Języki Modelowania i Symulacji

Podstawowe narzędzia przetwarzania danych w MATLABIE

Marcin Ciołek

Katedra Systemów Automatyki
WETI, Politechnika Gdańska

18 października 2011

Literatura:

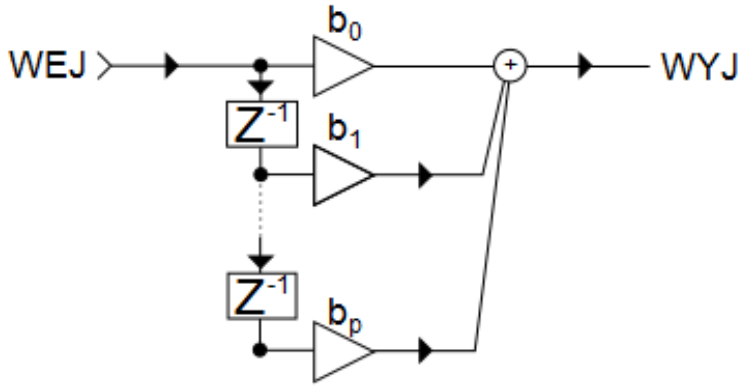
1. D. Kincaid, W. Cheney: „Analiza numeryczna”, Wydawnictwo Naukowo Techniczne, 2006.
2. P. Davis, W.: „Differential Equations - Modelling with MATLAB”, Prentice Hall, 1999.
3. Dokumentacja MATLABA i SIMULINKA.
4. B.Mrozek, Z. Mrozek: „MATLAB Uniwersalne środowisko do obliczeń naukowo-technicznych”, Kraków 1995.
5. T.P. Zieliński: „Cyfrowe przetwarzanie sygnałów - Od teori do zastosowań”, Warszawa 2009.

O czym będziemy dziś mówili?

Filtr FIR

Jak filtrujemy sygnał?

Filtr FIR (z ang. „finite impulse response”)



„filtr FIR”

$$H(z^{-1}) = b_0 + b_1 z^{-1} + \cdots + b_n z^{-n}$$

Filtr FIR

Własności filtrów FIR:

Zalety:

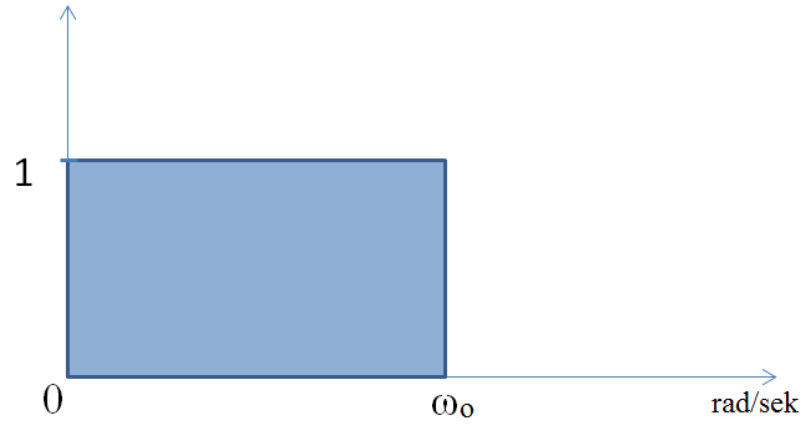
- liniowy przebieg fazy
- filtr stabilny
- względnie łatwe procedury projektowania filtrów
- łatwość implemaentacji hardware'owej
- wpływ warunków początkowych jest zawsze skończony

Wady:

- duży wymagany rząd filtru (większy niż odpowiednich filtrów IIR)
- większe opóźnienie w torach sygnałowych

idealny filr

Idelana charakterystyka amplitudowa filtru



odpowiedź impulsowa tego filtru

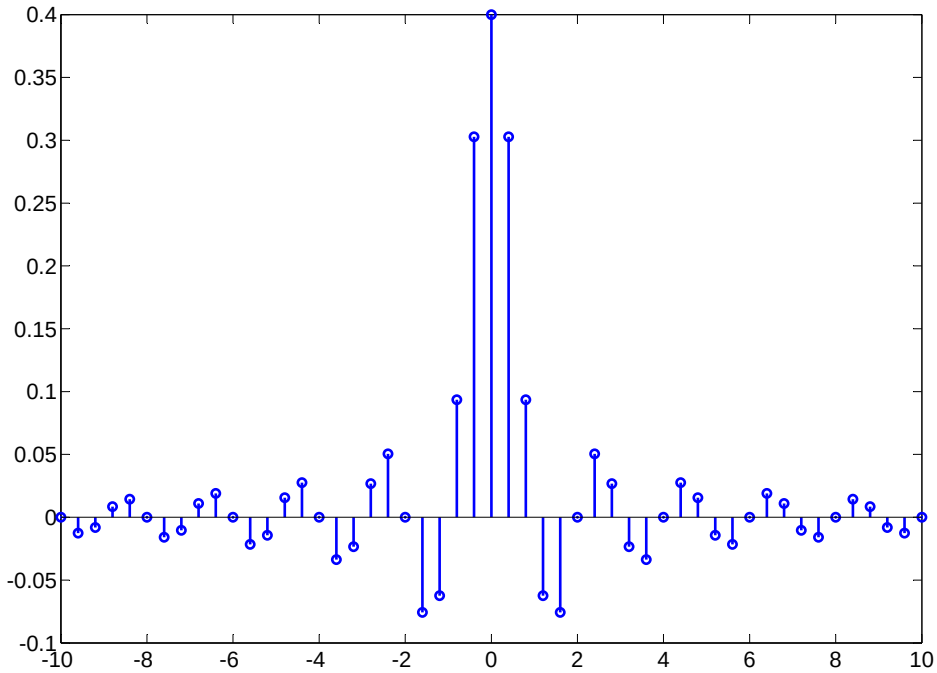
$$h(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)e^{j\omega n}d\omega = \frac{1}{2\pi}\int_{-\omega_o}^{\omega_o} e^{j\omega n}d\omega =$$

$$= \frac{\omega_o}{\pi}\mathrm{sinc}(\frac{\omega_o n}{\pi})$$

Co powiesz o odpowiedzi impulsowej idealnego filtru FIR?

sinc

Idelana charakterystyka amplitudowa filtru obcięta za pomocą okna prostokątnego

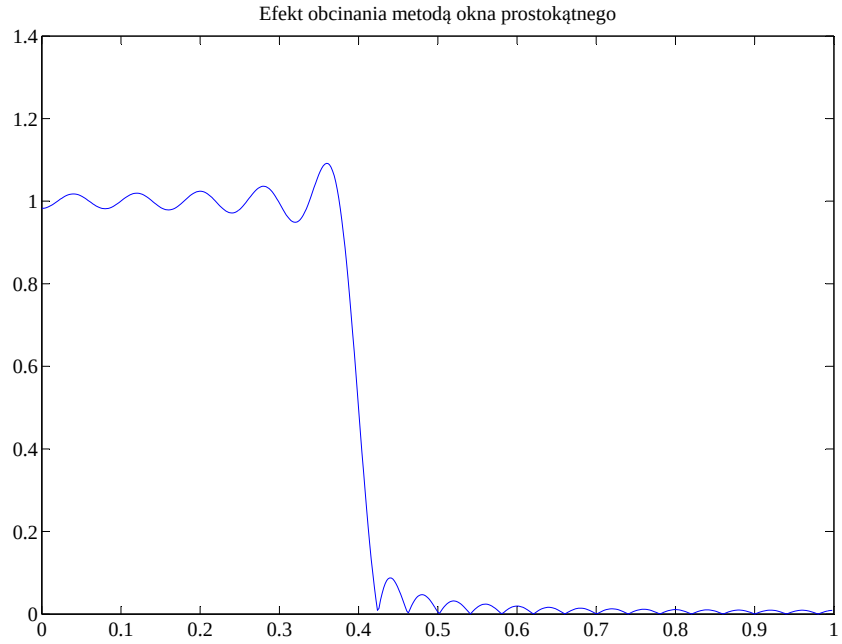


Ile wynosi częstotliwość odcięcia?

```
b = 0.4*sinc(0.4.*(-25:25));
stem(0.4.*(-25:25),b)
```

efekt Gibbsa

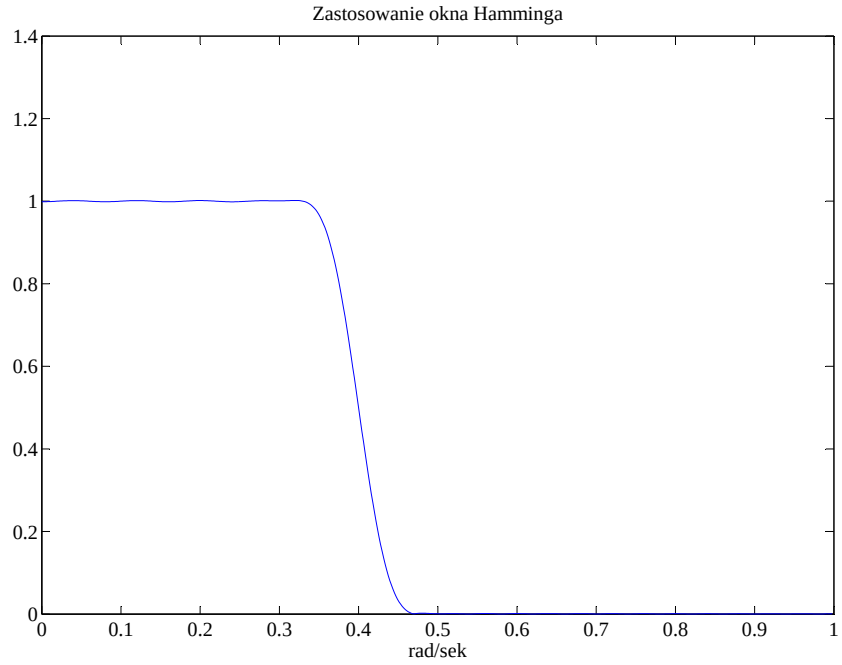
Jak można ograniczyć ten efekt?




```
Fp = 2; %Hz
n  = 512;
[H,w]=freqz(b,1,n,Fp);
plot(w,abs(H))
set(gca,'FontName', 'Times New Roman CE', 'FontSize',16)
title('Efekt obcinania metodą okna prostokątnego')
xlabel('rad/sek')
```

hamming

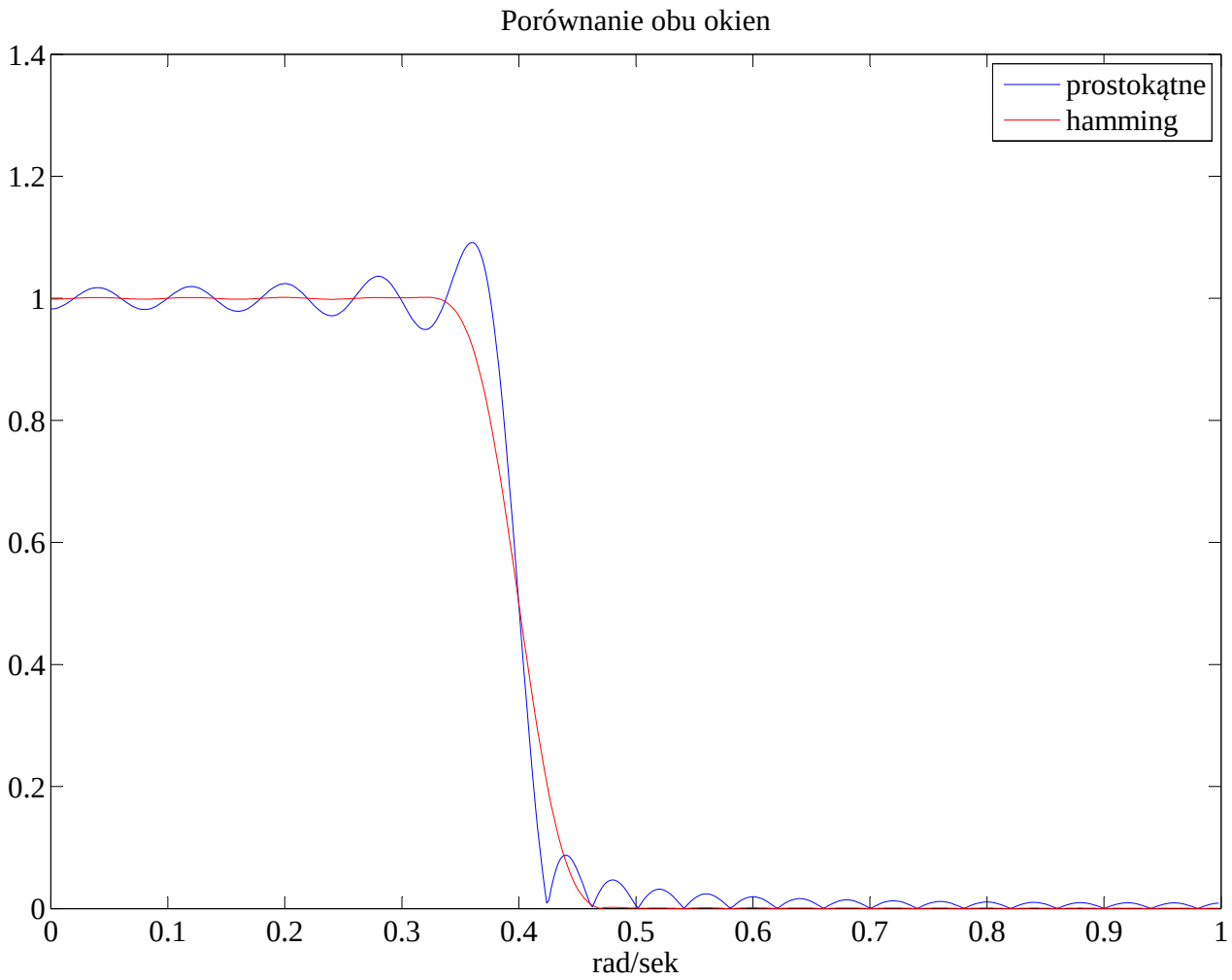
Co się poprawiło, a co się pogorszyło?




```
Fp = 2; %Hz
n  = 512;
b=b.*hamming(51)';
[H,w]=freqz(b,1,512,2);
plot(w,abs(H))
set(gca,'FontName', 'Times New Roman CE', 'FontSize',16)
title('Zastosowanie okna Hamminga')   xlabel('rad/sek')
```

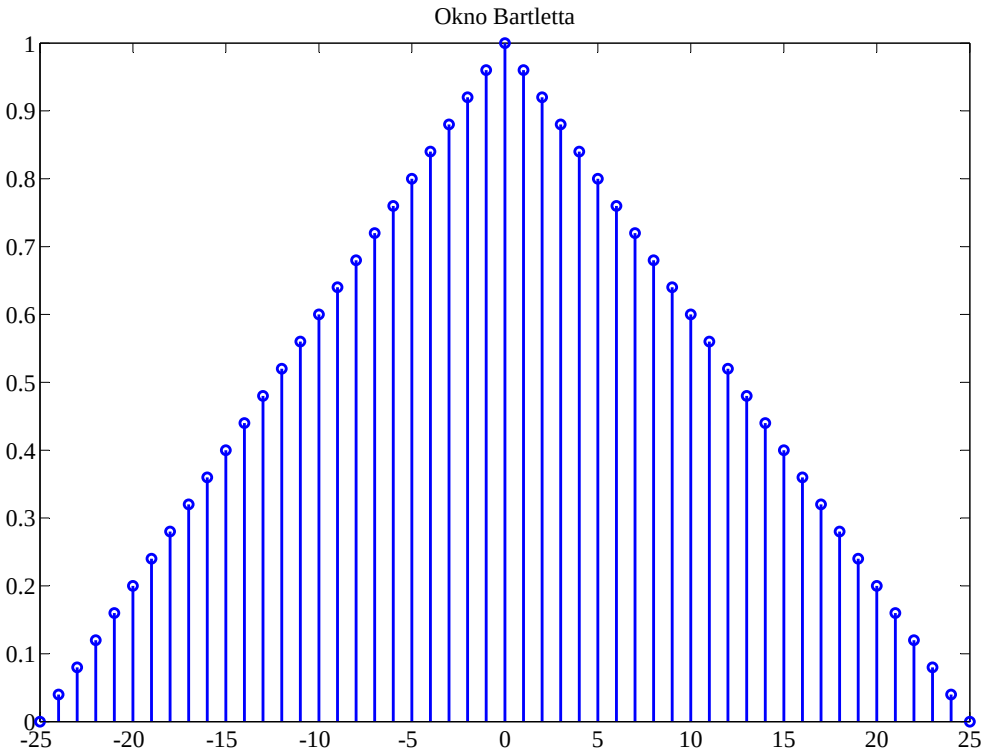
hamming

Co się poprawiło, a co się pogorszyło?



bartlett

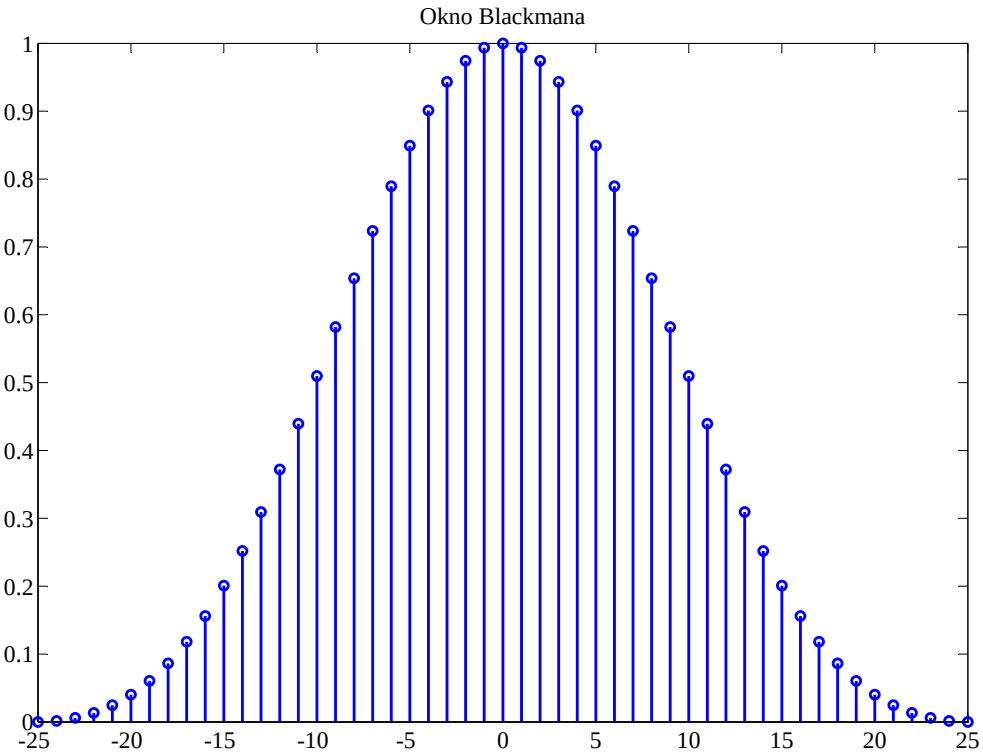
$$w[k] = \begin{cases} \frac{2(k-1)}{n-1} & \text{dla } 1 \leq k \leq \frac{n+1}{2} \\ 2 - \frac{2(k-1)}{n-1} & \text{dla } \frac{n+1}{2} \leq k \leq n \end{cases}$$




```
w3 = bartlett(51);
stem(-25:25,w3)
```

blackman

$$w[k] = 0.42 - 0.5\cos\left(2\pi\frac{k-1}{n-1}\right) + 0.08\cos\left(4\pi\frac{k-1}{n-1}\right)$$
$$k = 1, \ldots, n$$

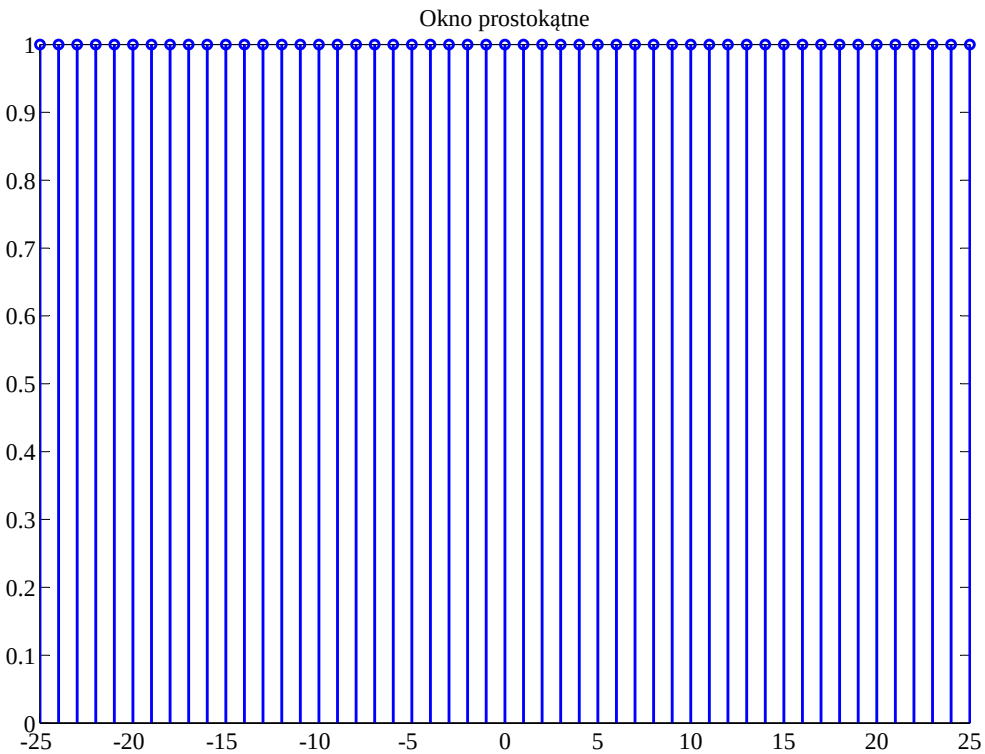



```
w3 = blackman(51);
stem(-25:25,w3)
```

boxcar

Okno prostokątne

w = boxcar(n)

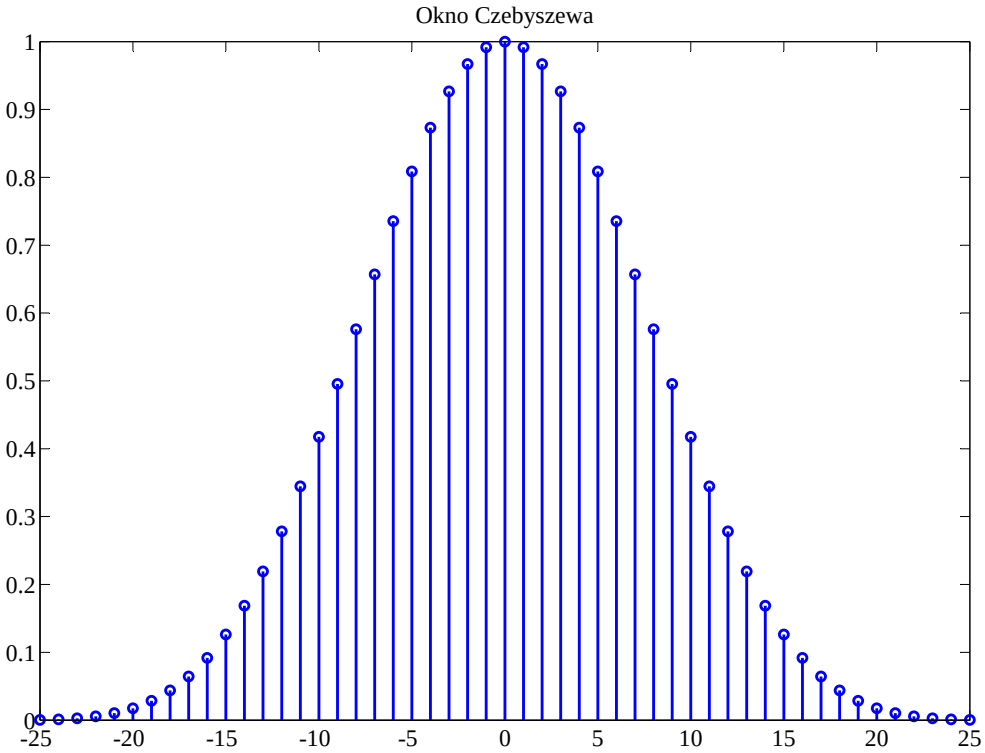



```
w3 = boxcar(51);
stem(-25:25,w3)
```

chebwin

Okno Czebyszewa

w = chebwin(n,r)




```
n=51;    r=100;%dB
w3 = chebwin(n,r);
h=stem(-25:25,w3)
```

chebwin

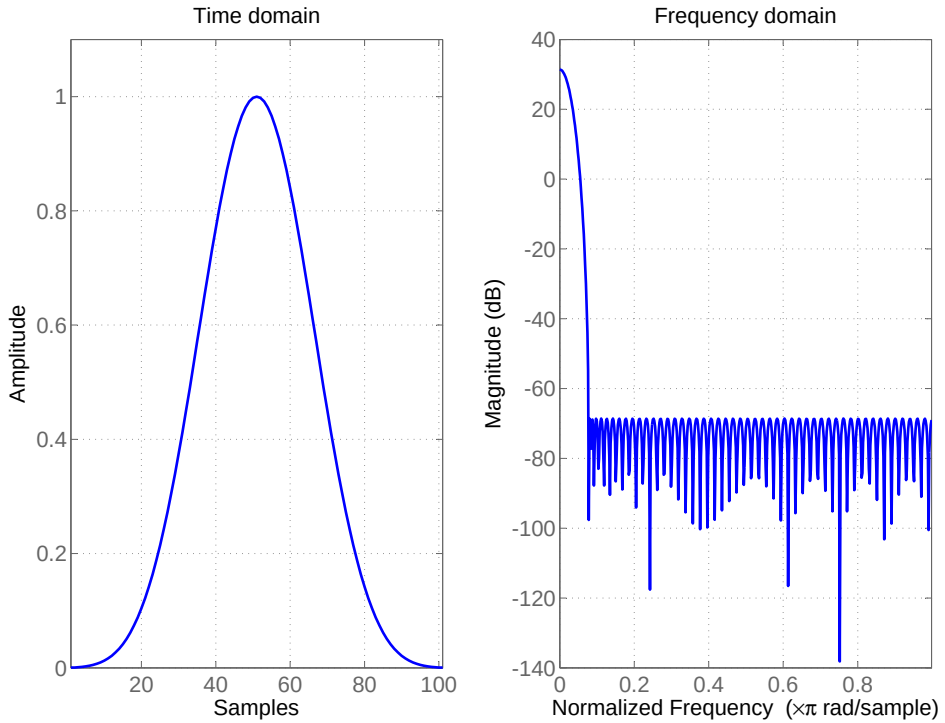
Okno Czebyszewa

w = chebwin(n,r)

n - punktowe okno dla *n* nieparzystego

$(n+1)$ - punktowe okno dla *n* parzystego

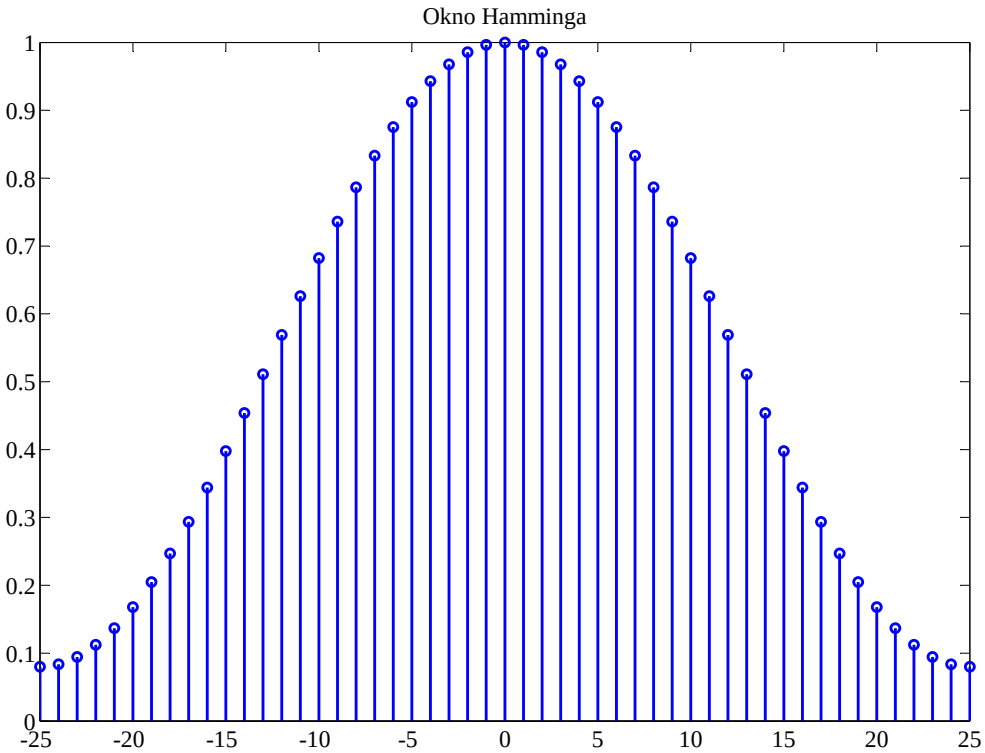
r - wielkość tłumienia zafalowań poza pasmem przenoszenia w [dB]




```
n=51;    r=100;%dB
wvtool(chebwin(n,r))
```

hamming

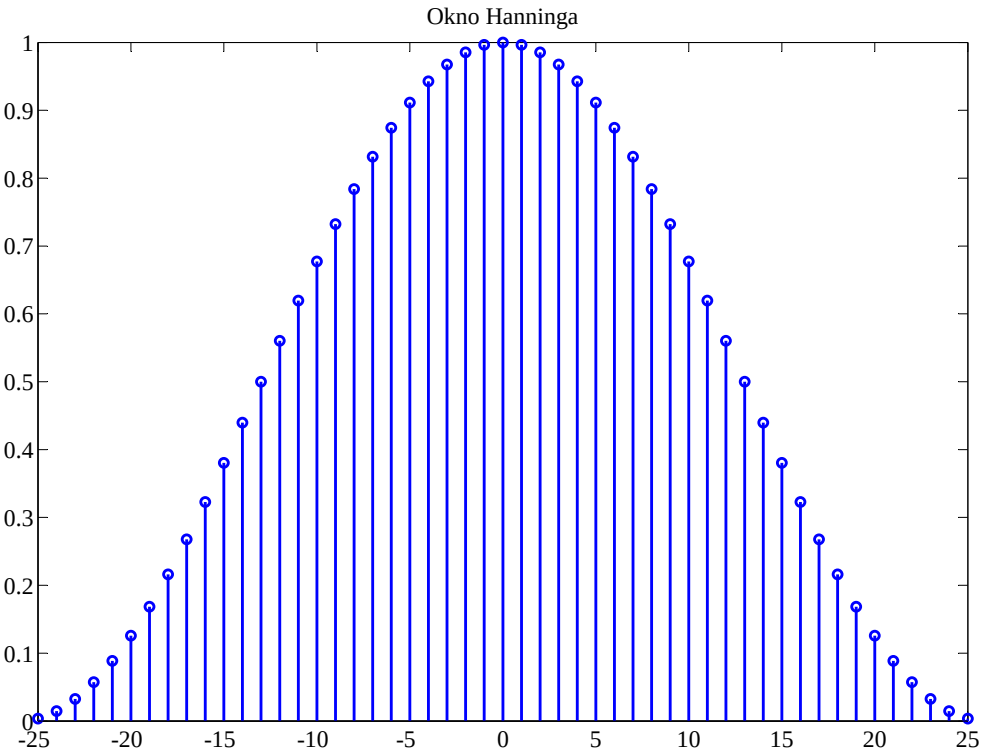
$$w[k] = 0.54 - 0.46\cos\left(2\pi\frac{k}{n-1}\right) \quad k = 0, \ldots, n-1$$




```
w3 = hamming(51);
stem(-25:25,w3)
```

hanning

$$w[k] = 0.5(1 - \cos\left(2\pi \frac{k}{n+1}\right) \quad k = 1, \ldots, n$$




```
w3 = hanning(51);
stem(-25:25,w3)
```

kaiser

Okno Kaisera

$$\mathbf{w = kaiser(n,\beta)}$$

β - współczynnik odpowiedzialny za tłumienie listków bocznych
α - [dB] tłumienie listków bocznych

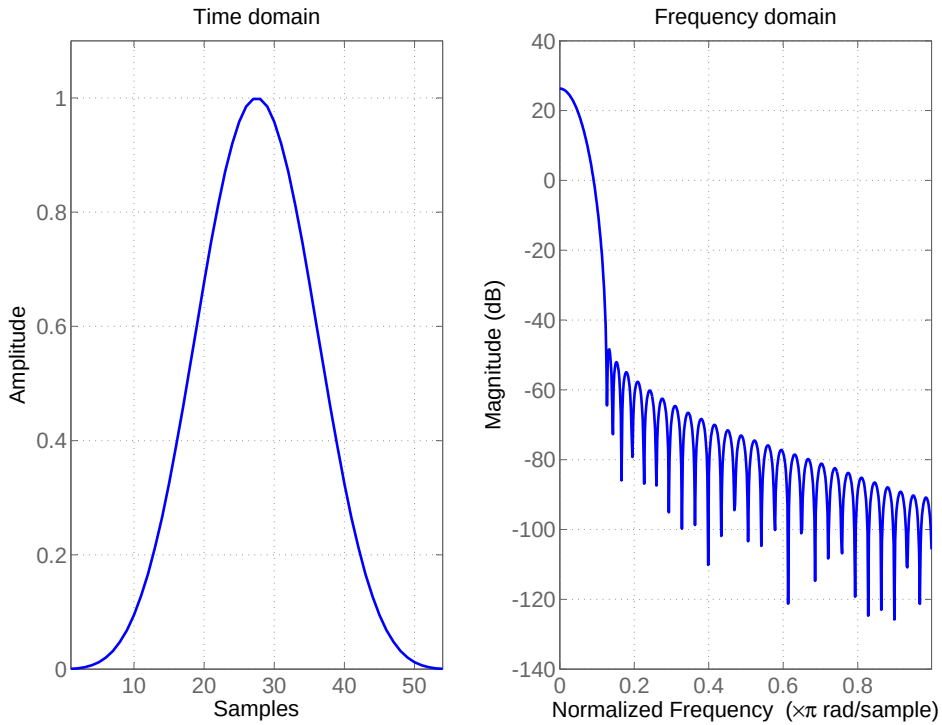
$$\beta = \begin{cases} 0.1102(\alpha - 8.7) & \text{dla } \alpha \geq 50 \\ 0.5842(\alpha - 21)^{0.4} + 0.07886(\alpha - 0.21) & \text{dla } 50 \geq \alpha \geq 21 \\ 0 & \text{dla } \alpha \leq 21 \end{cases}$$

zwiększając β uzyskuje się:

- poszerzenie listka głównego
- zwiększenie tłumienia listków bocznych

$$n \approx \frac{\alpha - 8}{2.285\Delta} + 1 \quad \text{gdzie } \Delta \text{ [rad/s] pasmo}$$

kaiser




```
n=54;   beta = 10.06126; alpha = 100;%dB
wvtool(kaiser(n,beta))
```

triang

Okno trójkątne

$$\mathbf{w = triang(n)}$$

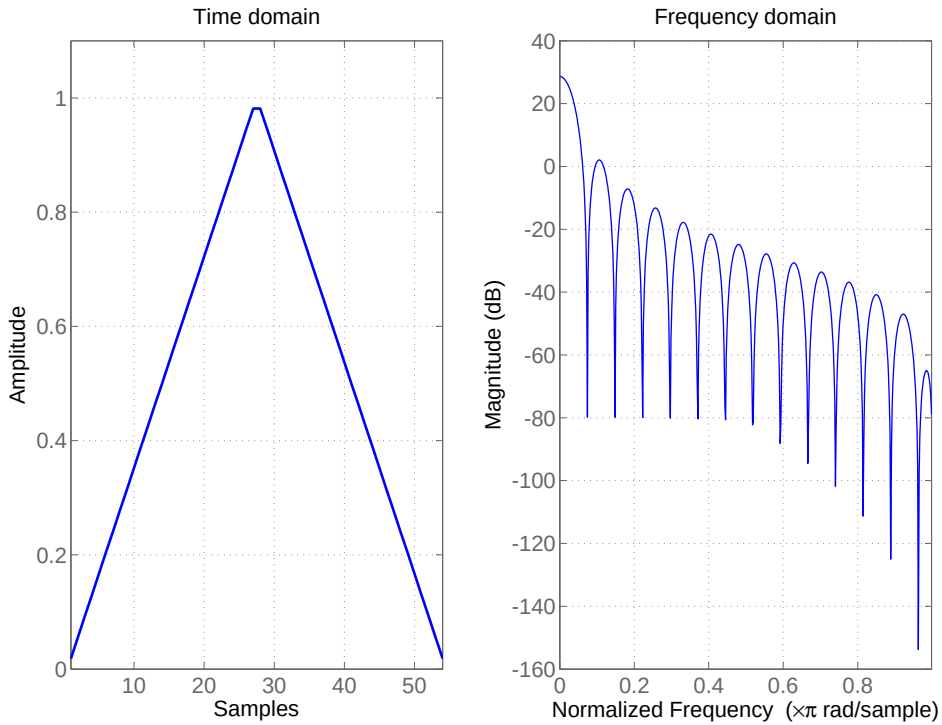
n - nieparzyste

$$w[k] = \begin{cases} \frac{2k}{n+1} & \text{dla } 1 \leq k \leq \frac{n+1}{2} \\ \frac{2(n-k+1)}{n+1} & \text{dla } \frac{n+1}{2} \leq k \leq n \end{cases}$$

n - parzyste

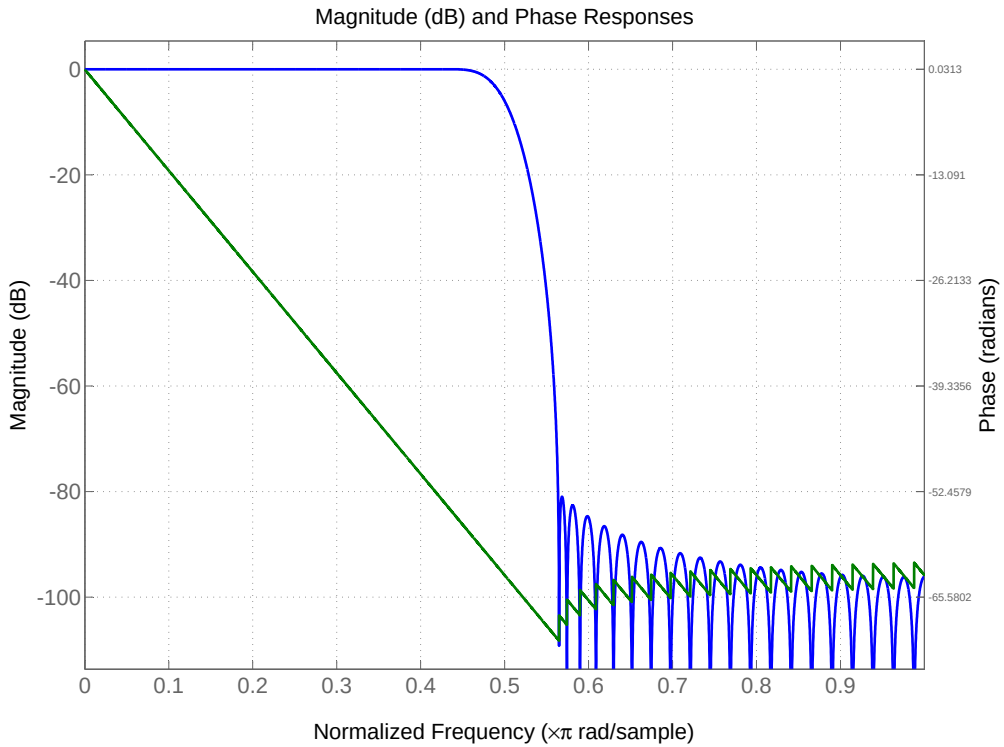
$$w[k] = \begin{cases} \frac{2k-1}{n+1} & \text{dla } 1 \leq k \leq \frac{n}{2} \\ \frac{2(n-k+1)}{n} & \text{dla } \frac{n+2}{2} \leq k \leq n \end{cases}$$

triang




```
n=54;
wvtool(triang(n))
```

kaiser




```
b = fir1(80,0.5,kaiser(81,8));
hd = dfilt.dffir(b);
freqz(hd);
```

wymnażanie okien

$w[n]$ - okno $1 \leq n \leq N$
$h[n]$ - odpowiedź impulsowa idealnego filtru (’prototypu’), czyli odwrotna transformata Fouriera idealnej charakterystyki częstotliwościowej

zatem

$$b[n] = w[n]h[n] \quad 1 \leq n \leq N$$

fir1

Klasyczna metoda syntezy filtru FIR-owego o liniowej fazie

b = fir1(n,Wn,'ftype','window')

$$b(z^{-1}) = b(1) + b(2)z^{-1} + \cdots + b(n+1)z^{-n}$$

n - rząd filtru
$W_n \in [0, 1]$ - wektor unormowanych pulsacji
(1 - odpowiada pulsacji Nyquista)
'ftype' - '**high**', '**stop**'
'window' - typ okna poprawjającego charakterystykę filtru
(domyślnie 'window' = 'hamming')

fir1

$$F_p, f_{g1}, f_{g2}, w_1 = \frac{2f_{g1}}{F_p}, w_2 = \frac{2f_{g2}}{F_p}$$

- filtr dolnopasmowy

  b = fir1(n,w1)

- filtr górnopasmowy (n - parzyste!)

  b = fir1(n,'high',w1,'hann')

- filtr pasmowoprzepustowy

  b = fir1(n,[w1 w2],'bartlett')

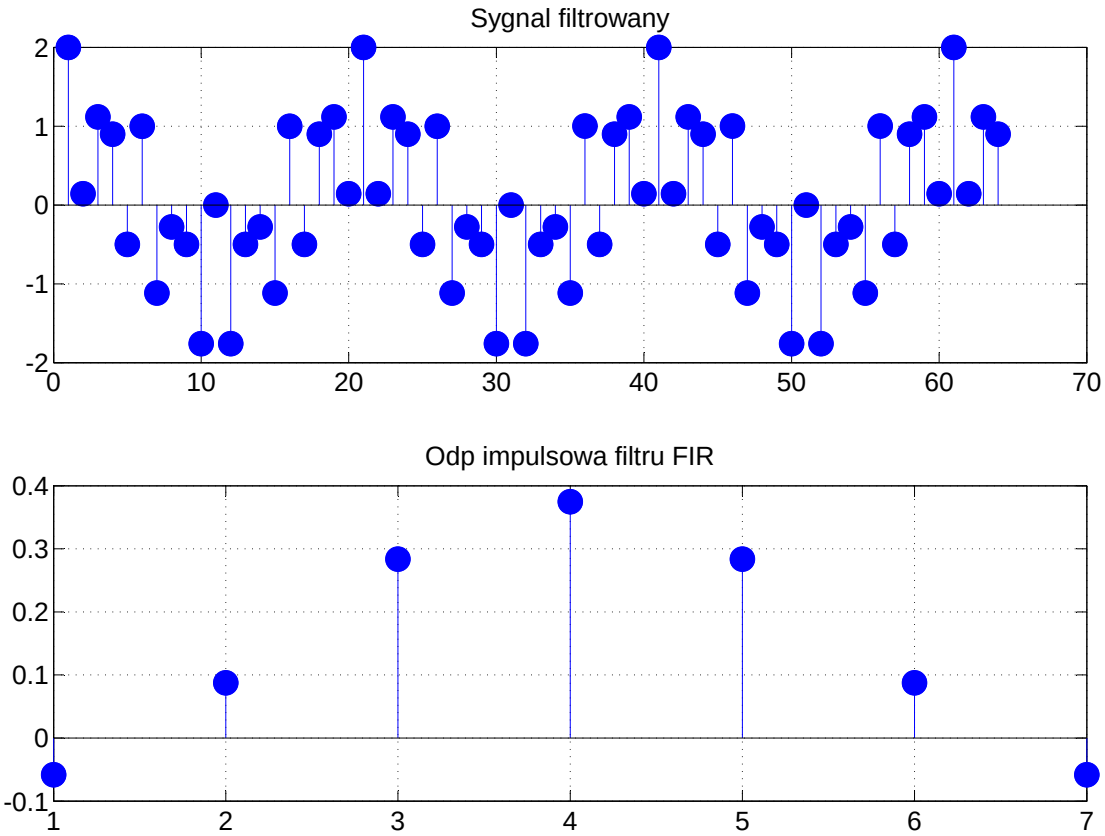
- filtr pasmowozaporowy (n - parzyste!)

  b = fir1(n,'stop',[w1 w2],'boxcar')

fir1

$$F_p = 100Hz, f_1 = 5Hz, f_2 = 40Hz, f_g = 20Hz, w_1 = \frac{2f_g}{F_p}$$

fir1(n,w1)



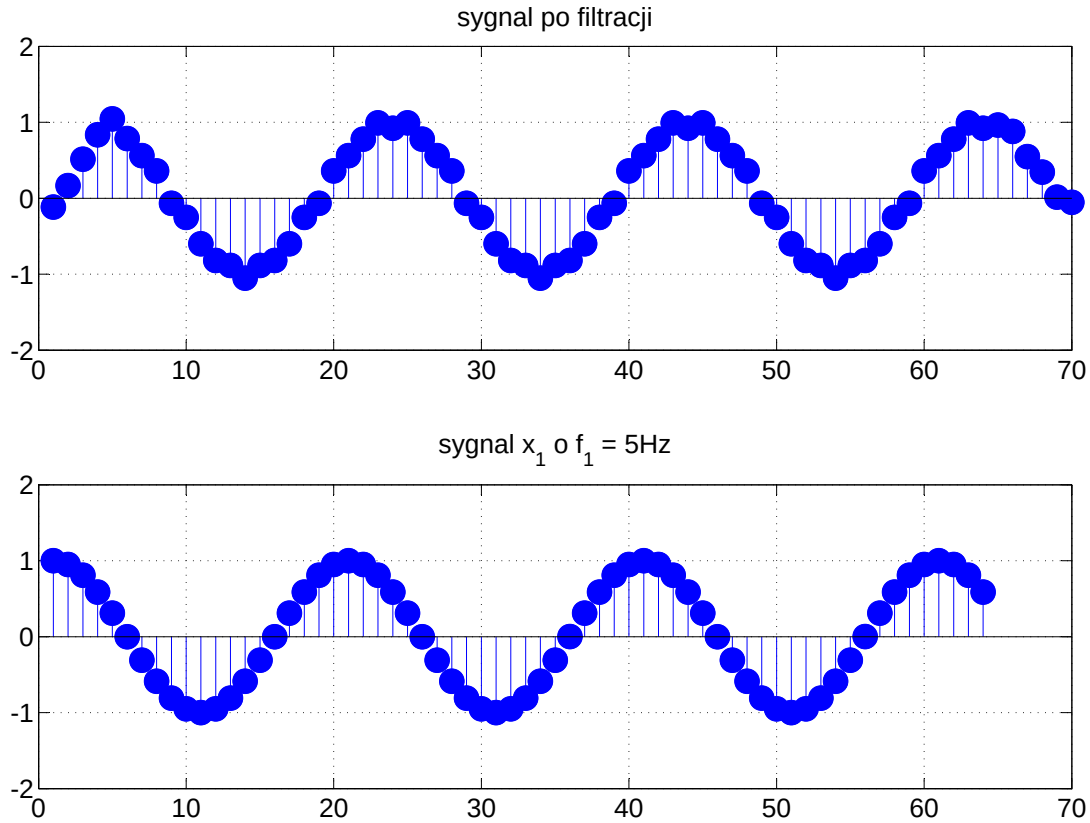
fir1

```
N = 64; fpr=100; fx1=5; fx2=40;
nx = 0:N-1; dt = 1/fpr; t = dt*nx;
x1 = cos(2*pi*fx1*t);
x2 = cos(2*pi*fx2*t);
x = x1 + x2;

M = 7; fg = 20;
h = fir1(M-1,2*fg/fpr,boxcar(M))
>>ans = -0.0584  0.0876  0.2835  0.3746  0.2835  0.0876
-0.0584

subplot(211);  stem(x,'filled'); grid;
title('Sygnał filtrowany');
subplot(212); stem(h,'filled'); grid;
title('Odp impulsowa filtru FIR');
```

conv

Funkcja MATLAB-a pozwala w łatwy sposób dokonać splotu dwóch wektorów

conv(x,h)



conv

Funkcja splotu realizowana przez MATLAB-a

```
% odwróć kolejność próbek
    h = h(M:-1:1);
% uzupełnij sygnał M-1 zerami po obu stronach
    xe = [zeros(1,M-1) x zeros(1,M-1)];
% sygnał wyjściowy
    ye = zeros(1,N+2*(M-1));
    subplot(311); stem(xe,'filled'); title('WE'); pause
% liczba próbek „niezerowych”
    for n = 1 : N+(M-1)
% przesuń (ustaw) odp impulsową
        he = [zeros(1,n-1) h zeros(1,(N-1)+(M-1)-(n-1))];
% wymnóż x i przesunięte h
        y(n) = sum( xe .* he ); ye((M-1)+n)=y(n);
        % narysuj przesunięte h
        subplot(312); stem(he,'filled'); title('odp impuls');
% narysuj aktualne wyjście
        subplot(313); stem(ye,'filled'); title('WY');
        pause
    end
    subplot(111); plot(t,x1,'r',t,y(1:N),'b');
     grid; title('We (R) i Wy (B)');
```

splot bezpośredni

```
%  długość sygnałów z dodanymi zerami
  K = N+M-1;
%  dodaj zera jeśli krótszy
  xz=[x zeros(1,K-N)];
%  odwróć kolejność próbek
  hh = h(M:-1:1);
%  bufor na próbki wejściowe
  bx = zeros(1,M);
%  wyzeruj sygnał wyjściowy
  y = [];
%  pętla „po” próbkach
  for n = 1 : K
%  przesuń próbki w buforze o jedną do tyłu, bx(1)=x(n);
     bx = [xz(n) bx(1:M-1)];
%  filtracja; inaczej (szybciej): y(n)=bx*h'
     y(n) = sum(bx.*hh);
  end
```

szybki splot

```
%  znajdź dłuższy
  NM = max(N,M);
%  dodaj zera, jeśli krótszy
  xz=[x zeros(1,NM-N)];
%  dodaj zera, jeśli krótszy
  hz=[h zeros(1,NM-M)];
  X = fft(xz); H=fft(hz);
  Y = X .* H;
  yfft = ifft(Y);
  y1 = real(yfft);
```

splot liniowy

```
%  długość sygnałów z dodanymi zerami
  K = N+M-1;
%  dodaj zera, jeśli krótszy
  xz=[x zeros(1,K-N)];
%  dodaj zera, jeśli krótszy
  hz=[h zeros(1,K-M)];
  X = fft(xz); H=fft(hz);
  Y = X .* H;
  yfft = ifft(Y);
  y2 = real(yfft);
```

fir2

Arbitralne kształtowanie charakterystyki amplitudowej filtru FIR

b = fir2(n,f,m)

n - rząd filtru

f - wektor unormowanych częstotliwości względem częstotliwości Nyquista $\in [0, 1]$ (uporządkowanie narastające!)

m - wektor amplitud (modułów) charakterystyki filtru dla częstotliwości z wektora ***f***

fir2

Arbitralne kształtowanie charakterystyki amplitudowej filtru FIR

b = fir2(n,f,m,'window')

'**window**' - okno użyte w projekcie filtru (wektor o $n+1$ współczynnikach, dom. okno Hamminga)

fir2

Arbitralne kształtowanie charakterystyki amplitudowej filtru FIR

b = fir2(n,f,m,p,'window')

b = fir2(n,f,m,p)

p - liczba punktów interpolacji, które są przyjmowane, aby uzyskać dokładniejszą specyfikacją ch-ki amplitudowej (dom. 512)

fir2

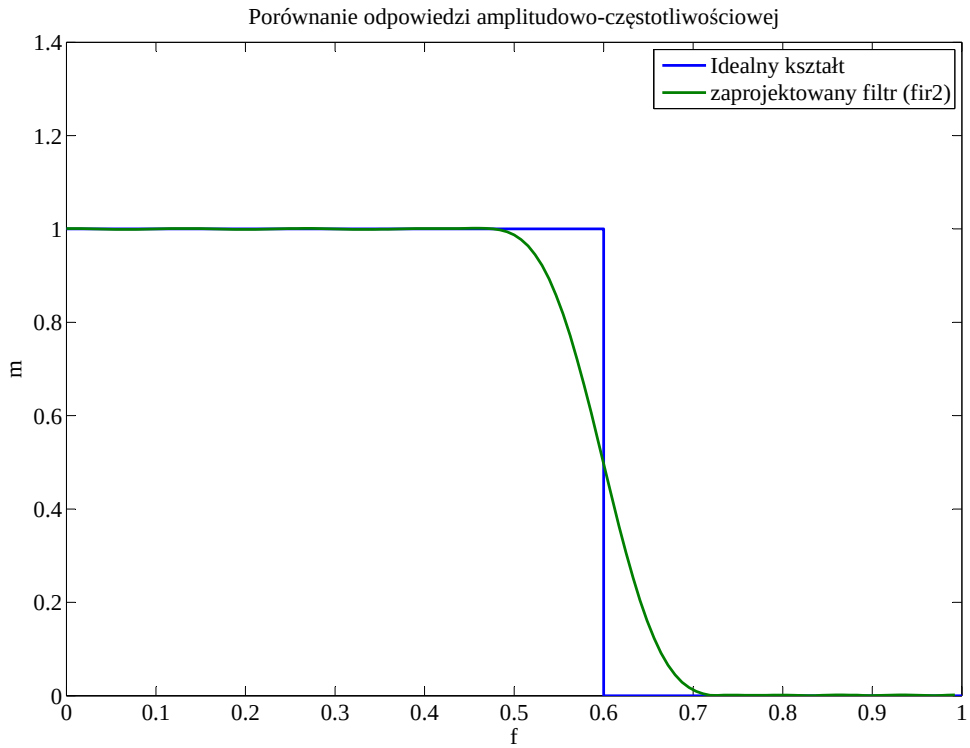
Arbitralne kształtowanie charakterystyki amplitudowej filtru FIR

b = fir2(n,f,m,p,lap)

b = fir2(n,f,m,p,lap,'window')

lap - specyfikuje się rozmiar obszaru, który otaczaa punkty (na skali częstotliwości, wektor ***f***) o jednakowych wartościach (dom. 25)

fir2




```
n=30;     f = [0 0.6 0.6 1];     m = [1 1 0 0];
b = fir2(n,f,m);
[h,w] = freqz(b,1,128);
plot(f,m,w/pi,abs(h))
```

firls

Projektowanie filtrów FIR-owych o liniowej fazie metodą najmniejszych kwadratów (aproksymacja zadanej ch-ki amplitudowej)

b = firls(n,f,m,w)

n - rząd filtru

b - wektor $(n+1)$ współczynników filtru, którego ch-ka amplitudowa aproksymuje ch-kę opisaną przez wektory ***f*** i ***m***

f - wektor unormowanych częstotliwości względem częstotliwości Nyquista $\in [0, 1]$ (uporządkowanie narastające!)

m - wektor amplitud (modułów) charakterystyki filtru dla częstotliwości z wektora ***f***

w - wektor wagowy równy o długości równej połowie ***f***

firls

- Współczynniki wektora ***b*** są symetrycznie ułożone

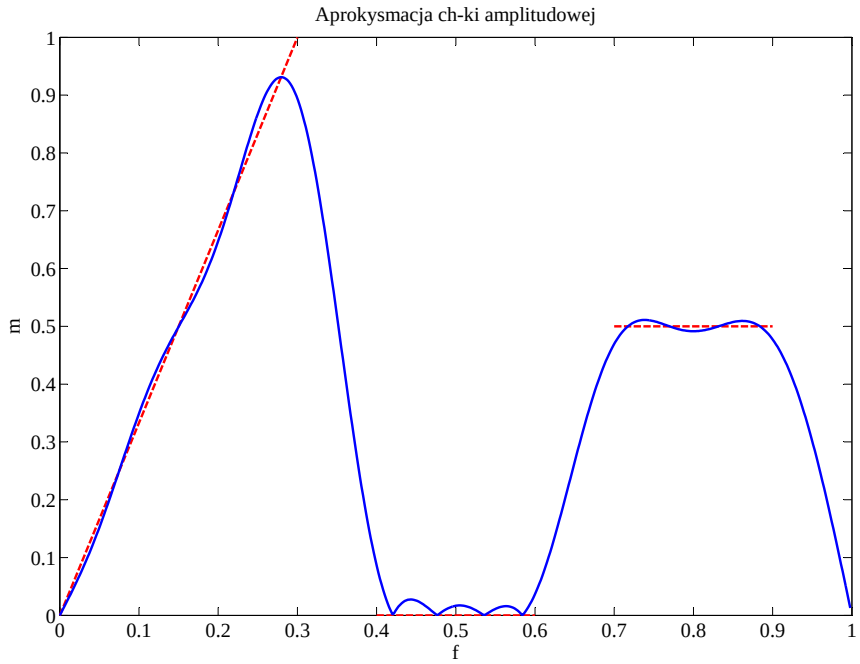
$$b(k) = b(n+2-k), \quad k = 1, \ldots, n+1$$

- ***f*** i ***m*** muszą być tej samej długości i musi to być liczba parzysta

- funkcja łączy pary punktów $(f(k), m(k))$ i $(f(k+1), m(k+1))$ odcinkiem liniowym dla k nieparzystych

- funkcja nie łączy par punktów $(f(k), m(k))$ i $(f(k+1), m(k+1))$ odcinkiem liniowym dla k parzystych (nie są uwzględniane w aprokysmacji)

firls




```
f = [0 0.3  0.4 0.6  0.7 0.9];   m = [0  1   0  0  0.5 0.5];
b = firls(24,f,m,'hilbert');
for i=1:2:6,
   plot([f(i) f(i+1)],[m(i) m(i+1)],'r--'), hold on
end
[H,w] = freqz(b,1,512,2);
plot(w,abs(H))
```

podsumowanie

W MATLAB-ie:

- filtry cyfrowe projektuje się dla unormowanych częstotliwości
- częstotliwość jednostkowa to częstotliwość Nyquista $= \frac{1}{2}$ częstotliwości próbkowania
- częstotliwość unormowana $\in [0, 1]$
- np.: próbkowanie z częstotliwością 1000 Hz, rzeczywista częstotliwość sygnału $f = 300$ Hz, co odpowiada częstotliwości unormowanej $\frac{300}{500} = 0.6$
- zamiana częstotliwości unormowanej na pulsację: $x\pi$
- zamiana częstotliwości unormowanej na częstotliwość rzeczywistą: $x\frac{1}{2}F_s$

IIR

Filtr o nieskończonej odpowiedzi impulsowej

Zalety:

- niska złożoność obliczeniowa
- niewielkie zapotrzebowanie na pamięć operacyjną.

Wady:

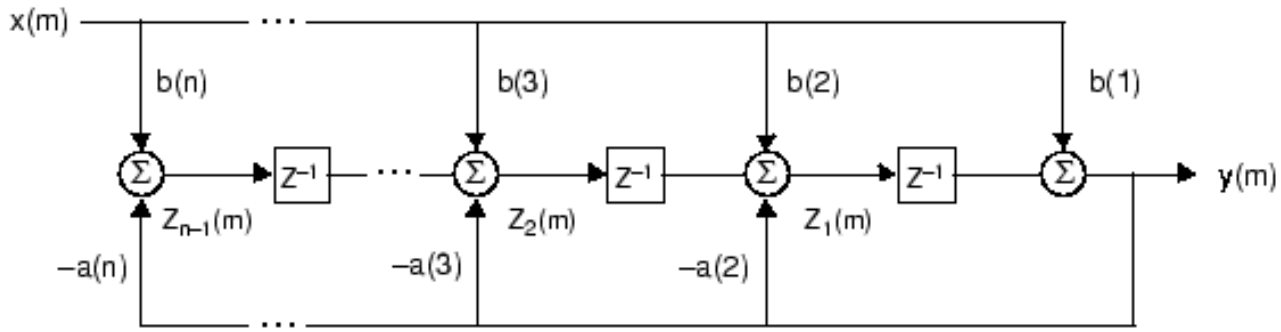
- zagrożenie utraty stabilności
- projektowanie filtrów IIR jest znacznie trudniejsze niż w przypadku filtrów FIR
- filtry IIR są znacznie bardziej wrażliwe na błędy zaokrągleń
- nieliniowość fazy

IIR

Transmitancja filtru w dziedzinie zmiennej zespolonej **Z**

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \cdots + b(nb+1)z^{-nb}}{1 + a(2)z^{-1} + \cdots + a(na+1)z^{-na}} X(z)$$

Schemat filtru - bezpośrednia struktura typu II



IIR

Równanie różnicowe

$$y(n) = b(1)x(n) + b(2)x(n-1) + ... + b(nb+1)x(n-nb) -$$

$$-a(2)y(n-1) - ... - a(na+1)y(n-na)$$

Filtracji sygnału dokonujemy za pomocą gotowej funkcji MATLAB-a

y = filter(b,a,x)

x - sygnał wejściowy
y - sygnał wyjściowy po filtracji

butter

Filtr Butterwortha (LP):

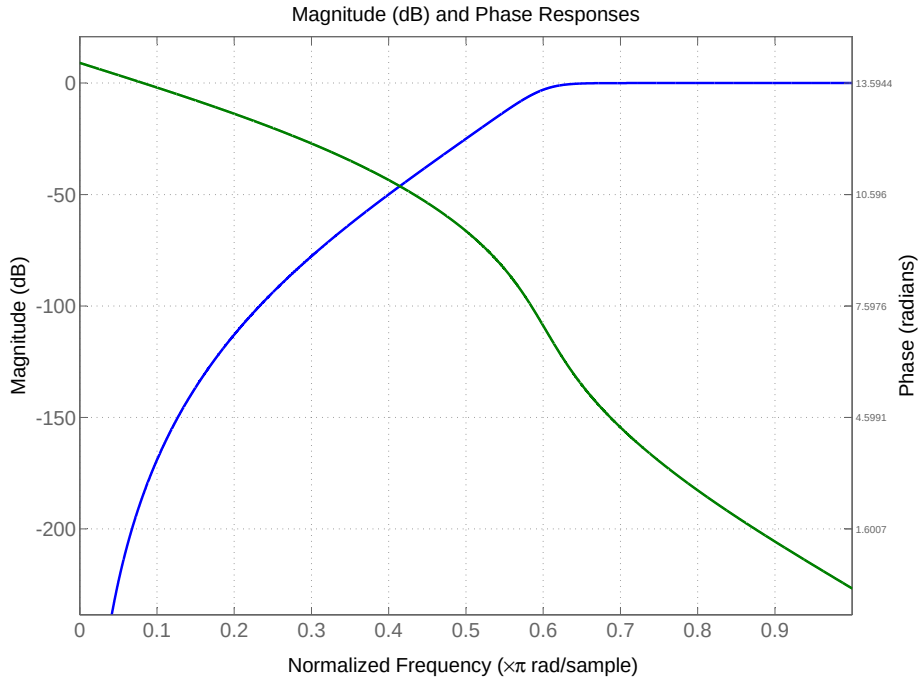
- maksymalnie płaska charakterystka modułu w paśmie przenoszenia
- poza pasmem ch-ka ta monotonicznie maleje

[b,a,k] = butter(n,Wn,'ftype')

[A,B,C,D] = butter(n,Wn,'ftype')

Wn - odpowiada modułowi ch-ki widmowej $= \frac{1}{\sqrt{2}}$ (3dB)
'ftype' - 'high', 'low', 'stop'
[A, B, C, D] - model w przestrzeni stanów

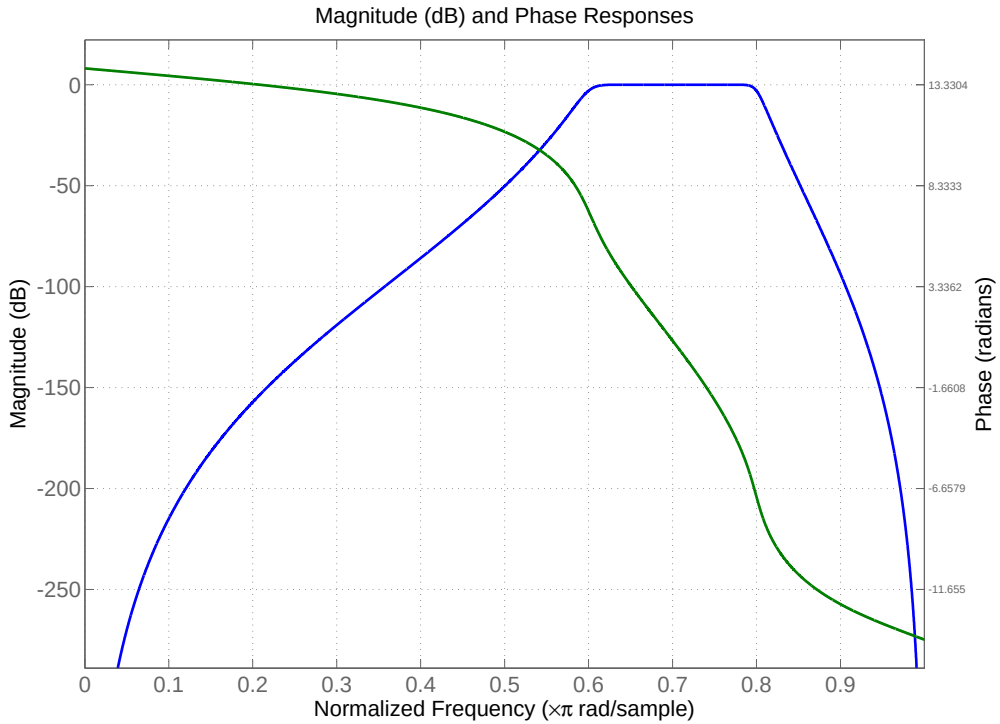
butter




```
n=9; Fs = 1000; F = 300; w1 = F/(Fs/2);
[b,a] = butter(n,w1,'high');
[sos,g] = zp2sos(b,a);
Hd = dfilt.df2tsos(sos,g);
h = fvtool(Hd);
set(h,'Analysis','freq')
```

butter

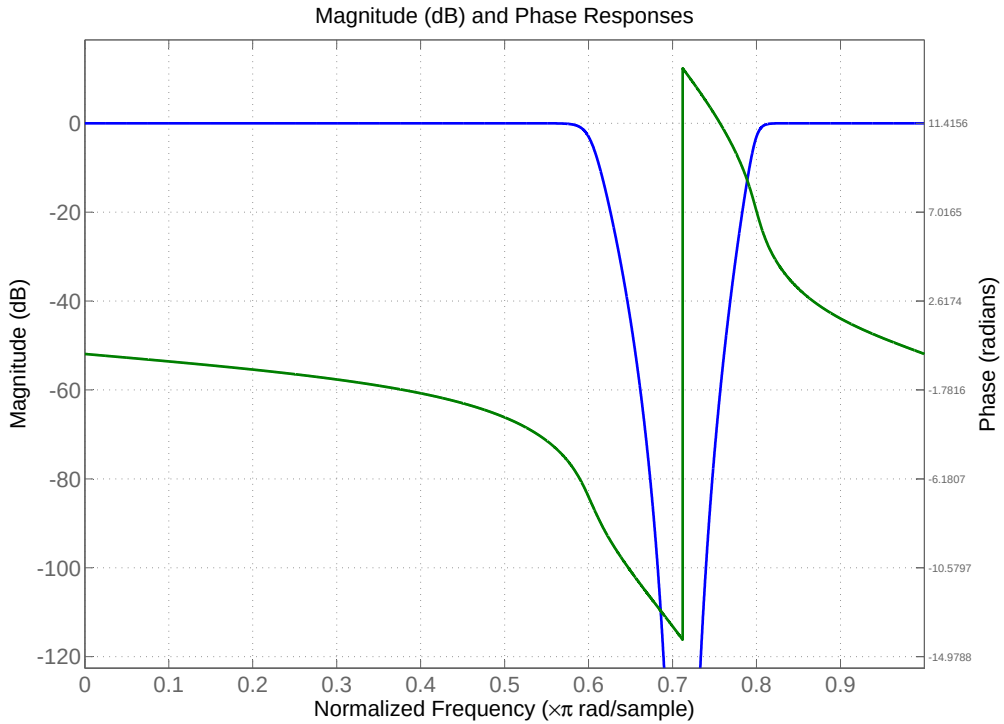
Otrzymujemy $2n$-rzędowy filtr




```
n=9; Fs = 1000; F1 = 300; F2 = 400;
w1 = F1/(Fs/2); w2 = F2/(Fs/2);
[b,a] = butter(n,[w1,w2]);
[sos,g] = zp2sos(b,a);
Hd = dfilt.df2tsos(sos,g);
h = fvtool(Hd);
set(h,'Analysis','freq')
```

butter

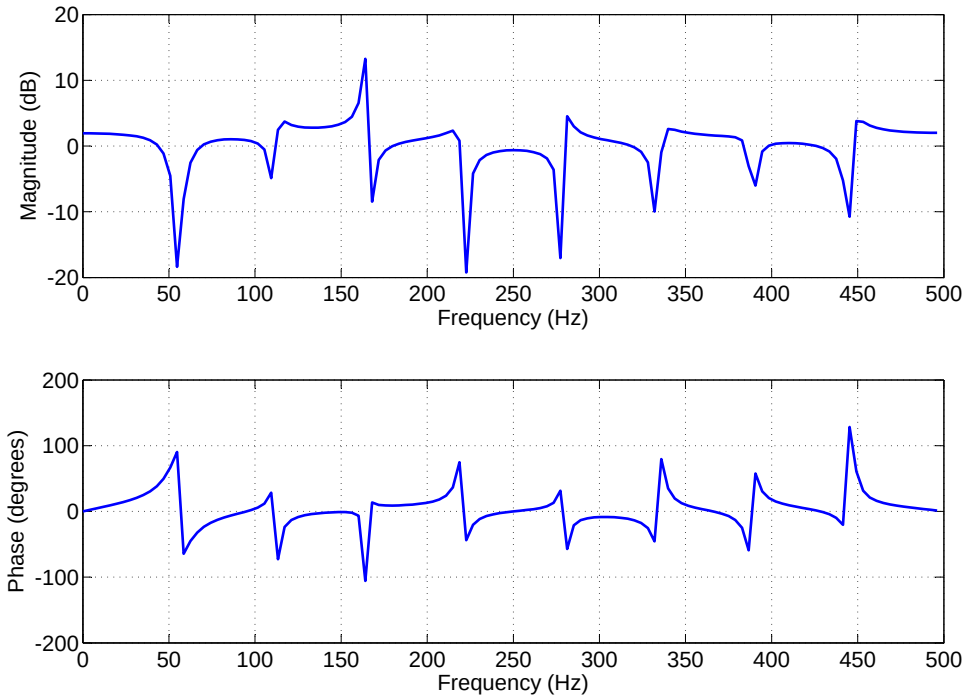
Otrzymujemy 2*n*-rzędowy filtr




```
n=9; Fs = 1000; F1 = 300; F2 = 400;
w1 = F1/(Fs/2); w2 = F2/(Fs/2);
[b,a] = butter(n,[w1,w2],'stop');
[sos,g] = zp2sos(b,a);
Hd = dfilt.df2tsos(sos,g);
h = fvtool(Hd);
set(h,'Analysis','freq')
```

butter

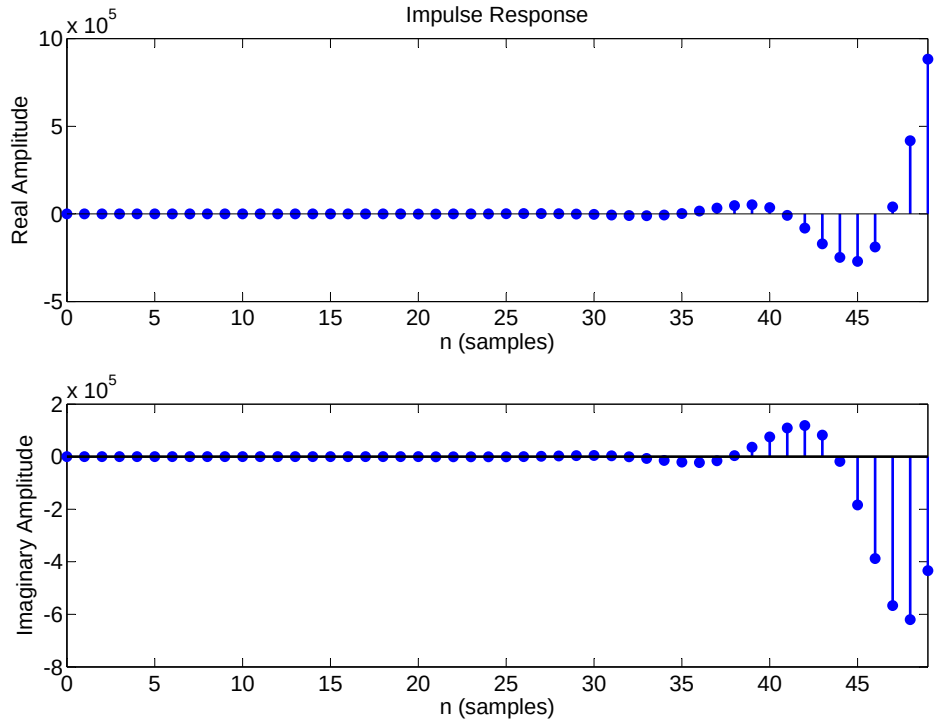
Odpowiedź częstotliwościowa filtru cyfrowego




```
n=9; Fs = 1000; F1 = 300; F2 = 400;
w1 = F1/(Fs/2); w2 = F2/(Fs/2);
[b,a] = butter(n,[w1,w2],'stop');
freqz(b,a,128,Fs)
```

butter

Odpowiedź impulsowa filtru cyfrowego




```
n=5; Fs = 1000; F1 = 100; F2 = 200;
w1 = F1/(Fs/2); w2 = F2/(Fs/2);
[z,p,k] = butter(n,[w1 w2]);
impz(z,p,50)
```

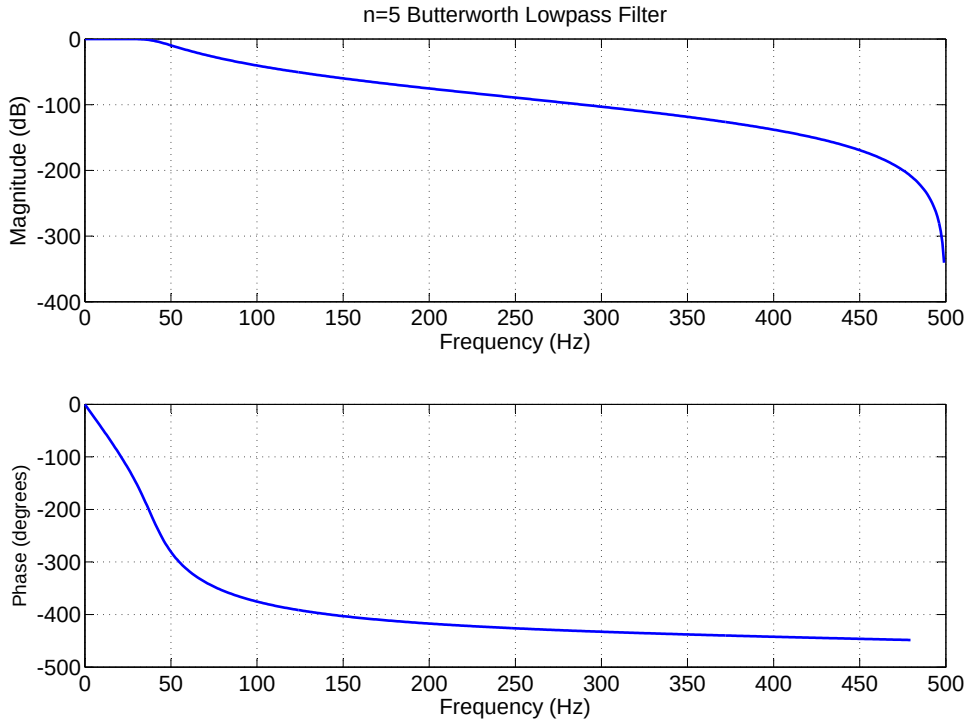
buttord

Selekcja minimalnego rzędu filtru spełniającego wymagania projektowe

[n,Wn] = buttord(Wp,Ws,Rp,Rs)

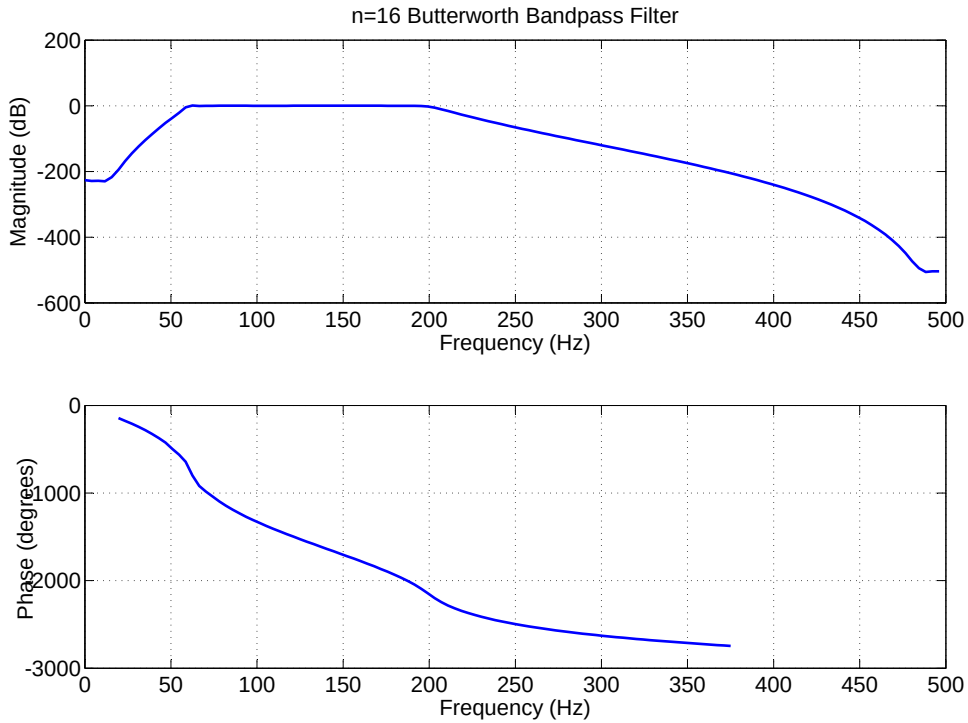
Wp - częstotliwość odcięcia (pasmo $\in [0, 1]$)
Ws - częstotliwość pasma zaporowego $\in [0, 1]$
Rp - dopuszczalne zafalowania w paśmie przenoszenia [dB]
Rs - tłumienie w paśmie zaporowym [dB]

buttord




```
Fs = 1000; Wp = 40/500; Ws = 150/500;
Rp = 3; Rs = 60;
[n,Wn] = buttord(Wp,Ws,3,60);     >> n = 5;    >> Wn = 0.0810;
[b,a] = butter(n,Wn);
freqz(b,a,512,Fs);
```

buttord




```
Fs = 1000; Wp = [60 200]/500; Ws = [50 250]/500;
Rp = 3; Rs = 40;
[n,Wn] = buttord(Wp,Ws,Rp,Rs);
>> n =16; >> Wn =[0.1198 0.4005];
[b,a] = butter(n,Wn);
freqz(b,a,128,Fs)
```

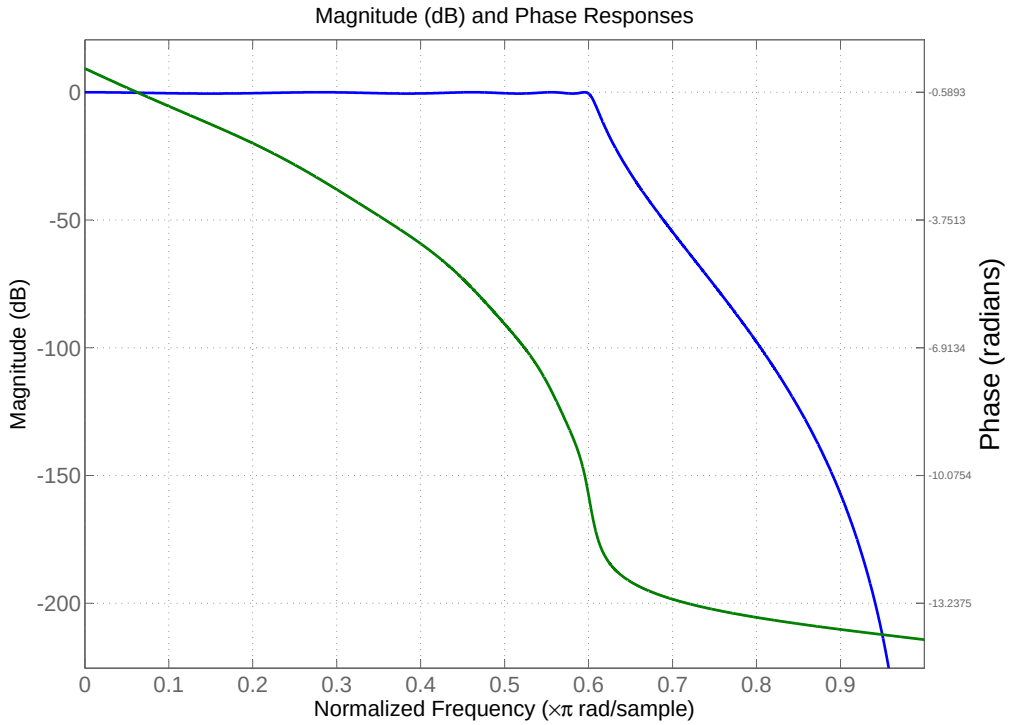
cheby1

Filtr Czebyszewa

[z,p,k]=cheby1(n,Rp,Wp,'ftype')

- jednakowe zafalowania w paśmie przenoszenia Rp [dB]
- maksymalnie płaska ch-ka w pasmie zaporowym
- przejście od pasma przenoszenia do pasma zaporowego szybsze niż w przypadku f. Butterwortha

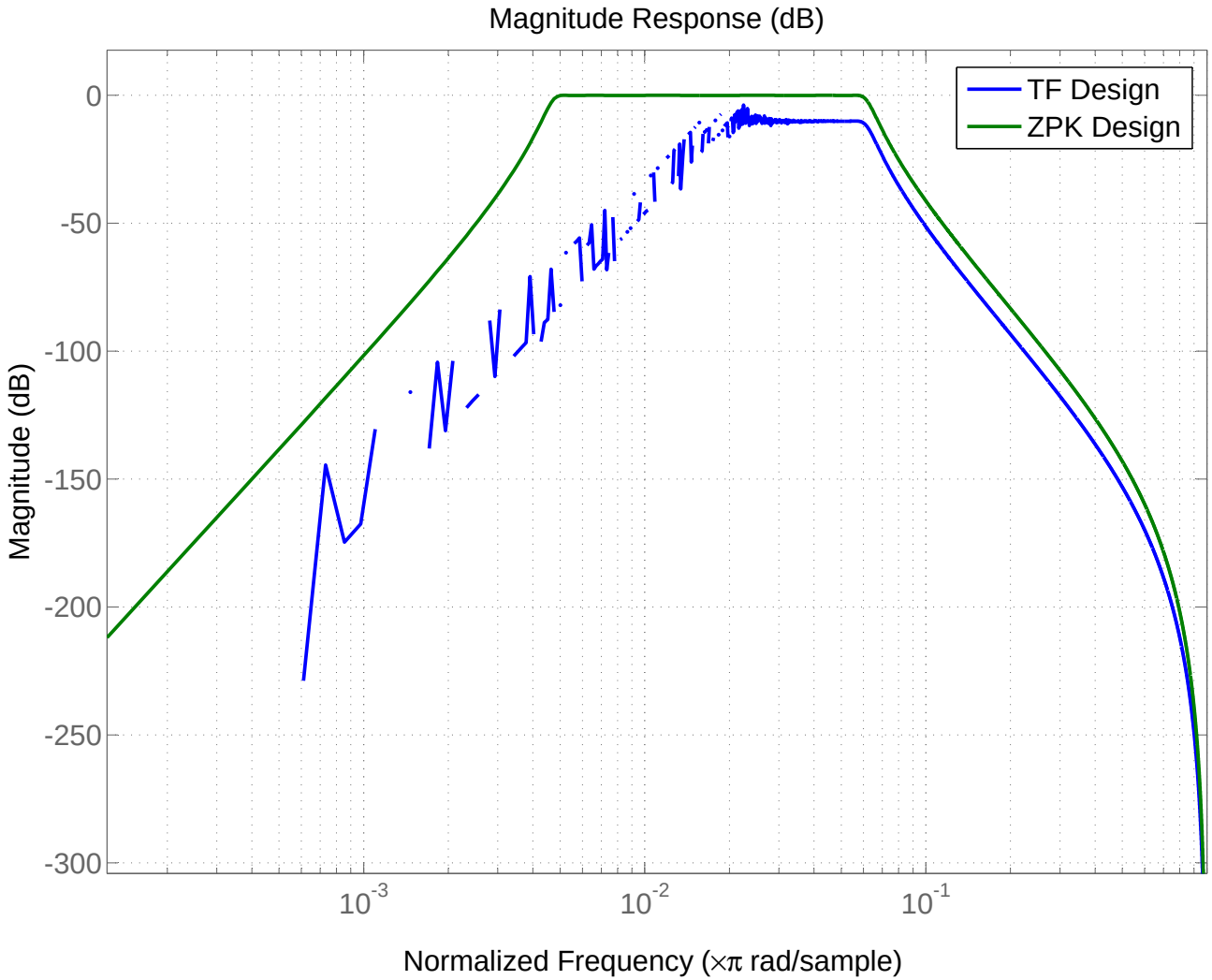
cheby1




```
n = 9; Rp = 0.5; F = 300Hz; Fs=1000Hz;
[z,p,k] = cheby1(9,0.5,300/500);
[sos,g] = zp2sos(z,p,k);
Hd = dfilt.df2tsos(sos,g);
h = fvtool(Hd)
set(h,'Analysis','freq')
```

cheby1

Problemy numeryczne związane z zaokrągleniem



zp2sos

Metody projektowania filtrów IIR

[z,p,k]=cheby1(n,Rp,Wp,'ftype')

$$H(z) = k\frac{(z-z_1)(z-z_2)\dots(z-z_n)}{(z-p_1)(z-p_2)\dots(z-p_m)}$$

[sos,g]=zp2sos(z,p,k)

$$\boldsymbol{sos} = \begin{bmatrix} b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & 1 & a_{1L} & a_{2L} \end{bmatrix}$$

$$H(z) = g\prod_{k=1}^{L} H_k(z) = g\frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{1 + a_{1k}z^{-1} + a_{2k}z^{-2}}$$

cheby1

```
n = 6;
Rp = 0.1;
Wn = ([2.5e6 29e6]/500e6);
ftype = 'bandpass';

%Filtr niestabilny !!!
[b,a] = cheby1(n,Rp,Wn,ftype);
h1=dfilt.df2(b,a);


% Zero-Pole-Gain design
[z, p, k] = cheby1(n,Rp, Wn,ftype);
[sos,g]=zp2sos(z,p,k);

%Filtr stabilny!!!
h2=dfilt.df2sos(sos,g);

hfvt=fvtool(h1,h2,'FrequencyScale','log');
legend(hfvt,'TF Design','ZPK Design')
```

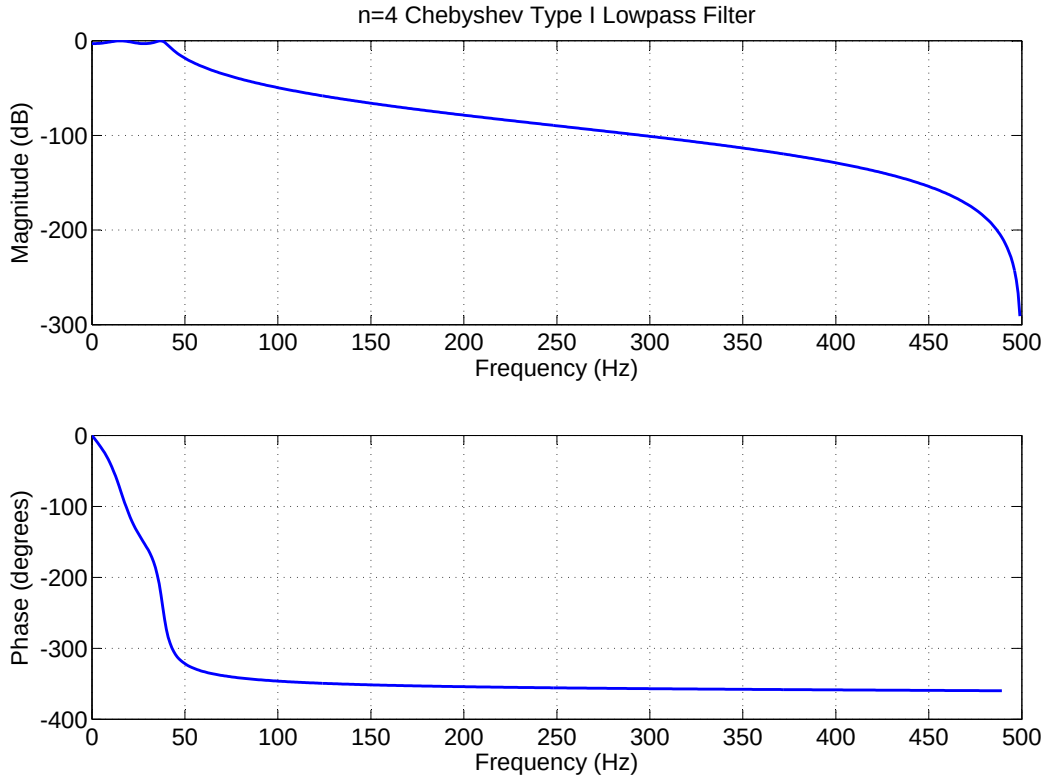
cheb1ord

Szacowanie rzędu filtru dla zadanych specyfikacji

[n,Wp]=cheb1ord(Wp,Ws,Rp,Rs)

Wp - częstotliwość odcięcia (pasmo $\in [0, 1]$)
Ws - częstotliwość pasma zaporowego $\in [0, 1]$
Rp - dopuszczalne zafalowania w paśmie przenoszenia [dB]
Rs - tłumienie w paśmie zaporowym [dB]

cheb1ord




```
Wp = 40/500; Ws = 150/500;
Rp = 3; Rs = 60;
[n,Wp] = cheb1ord(Wp,Ws,Rp,Rs)
% Returns n = 4 Wp =0.0800
[b,a] = cheby1(n,Rp,Wp);
freqz(b,a,512,1000);
title('n=4 Chebyshev Type I Lowpass Filter')
```

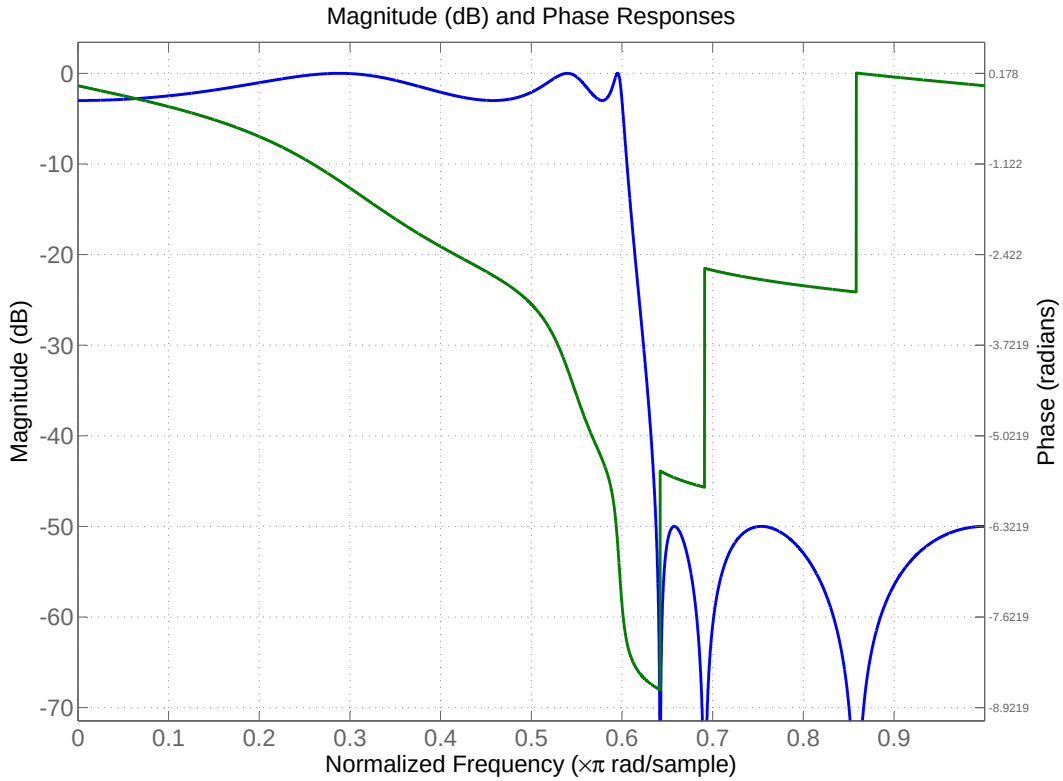
ellip

Filtr eliptyczny zapewnia „najszybsze” przejście dla danego rzdęu z pasma przenoszenia do pasma zaporowego

[z,p,k] = ellip(n,Rp,Rs,Wp,'ftype')

Wp - częstotliwość odcięcia (pasmo $\in [0, 1]$)
Rp - dopuszczalne zafalowania w paśmie przenoszenia [dB]
Rs - tłumienie w paśmie zaporowym [dB]

ellip




```
n=6; Rp = 3; Rs=50; F=300; Fs=1000; w = F/(Fs/2);
[z,p,k] = ellip(n,Rp,Rs,w);
[sos,g] = zp2sos(z,p,k);
Hd = dfilt.df2tsos(sos,g);
h = fvtool(Hd)
set(h,'Analysis','freq')
```

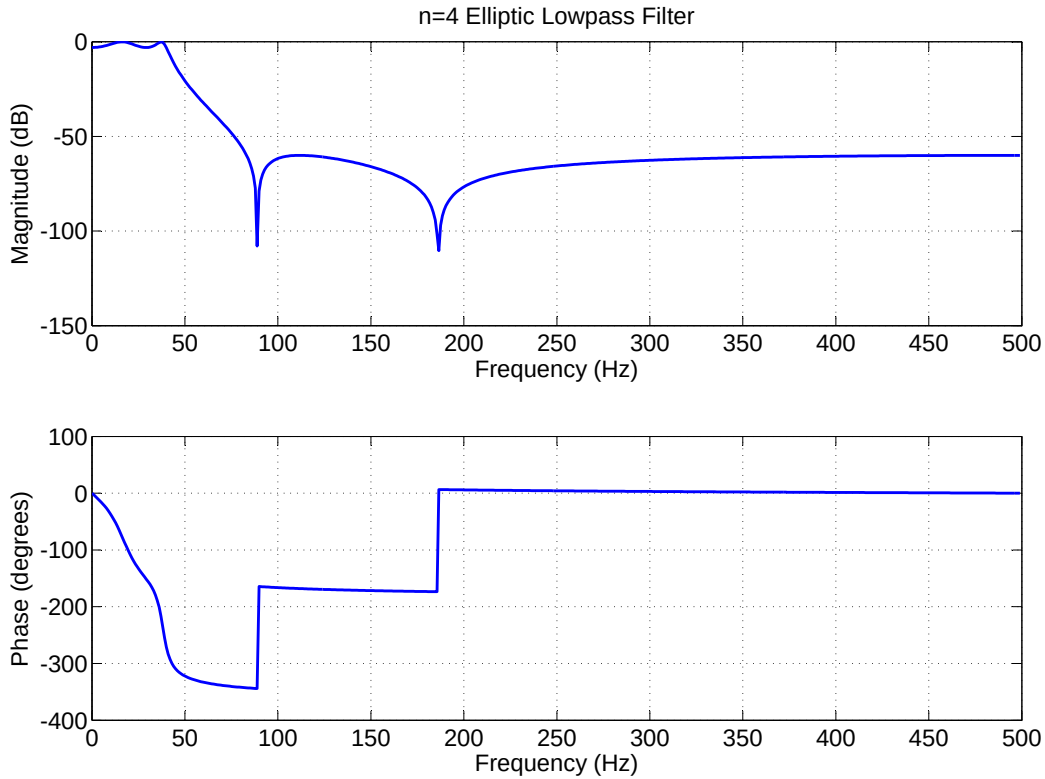
ellipord

Szacowanie rzędu filtru dla zadanych specyfikacji

[n,Wp]=ellipord(Wp,Ws,Rp,Rs)

Wp - częstotliwość odcięcia (pasmo $\in [0, 1]$)
Ws - częstotliwość pasma zaporowego $\in [0, 1]$
Rp - dopuszczalne zafalowania w paśmie przenoszenia [dB]
Rs - tłumienie w paśmie zaporowym [dB]

ellipord




```
Wp = 40/500; Ws = 150/500;
Rp = 3; Rs = 60;
[n,Wp] = ellipord(Wp,Ws,Rp,Rs)
% Returns n =4 Wp =0.0800
[b,a] = ellip(n,Rp,Rs,Wp);
freqz(b,a,512,1000);
title('n=4 Elliptic Lowpass Filter')
```