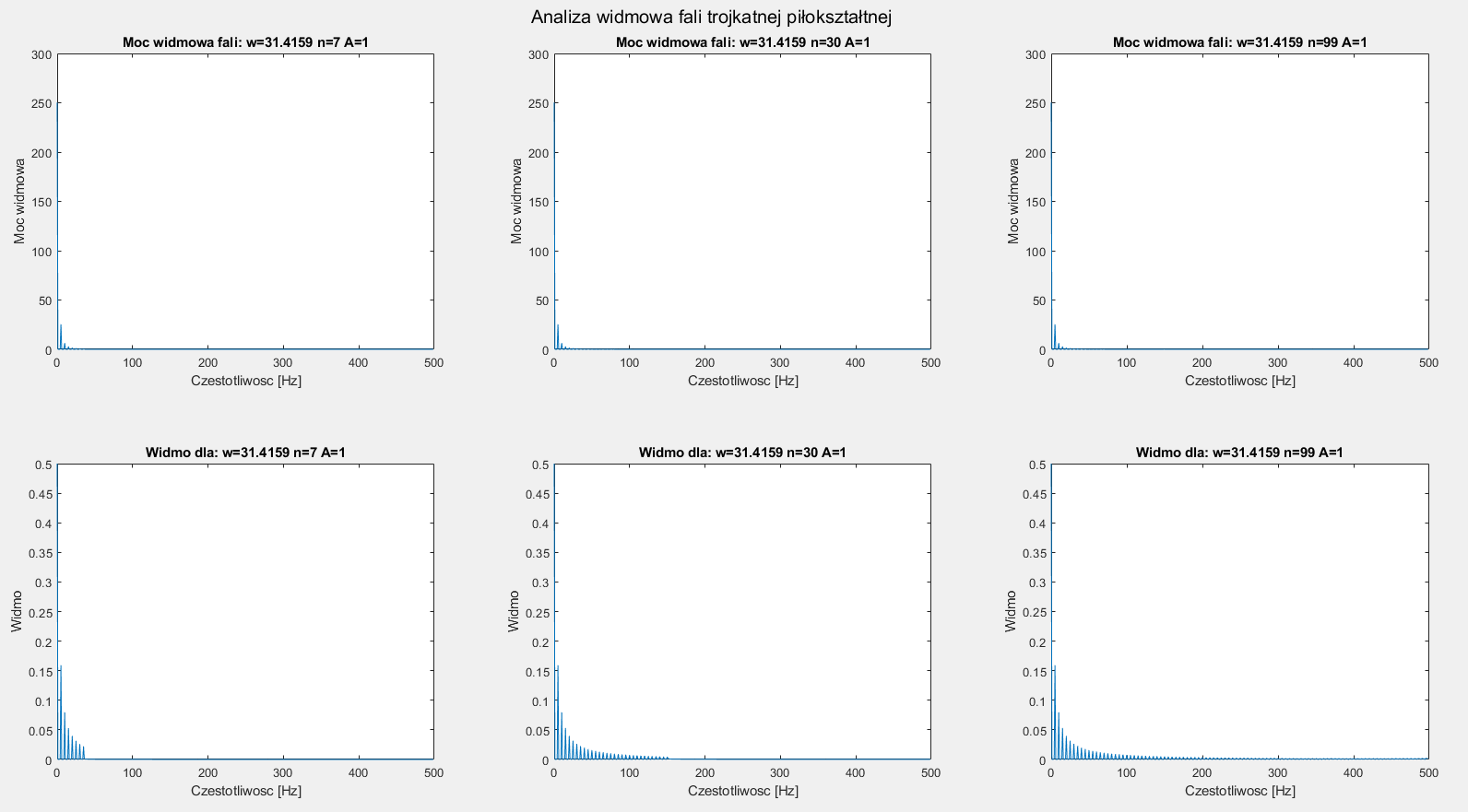


Rys. .. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej bipolarnej piłokształtnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

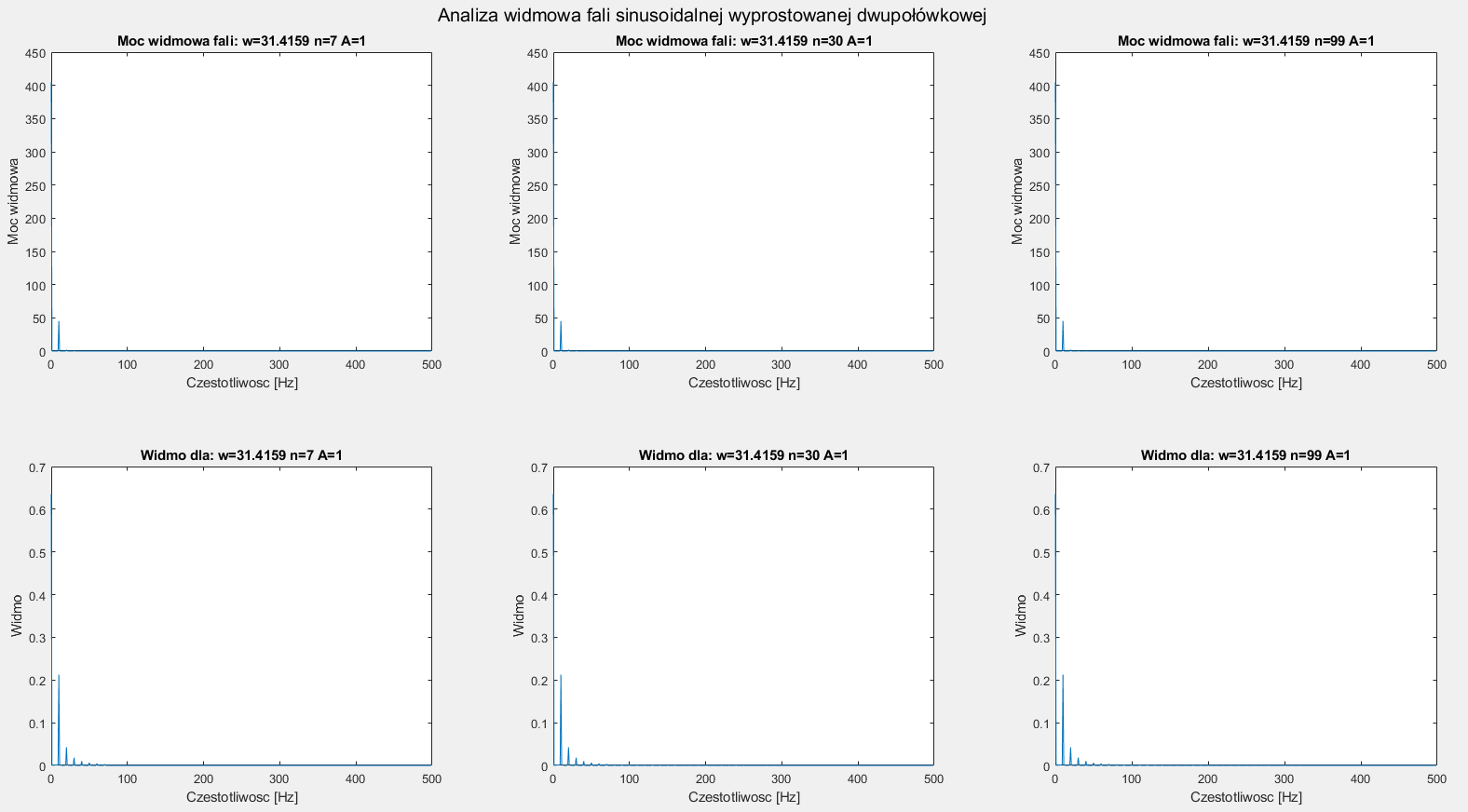


Rys. .. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej unipolarnej. Każda kolumna odpowiada większęmu rzędowi ciągu.

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.

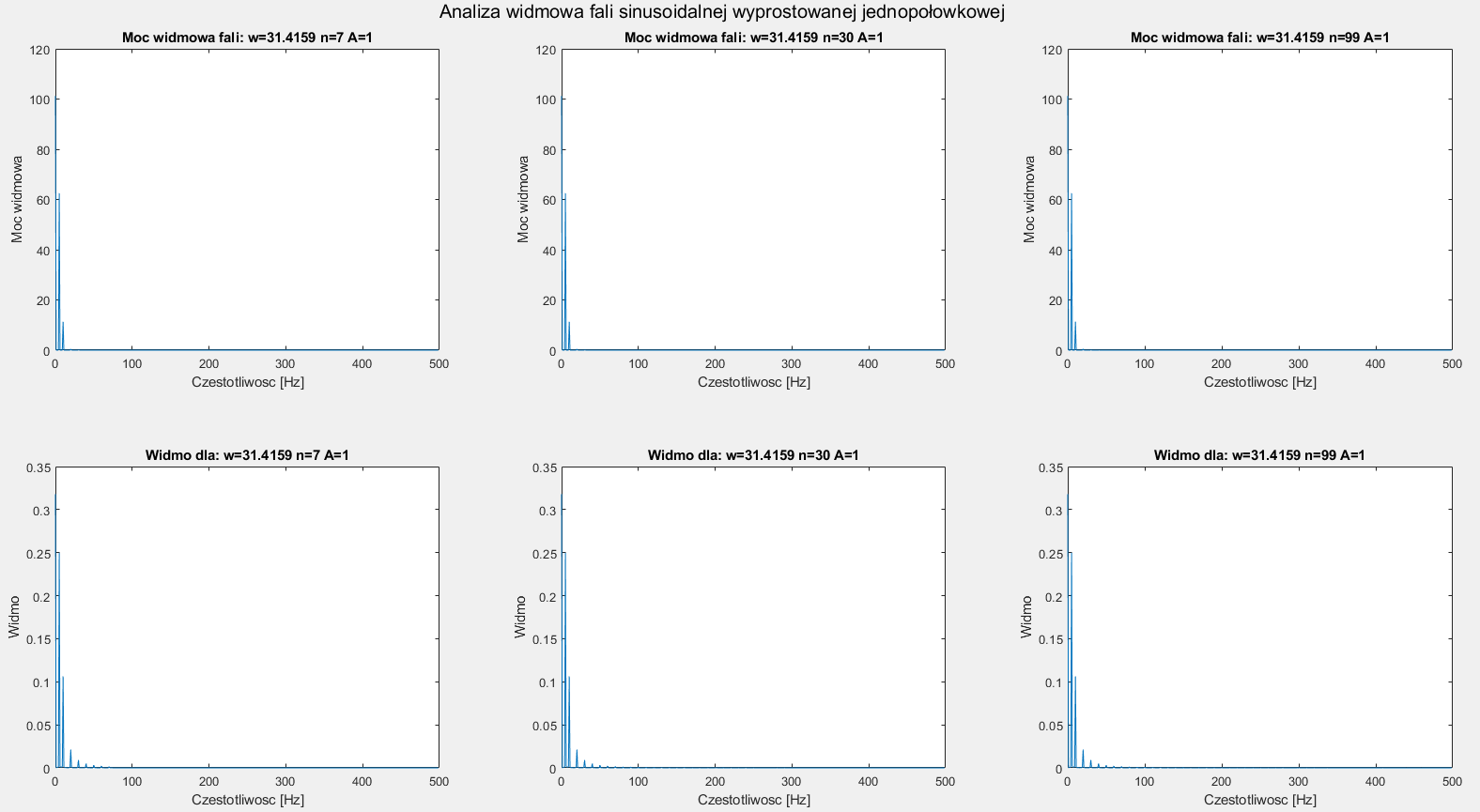


Rys. .. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali trójkątnej unipolarnej piłokształtnej. Każda kolumna odpowiada większęmu rzędowi ciągu.



Rys. .. 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.

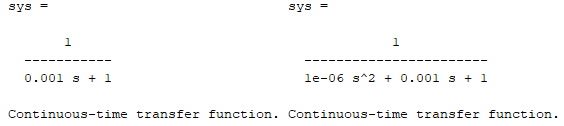


Rys. .. Moc widmowa (1 rząd wykresów) i widmo (2 rząd wykresów) dla fali sinusoidalnej jednopołówkowej wyprostowanej. Każda kolumna odpowiada większęmu rzędowi ciągu.

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

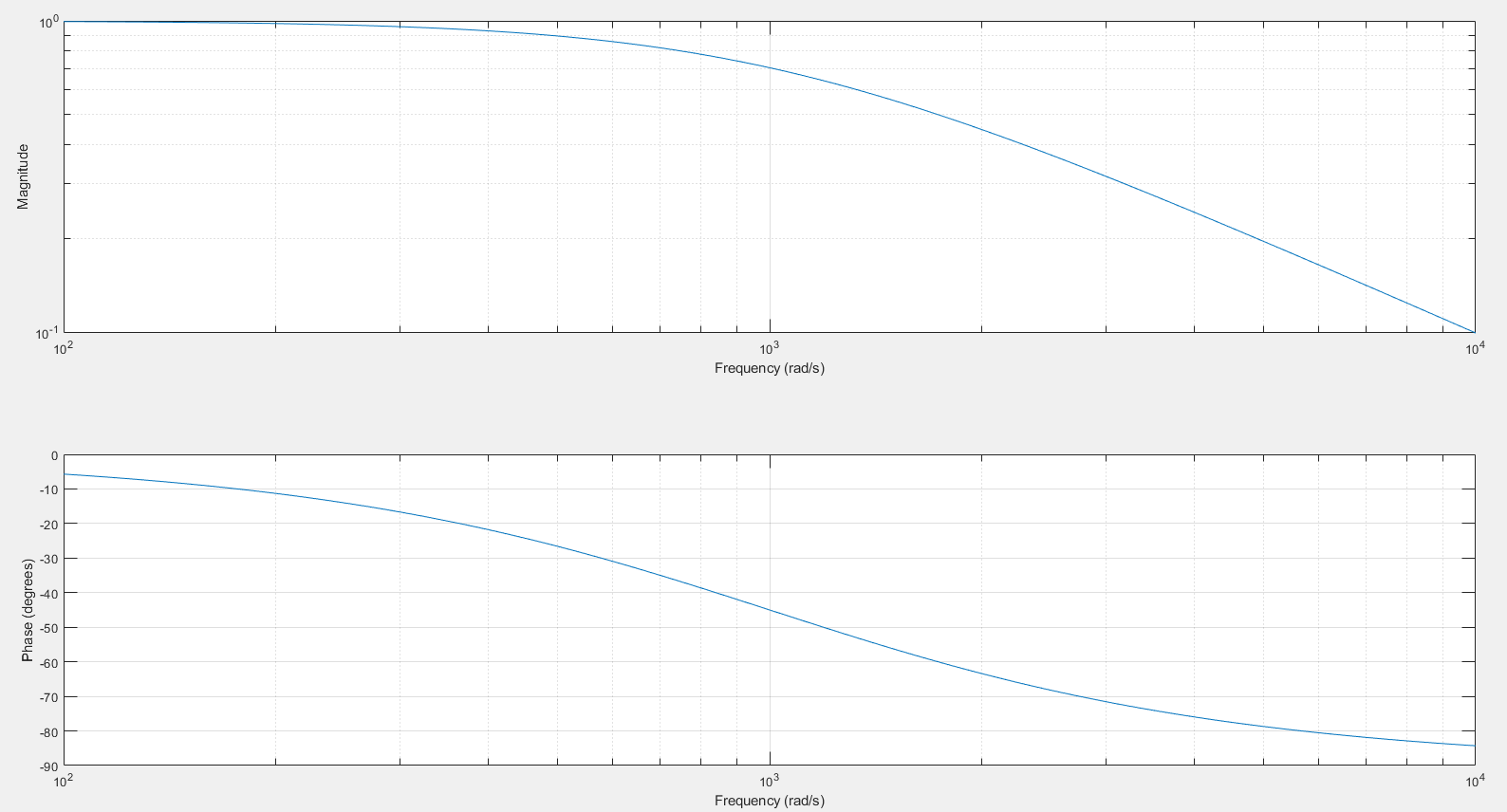
Przyjęto parametry dla układu RC: R=1000[Ω], C=10-6[F] oraz dla układu RLC: : R=1000[Ω], C=10-6[F], L=1[H]. Za pomocą tych wartości zdefiniowane zostały liczniki i mianowniki transmitancji układów danymi wzorami poniżej.

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.

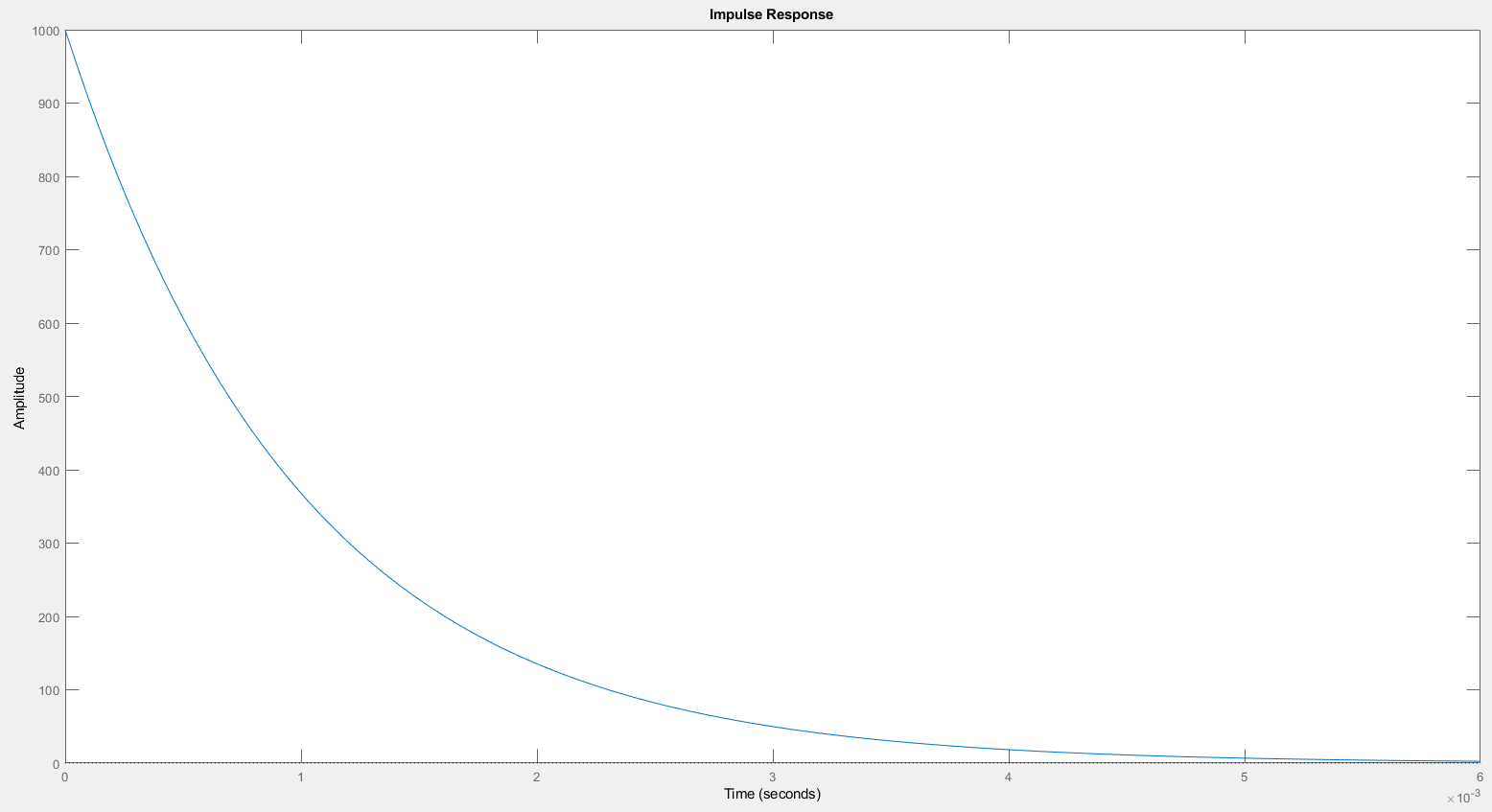


Rys. .. Wynik z funkcji tf(L, M).

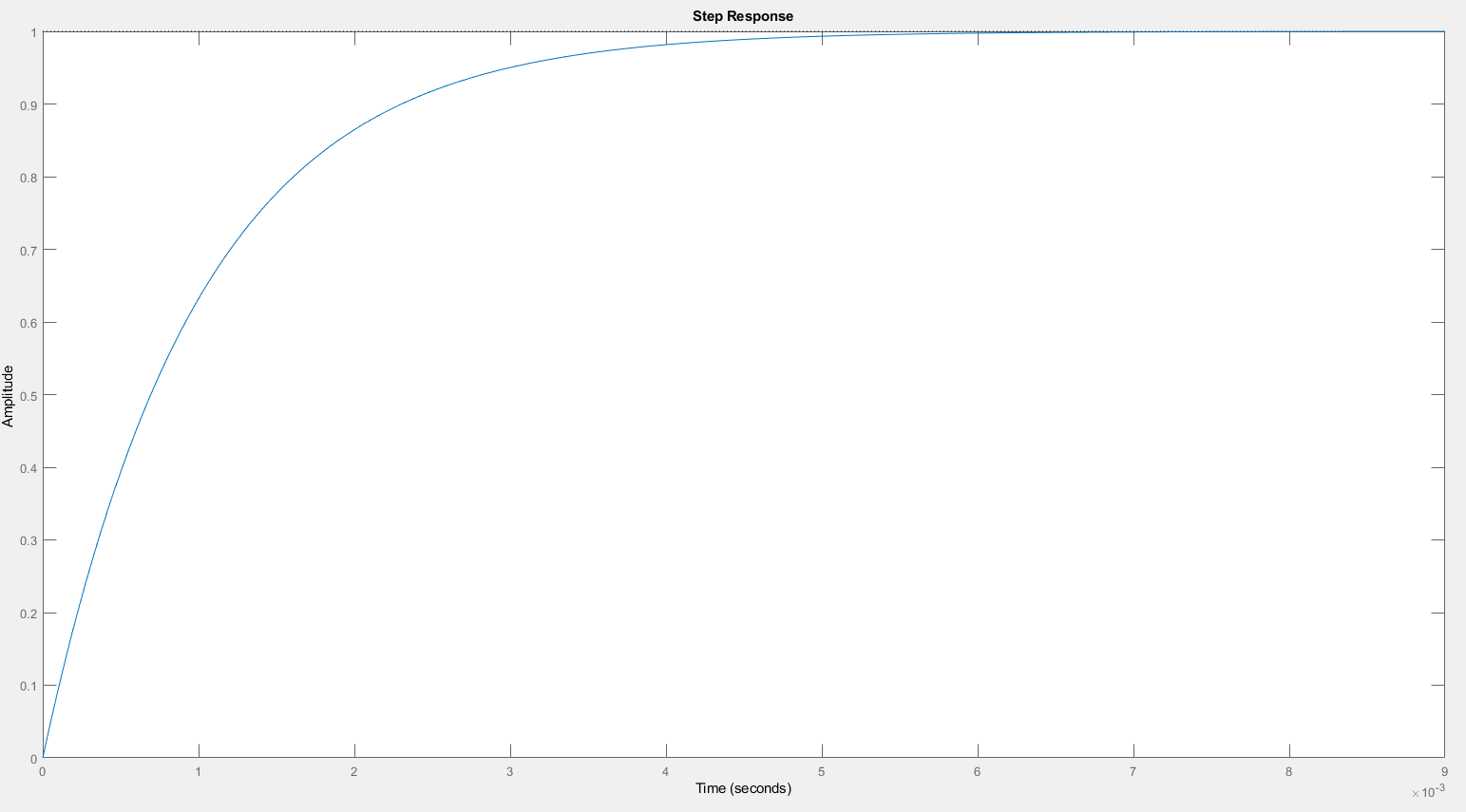
Następnie dla stworzonych modeli wyznaczone zostały charakterystyki częstotliwościowe i czasowe za pomocą funkcji freqs(L, M), step(L, M), impulse(L, M). Są to funkcje zwracające charakterystyki Bodego (amplitudowo i fazowo-częstotliwościowe), skokową i impulsową. Ostatnim etapem było wykreślenie zer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



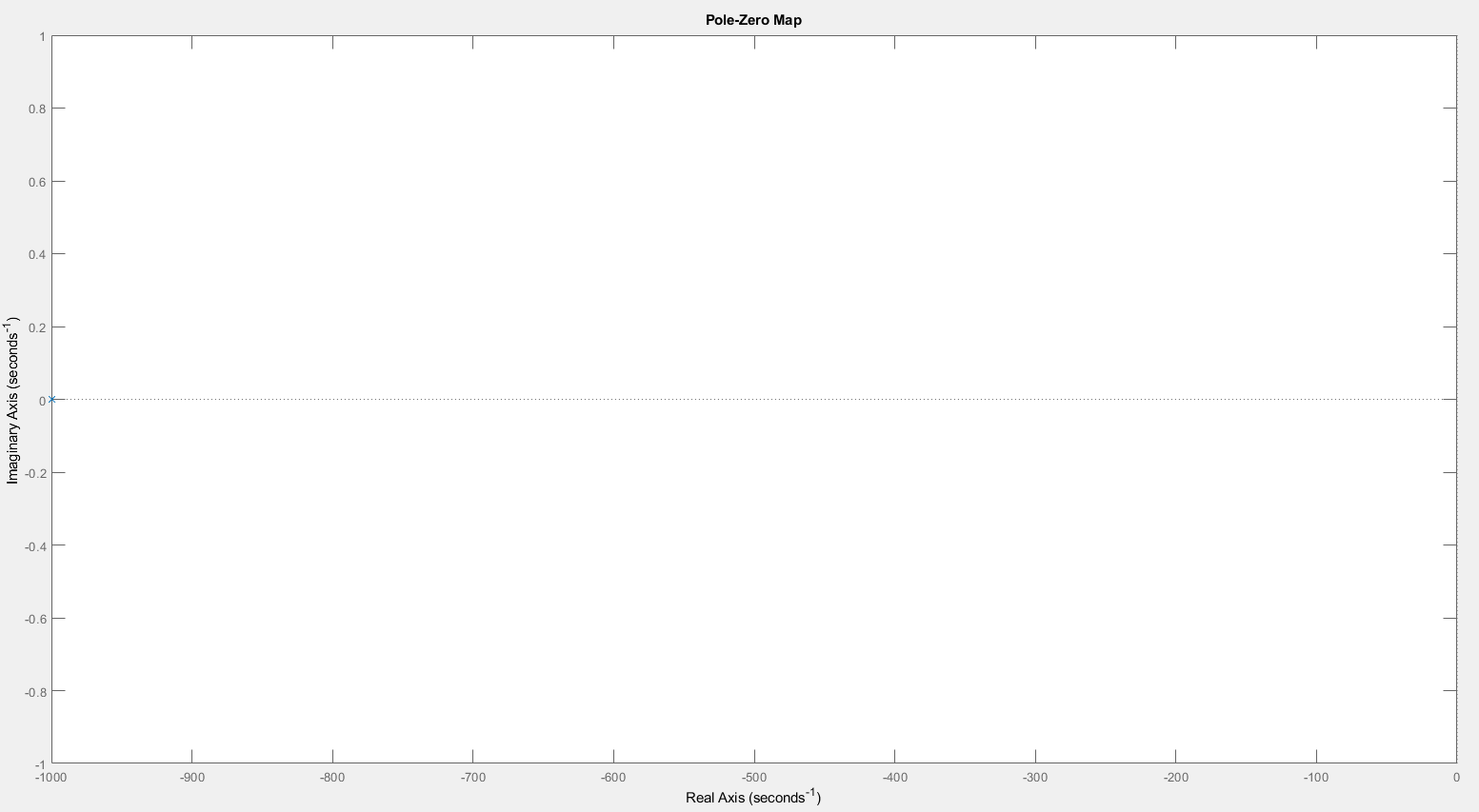
Rys. .. Charakterystyki Bodego dla układu RC.



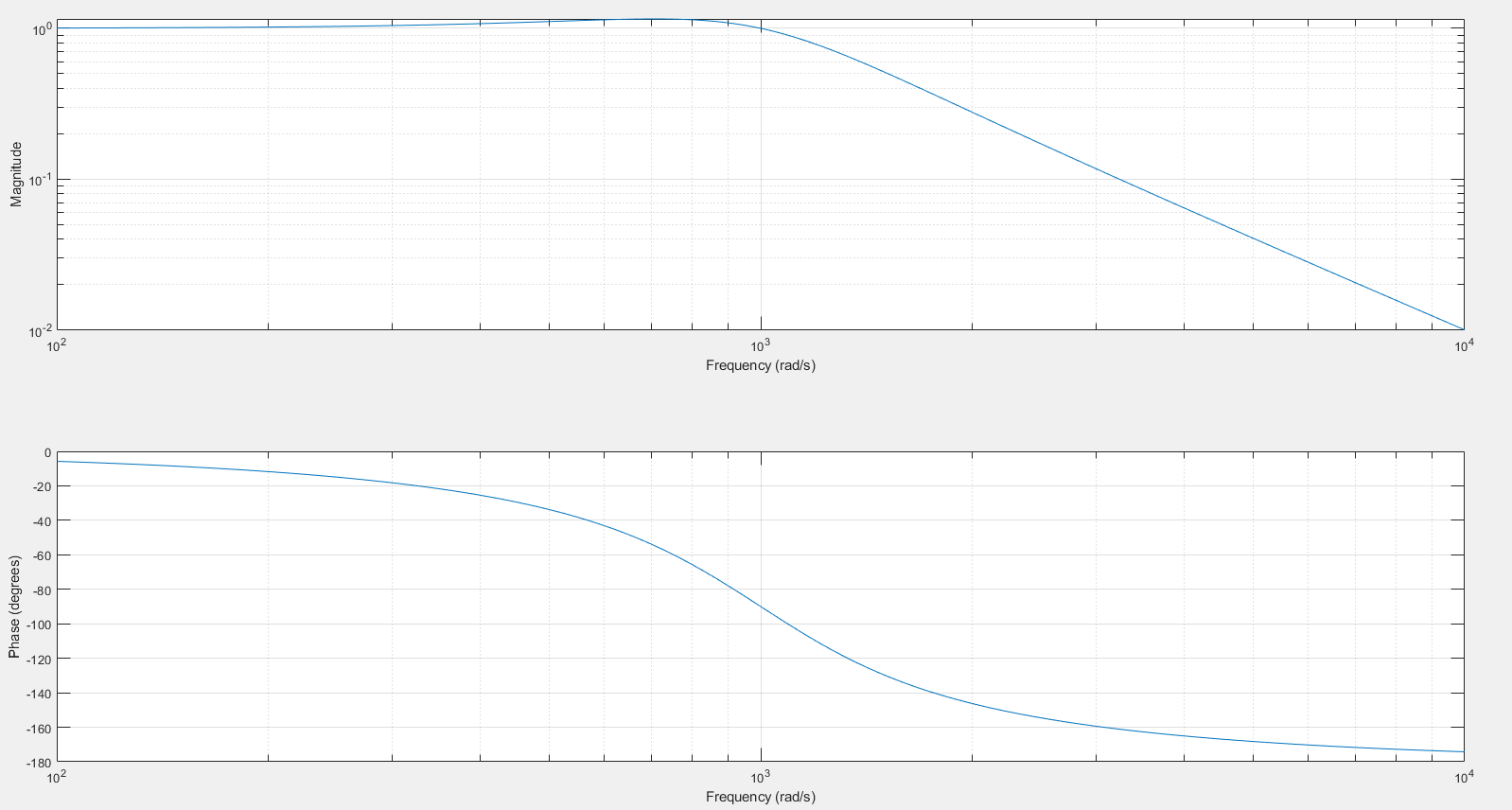
Rys. .. Odpowiedź impulsowa układu RC.



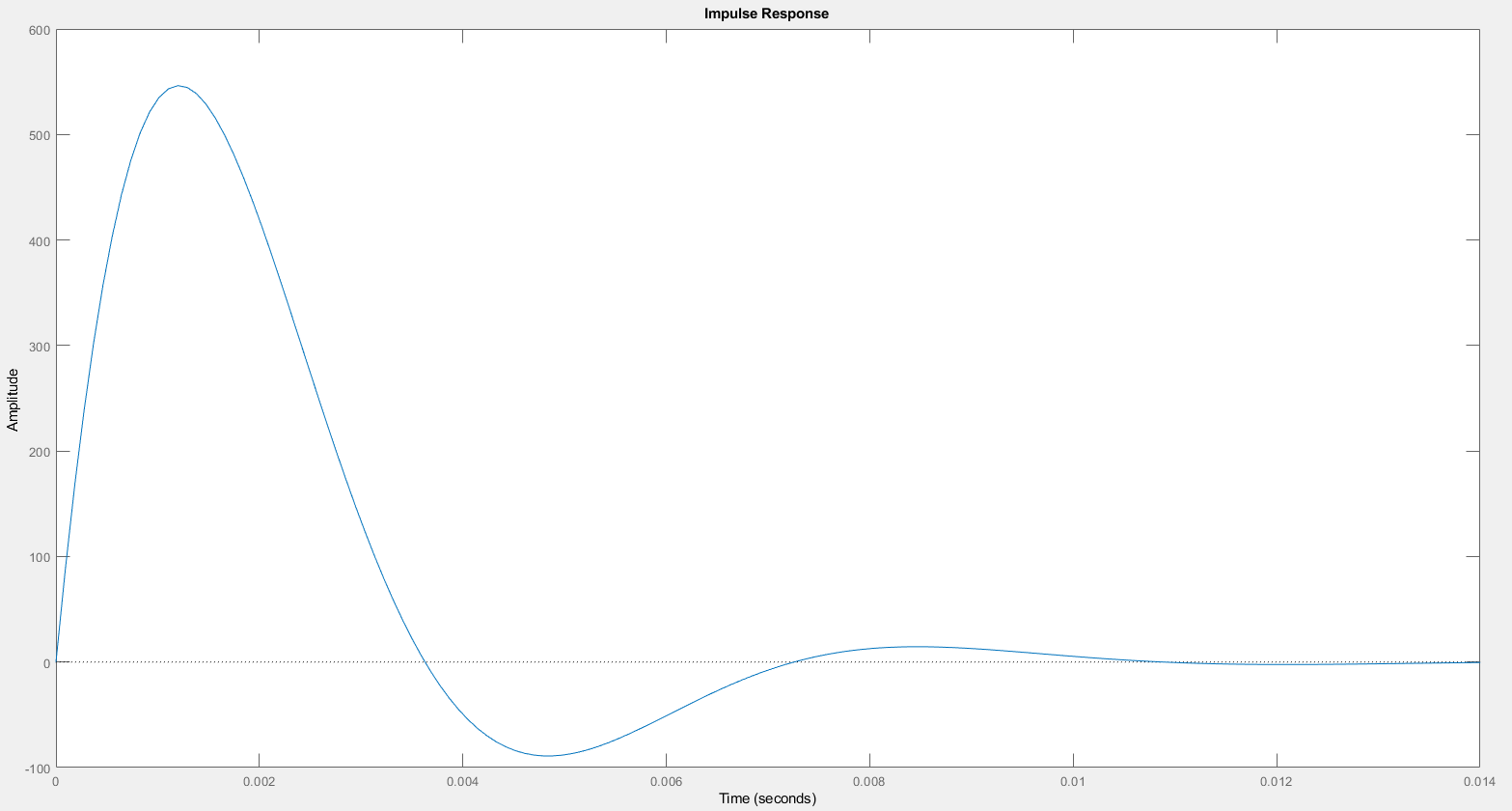
Rys. .. Odpowiedź skokowa układu RC.



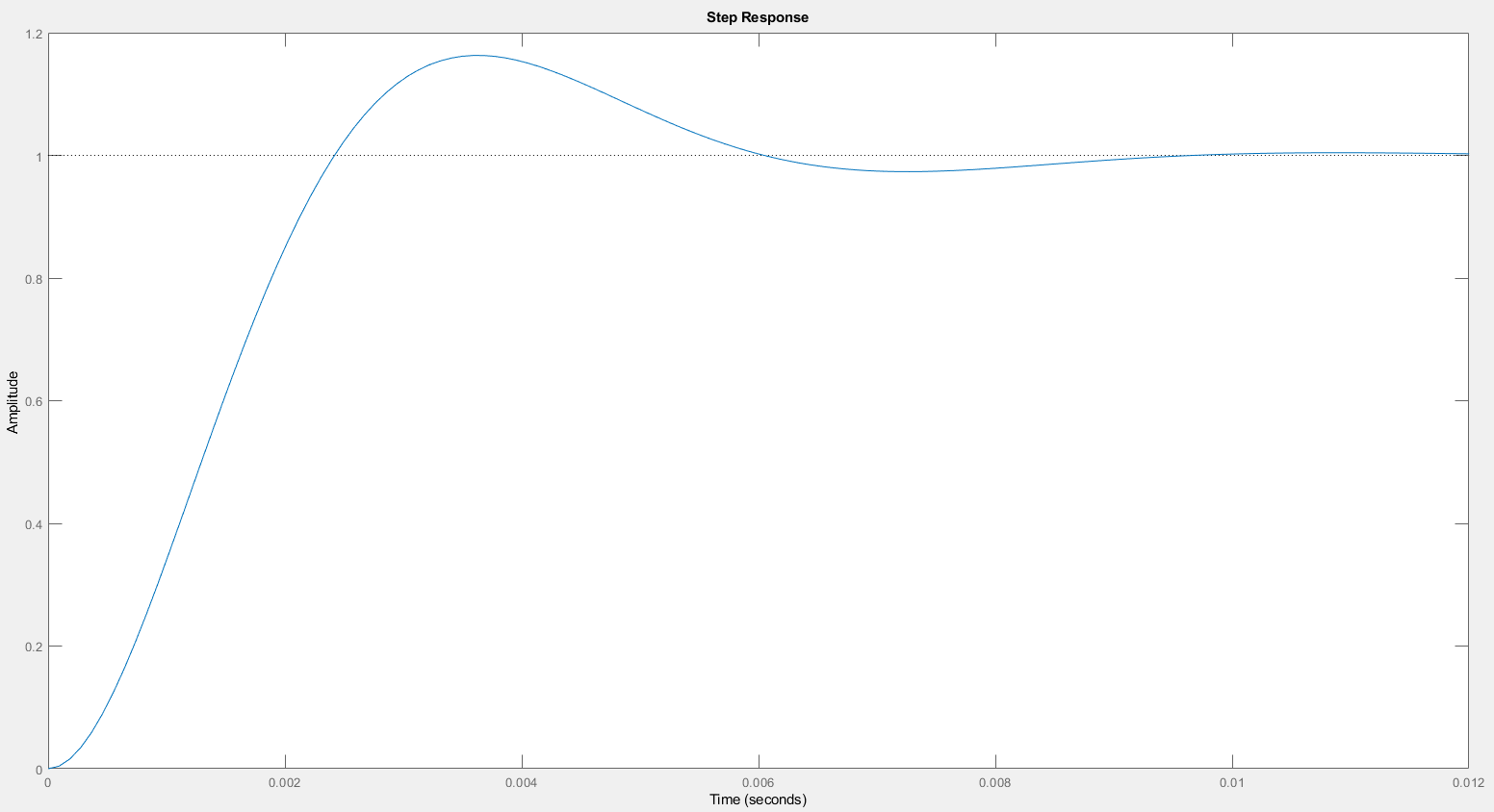
Rys. .. Biegun układu RC (prawie w zerze).



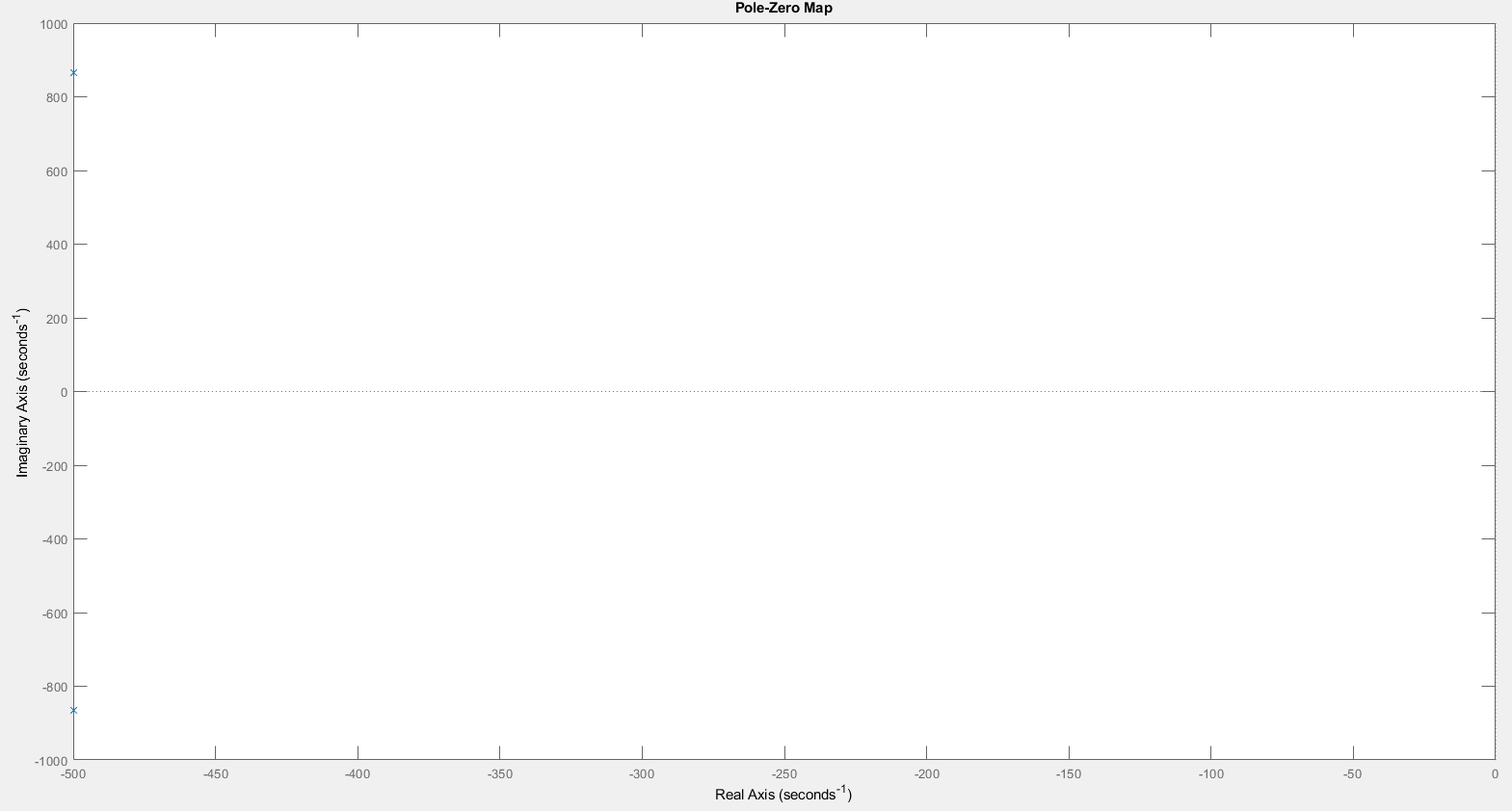
Rys. .. Charakterystyki Bodego dla układu RLC.



Rys. .. Odpowiedź impulsowa układu RLC.



Rys. .. Odpowiedź skokowa układu RLC.



Rys. .. 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.