Laboratorium 11

1. Filtracja sygnałów, Filtry IIR

Filtry FIR, dla przypomnienia, charakteryzują się m.in. stosunkowo prostą budową (algorytmem) i liniową fazą w zakresie częstotliwości przepuszczanych.

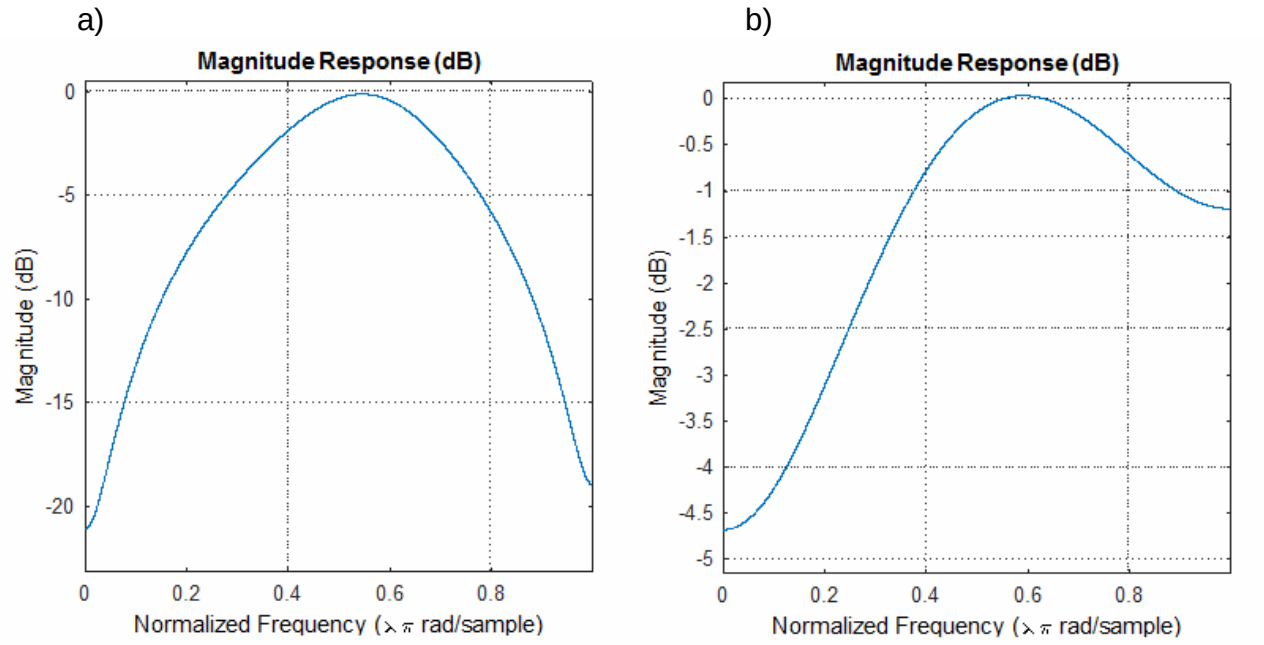
Drugą grupę stanowią filtry o nieskończonej odpowiedzi impulsowej, z angielska określane również skrótem IIR od angielskiej nazwy - Infinite Impulse Response. Filtry te są zasadniczo różne od filtrów FIR, ponieważ zawsze wymagają sprzężenia zwrotnego. O ile próbki sygnału wyjściowego filtru FIR zależą jedynie od przeszłych wartości sygnału wejściowego, to każda próbka wyjściowa filtru IIR zależy od poprzednich próbek sygnału wejściowego i poprzednich próbek sygnału wyjściowego. Stosowanie sprzężenia zwrotnego, czyli próbek wyjściowych podawanych z powrotem na wejście, jest zarówno korzystne, jak i niekorzystne.

Tak jak we wszystkich systemach ze sprzężeniem zwrotnym zmiany sygnału wejściowego filtru IIR mogą, w niektórych przypadkach, spowodować powstanie na wyjściu niestabilności i oscylacji o nieskończonym czasie trwania. Stąd też wzięła się nazwa "nieskończona odpowiedź impulsowa", czyli możliwość istnienia na wyjściu niezerowych próbek o nieskończonym czasie trwania w sytuacji, gdy na wejście filtru nie podawany jest żaden sygnał (lub próbki zerowe).

Jak się można więc domyślić już z tego opisu, struktura filtrów IIR nie będzie już taka prosta, jak filtrów FIR, a to oznacza, że są one trudniejsze do projektowania i analizy. Oprócz tego filtry IIR nie zapewniają, tak jak to było z filtrami FIR, liniowej fazy w zakresie przenoszenia.

Po cóż więc stosować i w ogóle zajmować się filtrami, które na pierwszy rzut oka mają same wady w stosunku do prostszych i łatwiejszych w projektowaniu filtrów FIR? Filtry IIR są po prostu bardziej efektywne, niż filtry FIR, czyli wymagają mniejszej liczby mnożeń dla wyliczenia pojedynczej próbki sygnału wyjściowego, przy zapewnieniu wymaganej charakterystyki częstotliwościowej. Ze sprzętowego punktu widzenia oznacza to, że filtry IIR mogą być bardzo szybkie, a więc mogą działać w czasie rzeczywistym, operując przy znacznie wyższych częstotliwościach próbkowania, niż filtry FIR.

Filtr IIR nie dość, że wymaga mniej, niż połowy mnożeń w stosunku do filtru FIR, to jeszcze ma mniejsze nierównomierności charakterystyki w paśmie przepustowym oraz większą stromość charakterystyki w paśmie przejściowym w porównaniu z filtrem o skończonej odpowiedzi impulsowej (Rys. 1).



Rys. 1. Porównanie filtru a) IIR, b) FIR

```
m = [0  1   1   0];
f = [0 0.5 0.6 1];
[b,a] = yulewalk(4,f,m);
fvtool(b,a)

b = fir1(4,[0.5 0.6]);
fvtool(b)
```

Zysk z powodu zmiany filtru z FIR na IIR wydaje się oczywisty. Aby wymusić dużą stromość obszaru przejściowego musieliśmy projektować filtry FIR o bardzo długiej odpowiedzi impulsowej. Im dłuższa będzie odpowiedź, tym bardziej charakterystyka częstotliwościowa będzie zbliżać się do ideału.

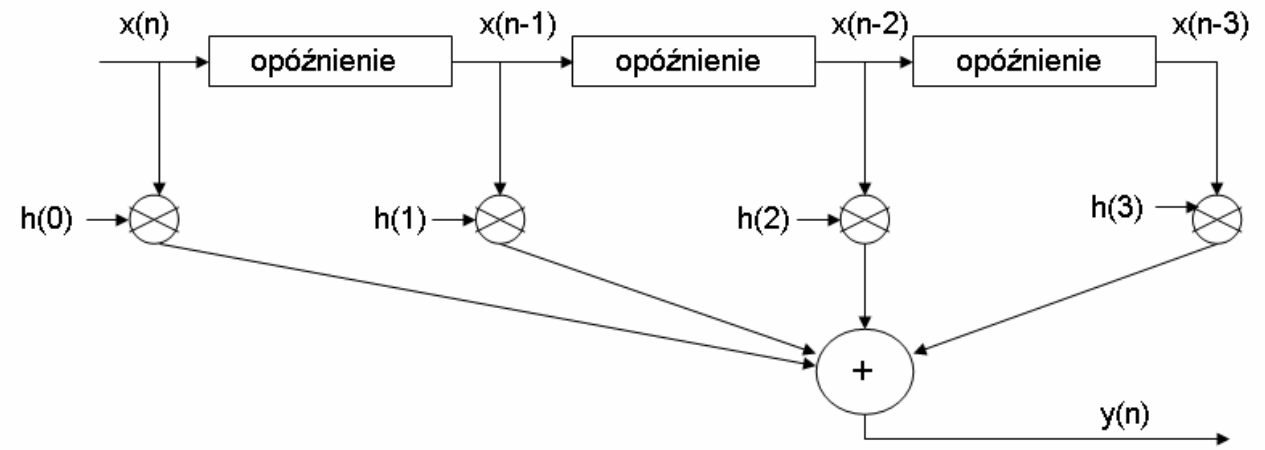
W aspekcie sprzętowym maksymalna liczba ogniw filtru FIR, którą może mieć filtr, zależy od tego jak szybko nasz sprzęt będzie w stanie wykonać określoną liczbę mnożeń i sumowań, wymaganą do wyznaczenia pojedynczej próbki sygnału wyjściowego, zanim na wejściu pojawi się następna próbka wejściowa. Filtry IIR zaś mogą być projektowane tak, że długość odpowiedzi impulsowej może być większa, niż liczba ogniw filtru. Zatem filtry IIR mogą zapewnić znacznie lepszą filtrację przy określonej liczbie mnożeń dla jednej próbki sygnału wyjściowego, niż filtry FIR.

Czas najwyższy aby nieco bliżej przyjrzeć się filtrom IIR.

Jak już wiemy nazwa filtru IIR wynika stąd, że do obliczenia bieżącej próbki sygnału wyjściowego korzysta się z wartości przeszłych próbek sygnału wejściowego. Efekt jest

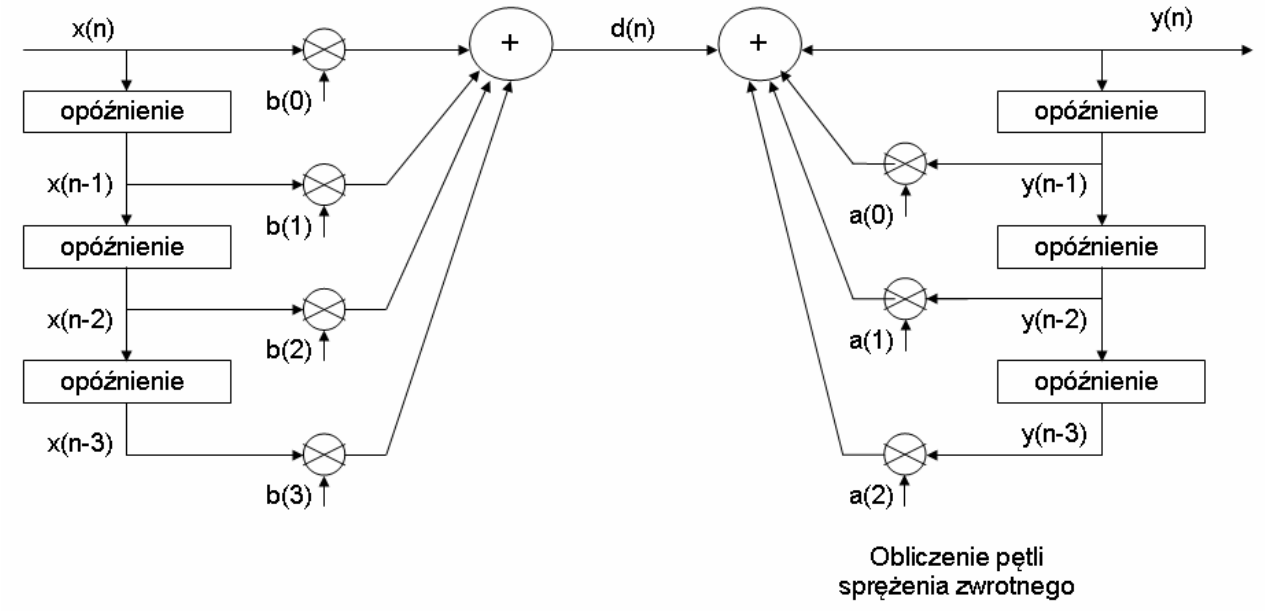
taki, że pewien skończony ciąg niezerowych wartości wejściowych może spowodować pojawienie się na wyjściu filtru IIR ciągu próbek o nieskończonym czasie trwania.

Zatem, jeśli sygnał wejściowy filtru IIR stałby się nagle ciągiem samych wartości zerowych, to sygnał wyjściowy mógłby - w pewnych określonych warunkach - już na zawsze pozostać niezerowym. Ta szczególna właściwość filtru IIR wynika ze sposobu jego realizacji, tzn. sprzężenia zwrotnego obejmującego bloki opóźnienia, mnożniki i sumatory.



Rys. 2. Struktura filtru FIR

Strukturę filtru IIR możemy niejako "wyprowadzić" ze struktury filtru FIR. Z rysunku 2 przypomnijmy sobie jak taka struktura cztero-ogniwowego filtru FIR wygląda.



Rys. 3. Struktura filtru IIR

Teraz, "dołączając" do struktury z rysunku 2 gałąź z pętlą sprzężenia zwrotnego, otrzymujemy filtr IIR (rysunek 3). Ponieważ filtry IIR mają dwa zestawy współczynników, stosujemy standardową notację $b(k)$ dla współczynników operujących na przeszłych próbkach wejściowych i $a(k)$ dla współczynników w pętli sprzężenia zwrotnego z rysunku 3. Równanie opisujące filtr IIR będzie miało postać:

$$y(n) = b(0)x(n) + b(1)x(n\text{-}1) + b(2)x(n\text{-}2) + b(3)x(n\text{-}3) + a(1)y(n\text{-}1) + a(2)y(n\text{-}2) + a(3)y(n\text{-}3)$$

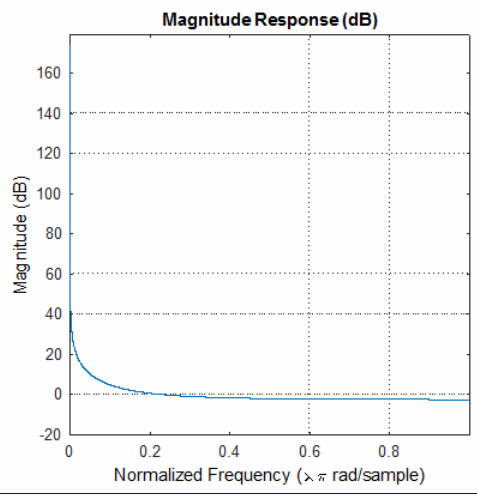
2. Projektowanie filtrów o nieskończonej odpowiedzi impulsowej (IIR)

W przypadku projektowania filtrów FIR wystarczyło zdefiniować funkcję transmitancji (czyli de facto określić charakterystykę częstotliwościową filtru), obliczyć odwrotną transformatę Fouriera tej funkcji, wynik transformaty poddać przesunięciu, i uzyskiwaliśmy odpowiedź impulsową filtru w dziedzinie czasu.

W przypadku filtru IIR pożądane współczynniki filtru $h(k)$ są dokładnie równe wartościom poszczególnych próbek odpowiedzi impulsowej. W przypadku filtrów IIR możemy postąpić tak samo (tzn. określić pożądaną funkcję transmitancji i obliczyć jej odwrotną transformatę Fouriera, aby uzyskać odpowiedź impulsową filtru), jednakże zła wiadomość jest taka, że nie ma bezpośredniej metody obliczenia współczynników $a(k)$ i $b(k)$ z odpowiedzi impulsowej. A więc stosunkowo proste techniki projektowania filtrów FIR nie mogą być zastosowane do projektowania filtrów IIR.

Nie ma to jednak wielkiego znaczenia, bo obecnie nikt nie projektuje filtrów cyfrowych z kartką i ołówkiem w ręku, wykonując mniej lub bardziej żmudne obliczenia. Obecnie wszystko to wykonuje komputer (za pomocą odpowiedniego oprogramowania), a nasze zadanie ogranicza się tylko do wpisania pożądanych parametrów, jakimi ma wykazywać się nasz filtr.

```
b = [1 -0.5];
a = [1 -0.999999999];
fvtool(b,a)
```



Z uwagi na to, że w filtrach IIR stosuje się sprzężenie zwrotne, przy implementacji zaprojektowanego filtru mogą występować problemy ze stabilnością (czyli wspomniane już występowanie niezerowych próbek na wyjściu filtru w sytuacji gdy na wejściu nie ma sygnału lub jest on zerowy - wszak projektujemy filtr, a nie generator, więc chcemy, aby na wyjściu również był sygnał zerowy, gdy na wejściu są same 0).

Innymi problemami, z jakimi możemy się spotkać przy implementacji, są zniekształcenia charakterystyki częstotliwościowej. Takie problemy występują, ponieważ jesteśmy zmuszeni stosować dwójkową (czyli 0 lub 1 - wszak operujemy w domenie cyfrowej) reprezentację współczynników filtru i wyników pośrednich. A liczby dwójkowe mają tę paskudną przypadłość, że mają skończoną liczbę bitów. Efekty skończonej długości słowa są szczególnie dotkliwe, jeśli wielomian funkcji transmitancji $H(z)$ jest wysokiego rzędu, tj. jeśli filtr ma dużą liczbę elementów opóźniających.

Istnieją trzy główne rodzaje błędów pochodzących od skończonej długości słowa, które utrudniają implementację filtrów IIR:
- reprezentacja współczynników przy użyciu zmniejszonej liczby bitów
- błędy przepełnienia (ang. overflow)
- błędy zaokrąglenia (ang. roundoff)

Króciutko kilka zdań o tych błędach i jak im zapobiegać.

Przy obliczaniu współczynników filtru uzyskuje się bardzo dużą dokładność, np. ułamkowe wartości współczynników filtru mogą mieć dokładność do ośmiu cyfr po przecinku i większą. Aby zapisać lub zapamiętać współczynniki o takiej dokładności potrzebować będziemy 32 bitów (cyfra dziesiętna potrzebuje ich minimum 4).

Co więc stanie się, jeśli do zapamiętania współczynników i wyników obliczeń dysponujemy rejestrami 4-bitowymi? Wtedy, np. zamiast obliczonego współczynnika $b(n)$ = 0,20482712, będziemy zmuszeni użyć współczynnika $b(n)$ = 0,2. To spowoduje przesunięcie biegunów i zer filtru na płaszczyźnie "z", co nie dość że spowoduje nam zniekształcenia charakterystyki częstotliwościowej, to jeszcze może przesunąć bieguny poza obszar stabilności (z filtru zrobi się generator przebiegów losowych).

Jak się okazuje filtry wyższych rzędów są bardziej wrażliwe na dokładność współczynników i dla takich filtrów reprezentacja współczynników ze zmniejszoną liczbą bitów jest najbardziej prawdopodobną przyczyną pojawienia się niemożliwych do zaakceptowania zniekształceń charakterystyki częstotliwościowej.

Przepełnienie, drugi efekt skończonej długości słowa, pojawia się, gdy wynik operacji arytmetycznej jest zbyt duży, aby mógł być przechowywany w rejestrach o stałej długości. Bez wprowadzenia zabezpieczeń pozwalających na obsługę błędów przepełnienia mogą pojawić się duże błędy nieliniowe w próbkach wyjściowych filtru - często w postaci oscylacji przepełnienia (ang. overflow oscillations).

Istnieje kilka sposobów redukowania szkodliwych skutków błędów reprezentacji współczynników ze zmniejszoną liczbą bitów i zjawiska oscylacji sygnału wyjściowego.

Możemy zwiększyć długość rejestrów sprzętowych, przechowujących wyniki pośrednie obliczeń. Aby uniknąć pojawienia się sygnału wejściowego o samych zerach niekiedy dodaje się do sygnału wejściowego niewielki sygnał zakłócający **dither**.

Sygnał ten jest ciągiem liczb pseudolosowych o małych wartościach. Ten sposób zmniejsza jednak efektywny stosunek sygnał/szum w sygnale wyjściowym filtru. Ostatecznie, aby uniknąć zjawiska oscylacji, możemy po prostu zastosować filtry FIR, które z definicji mają odpowiedź impulsową o skończonej długości i nie mają sprzężenia zwrotnego, nie mogą zatem generować żadnych oscylacji.

Jeżeli chodzi o oscylacje wywołane przepełnieniem, to możemy wyeliminować niebezpieczeństwo ich powstawania, jeżeli zwiększymy długość słowa, tak aby w filtrze IIR nigdy nie występowało przepełnienie w czasie sumowań. Jeżeli implementuje się filtr na sprzęcie o architekturze zmiennoprzecinkowej, to okazuje się, że stosowanie danych w reprezentacji zmiennoprzecinkowej istotnie ogranicza możliwość powstawania błędów przepełnienia i oscylacji.

Najpopularniejszym sposobem minimalizowania błędów wynikających ze skończonej długości słowa jest projektowanie struktury kaskadowej lub równoległej filtrów niskiego rzędu. Czyli np. zamiast stosować filtr 6. rzędu łączymy 3 filtry rzędu drugiego, bądź kaskadowo (szeregowo), bądź równolegle.

3. Porównanie filtrów FIR i IIR

Zła wiadomość jest taka, że nie ma takiej jednoznacznej odpowiedzi, który filtr jest lepszy. Nawet gdybyśmy chcieli wybrać jeden z filtrów do konkretnych struktur cyfrowych też trudno jednoznacznie orzec, że ten będzie OK, a tamten nie za bardzo. Spróbujmy jednak wskazać kilka czynników, które pozwolą nam dokonać takiego wyboru.

Tab. 1. Porównanie filtrów FIR i IIR

Właściwość	IIR	FIR
Liczba wymaganych mnożeń	Mała	Duża
Prawdopodobieństwo wystąpienia błędów przepełnienia	Może być duża dla struktury bezpośredniej	Bardzo małe
Stabilność	Musi być projektowana	Zagwarantowana
Liniowość fazy	Nie	Zagwarantowana
Sprzętowe wymagania dla pamięci	Małe	Duże
Złożoność sprzętowa układu sterowania filtru	Umiarkowana	Mała
Dostępność oprogramowania wspomagającego projektowanie	Dobra	Bardzo dobra
Łatwość projektowania	Umiarkowana	Prosta

Po pierwsze możemy przyjąć, że nie ma różnicy w stopniu trudności procedury projektowania dla obu typów filtrów, zwłaszcza że w 99,999% wykorzystuje się do tego celu

komputer wraz z odpowiednim oprogramowaniem (a ich obsługa jest bardzo podobna, zarówno w przypadku projektowania filtru IIR, jak i FIR). Wychodząc ze sprzętowego punktu widzenia, gdzie różnice pomiędzy filtrami FIR i IIR są zasadnicze, nasz wybór musi wynikać z tych właściwości filtru, które są najbardziej i najmniej dla nas istotne.

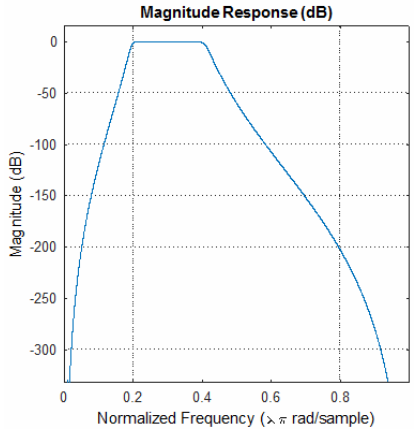
Na przykład, jeśli wymagany jest filtr o dokładnie liniowej fazie, to jedynym poprawnym wyborem będzie filtr FIR. Jeżeli jednak wymagane jest, aby filtr pracował z bardzo wielką częstotliwością, a dopuszczalna jest niewielka nieliniowość fazy, możemy skłonić się ku filtrom IIR z ich zredukowaną liczbą operacji mnożenia dla jednej próbki sygnału wyjściowego.

W celu całościowego ogarnięcia właściwości obu filtrów proszę spojrzeć na Tabelę 1.

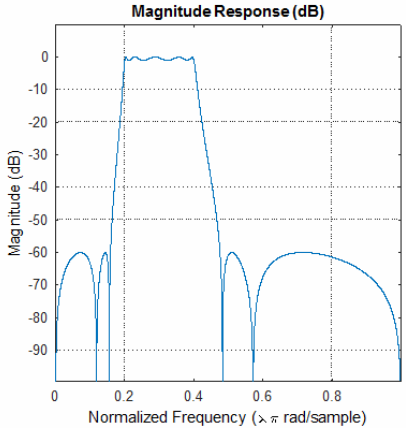
Przykład 1

Zaprojektować przykładowe filtry IIR: filtr Butterwortha `– 'buttord',`
filtr eliptyczny `–'ellipord'`

```
[n,Wn] = buttord([1000
2000]/5000,[500
2500]/5000,1,60)
[b,a] = butter(n,Wn);
fvtool(b,a)
```

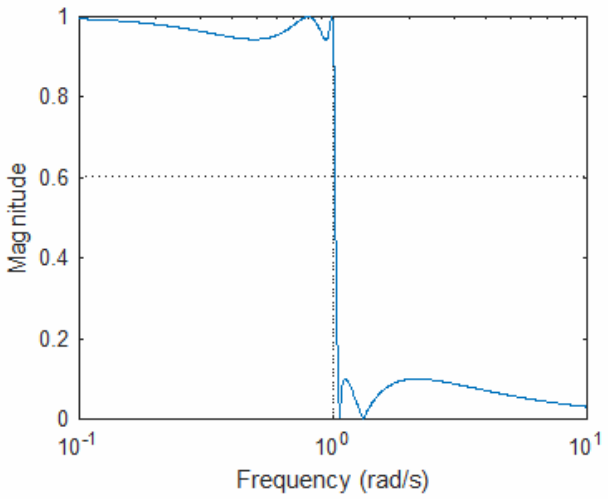



```
[n,Wn] = ellipord([1000
2000]/5000,[500
2500]/5000,1,60)
[b,a] = ellip(n,1,60,Wn);

fvtool(b,a)
```



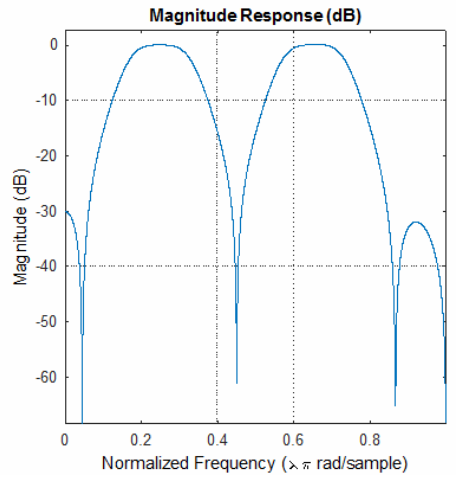
Przykład 2
Zaprojektować przykładowe filtry IIR:

```
[z,p,k] = ellipap(5,0.5,20);
w = logspace(-1,1,1000);
h =
freqs(k*poly(z),poly(p),w);
semilogx(w,abs(h)), grid
xlabel('Frequency (rad/s)')
ylabel('Magnitude')
```



Przykład 3
Zapoznać się z funkcją `yulewalk()`. Funkcja ta projektuje rekursywne filtry cyfrowe IIR.

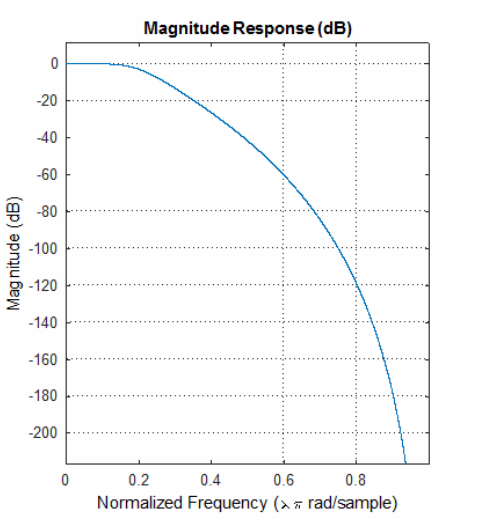
```
m = [0   0   1   1   0   0   1
1   0 0];
f = [0 0.1 0.2 0.3 0.4 0.5 0.6
0.7 0.8 1];
[b,a] = yulewalk(10,f,m);

fvtool(b,a)
```



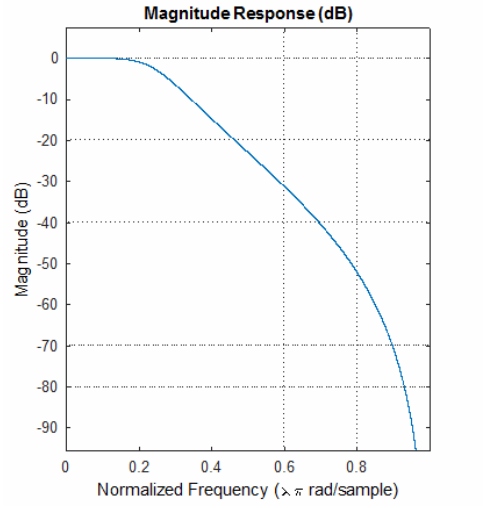
Przykład 4

Zapoznać się z funkcją `maxflat()`. Filtr Butterwortha uogólniony.

```
n = 10;
m = 2;
Wn = 0.2;

[b,a] = maxflat(n,m,Wn);
fvtool(b,a)
```



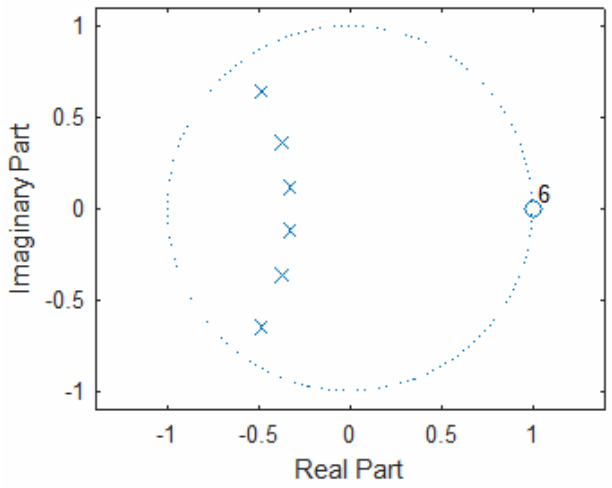

```
[b,a] = maxflat(3,3,0.25)
fvtool(b,a)
```



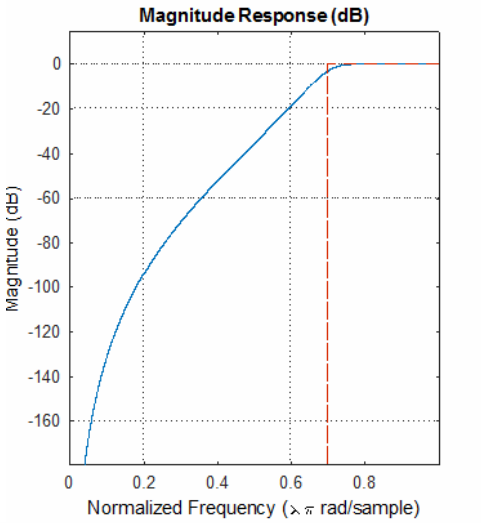
Przykład 5

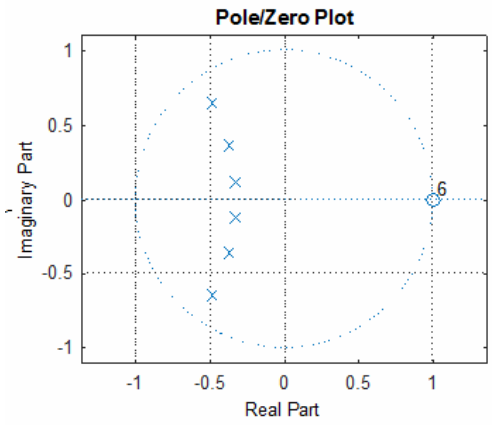
Zaprojektuj filtr Butterworth górnoprzepustowy IIR 6-rzędowy. Określ znormalizowaną częstotliwość 3-dB równą 0,7. Sprawdź, czy filtr jest stabilny? Użyć funkcji `isstable()`. Jeśli biegun "znak x" znajdzie się poza kołem wówczas mamy filtr niestabilny. Jeśli biegun będzie w kole wówczas mamy filtr stabilny.

```
[z,p,k] =
butter(6,0.7,'high');
SOS = zp2sos(z,p,k);
flag = isstable(SOS)

zplane(z,p)


% wynik flag = 1
```




```
d =
designfilt('highpassi
ir','DesignMethod','b
utter','FilterOrder',
6, ...

'HalfPowerFrequency',
0.7);
dflg = isstable(d)

fvtool(d)
zplane(d)



% wynik flag = 1
```

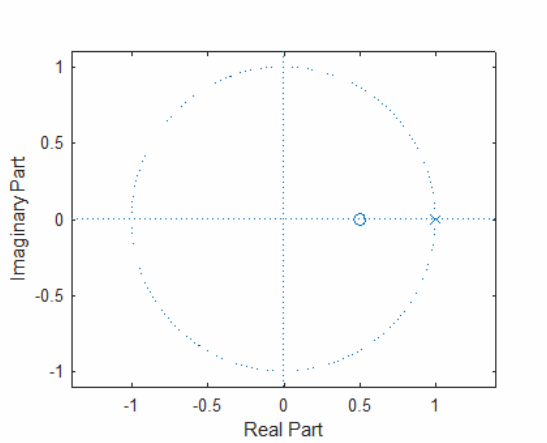




Przykład 6

Sprawdzić stabilność filtru IIR `b = [1 -0.5]; a = [1 -0.999999999];`

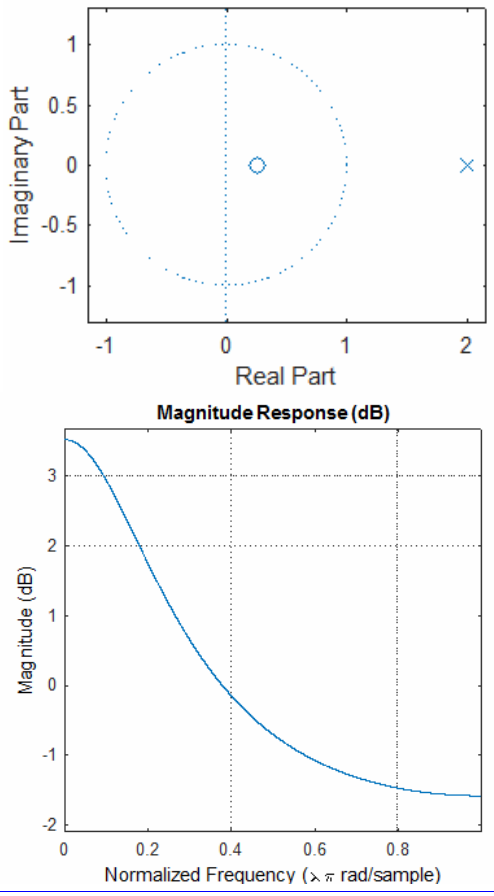
```
b = [1 -0.5];
a = [1 -0.999999999];
act_flag1 =
isstable(b,a)

zplane(b,a)



% wynik flag = 1
```

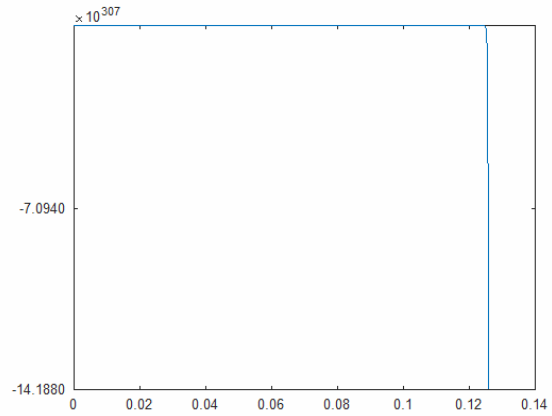


Przykład 7

Sprawdzić stabilność filtru IIR `b = [2 -0.5]; a = [1 -2];`
Przefiltrować sygnał `chirp` filtrem IIR `b = [2 -0.5]; a = [1 -2]`
Narysować przebieg czasowy i częstotliwościowy przefiltrowanego sygnału.

```
b = [2 -0.5];
a = [1 -2];
act_flag1 = isstable(b,a)

zplane(b,a)
% wynik flag = 0

load chirp
t = (0:length(y)-1)/Fs;
% 1.6 sekundy
xfft=abs(fft(y));
xfft=xfft/13129;
x1=1:1:6564;
bar(x1(1:6564), xfft(1:6564));
axis([0,6564, 0,0.01]) ;
plot(t,y);
fvtool(b,a)
outsignal = filter(b,a,y);
plot(t, outsignal);

xfft=abs(fft(outsignal));
xfft=xfft/13129;
x1=1:1:6564;
bar(x1(1:6564), xfft(1:6564));
axis([0,6564, 0,0.01]) ;
```





Inf
NaN
NaN
NaN
NaN
NaN
NaN
NaN
NaN
....
....

Zadania do wykonania

W sprawozdaniu powinny znaleźć się:
1) Informacje na temat w jaki sposób projektujemy filtry IIR.
2) Różnice między filtrami FIR i IIR.
3) Wykonane zadania - skrypty w m.plikach oraz otrzymane wykresy.
4) Wnioski z przeprowadzonych zadań.

Zad. 1
Korzystając z Przykładu 4 zaprojektować filtr IIR:
```
[b,a] = maxflat(4,1,0.3)
```
oraz
```
maxflat(4,'sym',0.3)
```

Zad2
Sprawdzić stabilność filtru o współczynnikach `b = [1 -0.5]; a = [1 -2]`

Zad3
Sprawdzić stabilność filtru o współczynnikach `b = [1 -0.1]; a = [-1 -0.1]`

Zad4
Sprawdzić stabilność filtru o współczynnikach
```
b = [0.9 -0.8];  a = [-0.9 -0.8]
```

Zad5
Sprawdzić stabilność filtru o współczynnikach
```
b = [0.9 -0.8 1 1];  a = [-0.9 -0.8 -1]
```

Zad6
Sprawdzić stabilność następującego filtru
```
m = [0   0   1   1   1   0   1   1   0 0];
f = [0 0.1 0.2 0.3 0.4 0.5 0.7 0.8 0.9 1];
```

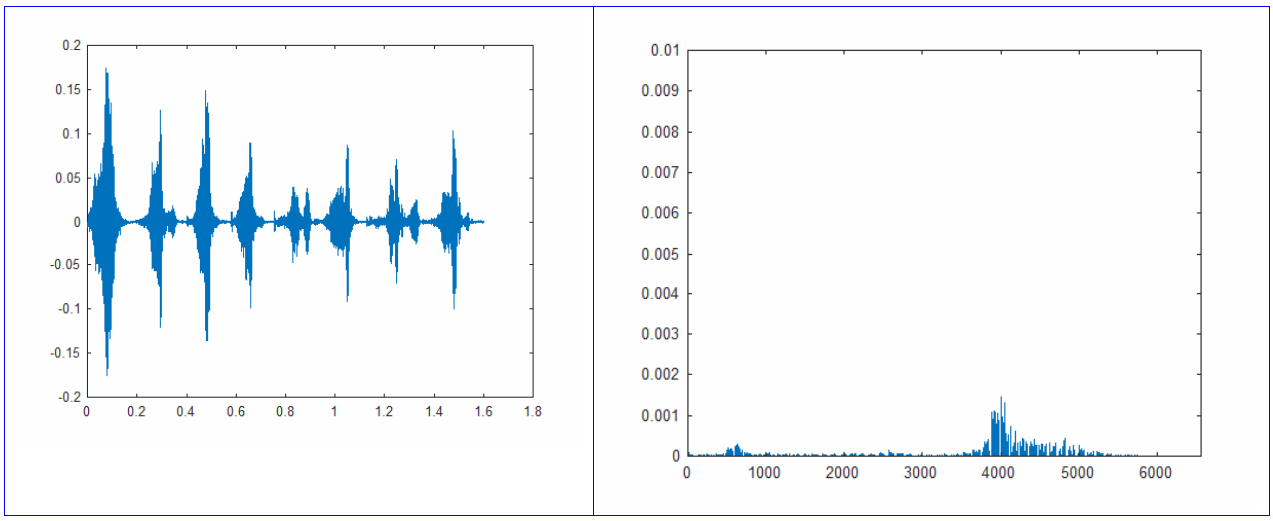
```
[b,a] = yulewalk(10,f,m);
```

Zad7

Przefiltrować sygnał `chirp` filtrem IIR `maxflat(4,1,0.3).` Narysować przebieg czasowy i częstotliwościowy przefiltrowanego sygnału.

Wskazówka

```
load chirp
t = (0:length(y)-1)/Fs;   % 1.6 sekundy

xfft=abs(fft(y));
xfft=xfft/13129;
x1=1:1:6564;
bar(x1(1:6564), xfft(1:6564));
axis([0,6564, 0,0.01]) ;
plot(t,y);
```



Zad8

Przefiltrować sygnał `chirp` filtrem IIR

```
m = [0   0   0   0   0   1   1   1   1 1];
f = [0 0.1 0.2 0.3 0.4 0.5 0.7 0.8 0.9 1];
[b,a] = yulewalk(10,f,m);
```

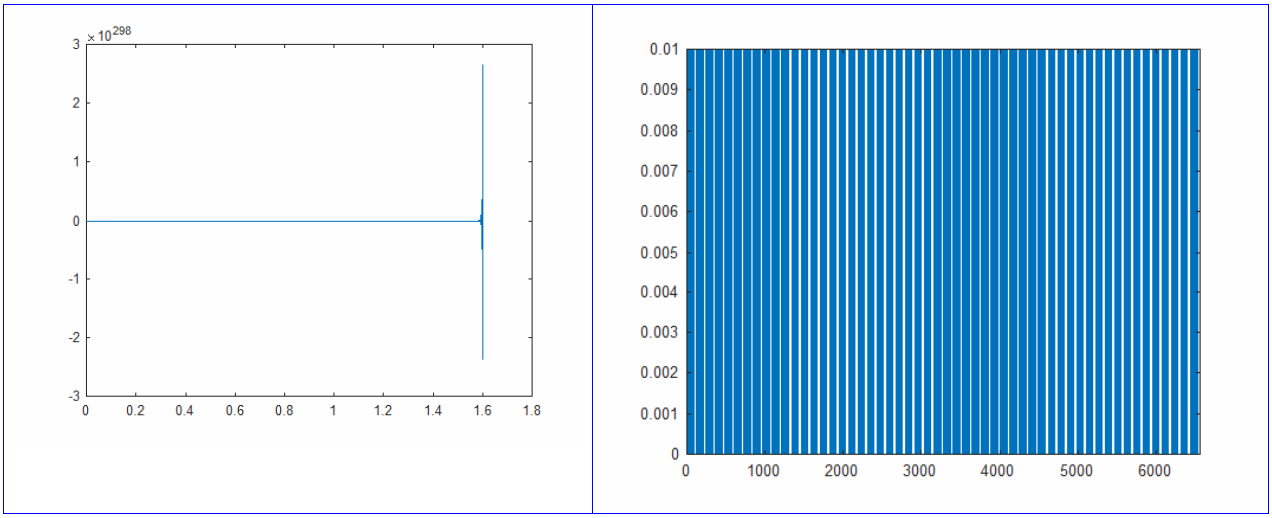
Narysować przebieg czasowy i częstotliwościowy przefiltrowanego sygnału.

Zad 9

Przefiltrować sygnał `chirp` filtrem IIR. Uwaga filtr jest niestabilny.

```
b = [0.9 -0.8 1 1];  a = [-0.9 -0.8 -1];
```

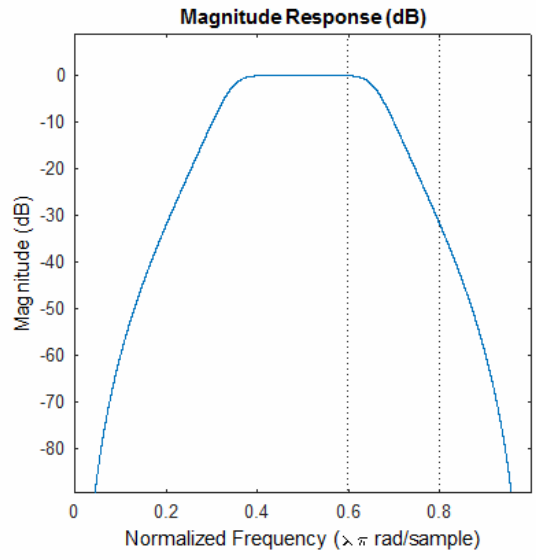
Narysować przebieg czasowy i częstotliwościowy przefiltrowanego sygnału.
Narysować charakterystykę częstotliwościową filtru.

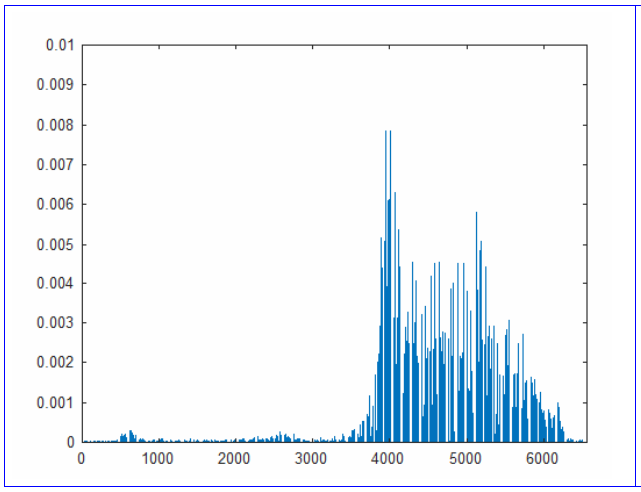


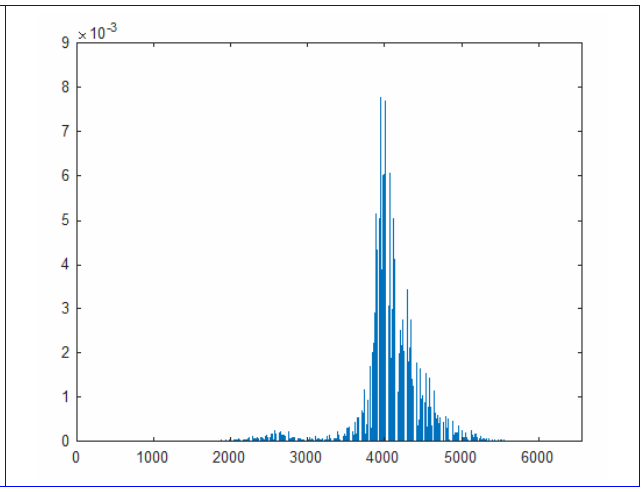
Zad 10
Przefiltrować sygnał `chirp` filtrem IIR, Butterwortha. Użyć funkcji `buttord` i `butter`. z parametrami

```
[n,Wn] = buttord([2000 3000]/5000,[500 4500]/5000,1,60)
```

Narysować przebieg częstotliwościowy przefiltrowanego sygnału.
Narysować charakterystykę częstotliwościową filtru.





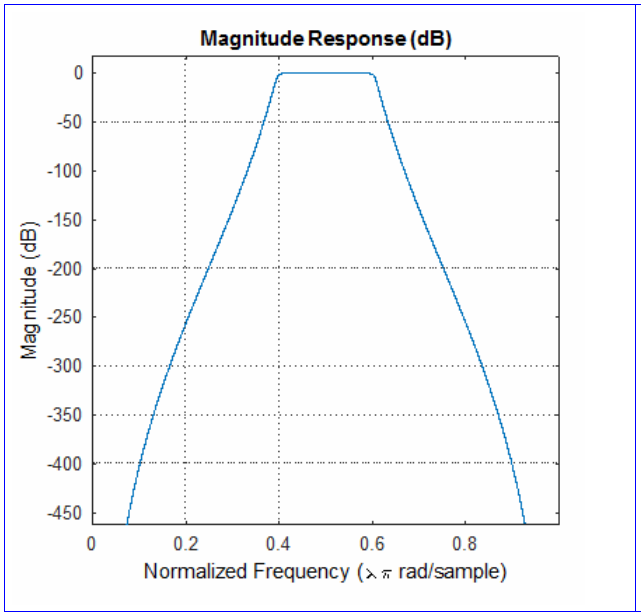


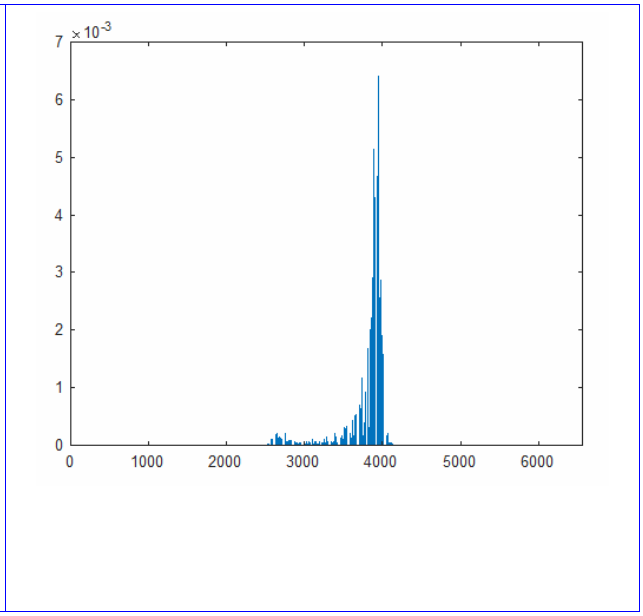
Zad 11

Przefiltrować sygnał `chirp` filtrem IIR, Butterwortha. Użyć funkcji `buttord` i `butter`. z parametrami

```
[n,Wn] = buttord([2000 3000]/5000,[1800 3200]/5000,1,60)
```

Narysować przebieg częstotliwościowy przefiltrowanego sygnału.
Narysować charakterystykę częstotliwościową filtru.





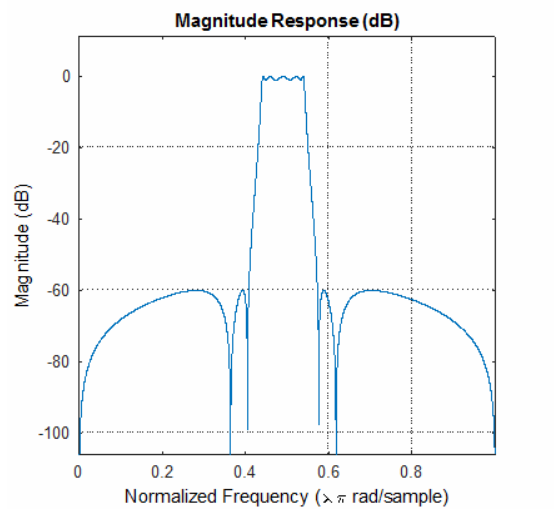
Zad 12

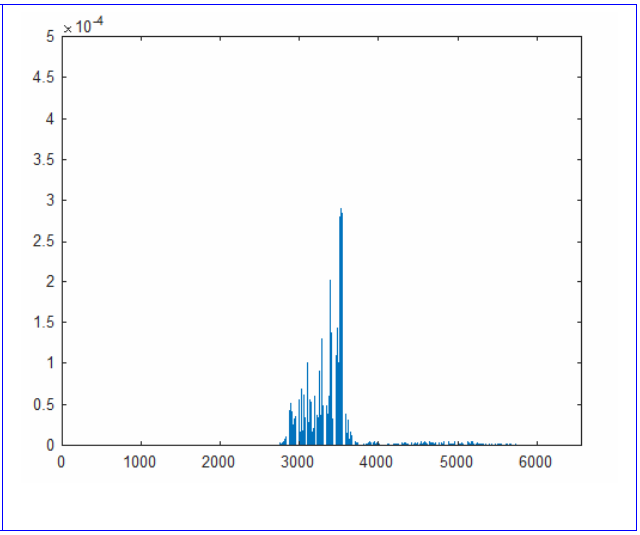
Przefiltrować sygnał `chirp` filtrem IIR, Butterwortha. Użyć funkcji `ellipord i ellip`. z parametrami

```
[n,Wn] = ellipord([2200 2700]/5000,[2000 2900]/5000,1,60)
```

Narysować przebieg częstotliwościowy przefiltrowanego sygnału.

Narysować charakterystykę częstotliwościową filtru.



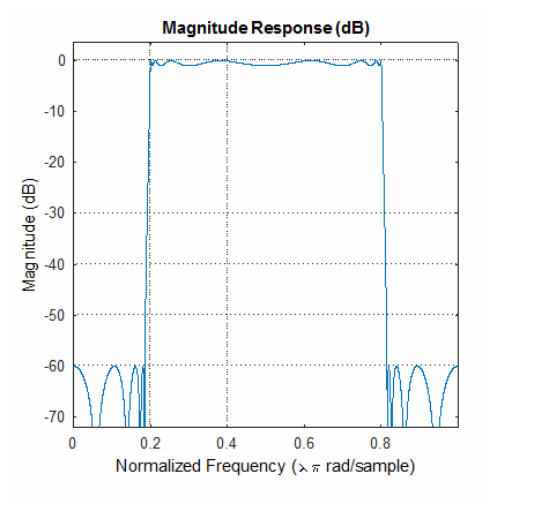


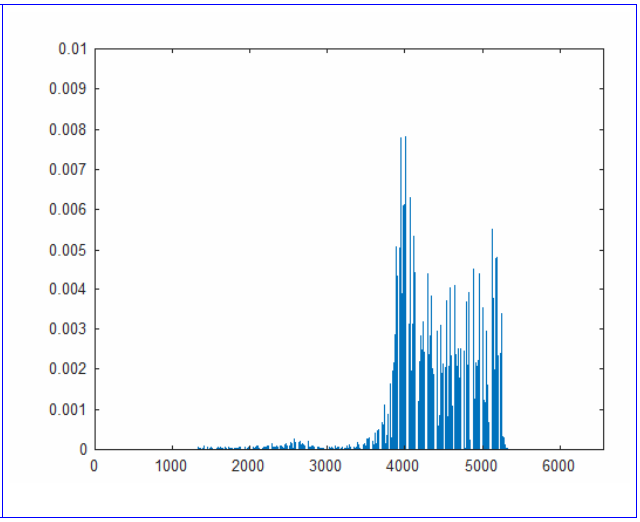
Zad 13

Przefiltrować sygnał `chirp` filtrem IIR, eliptycznym. Użyć funkcji `ellipord i ellip`.
z parametrami

```
[n,Wn] = ellipord([1000 4000]/5000,[900 4100]/5000,1,60)
```

Narysować przebieg częstotliwościowy przefiltrowanego sygnału.
Narysować charakterystykę częstotliwościową filtru.





Zad 14

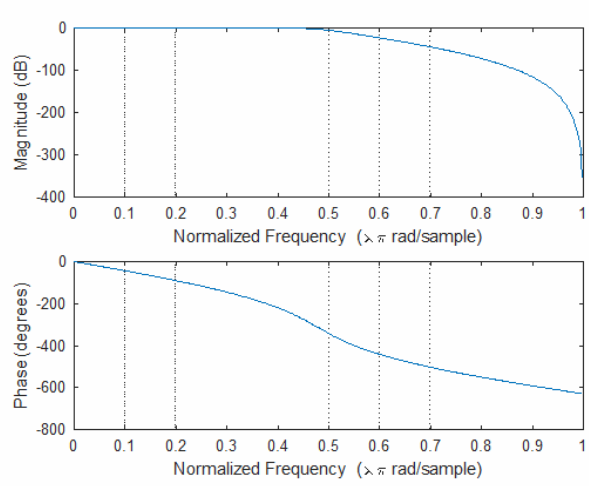
Zapoznać się z funkcją `sosfilt()`, filtrowanie drugiego rzędu.

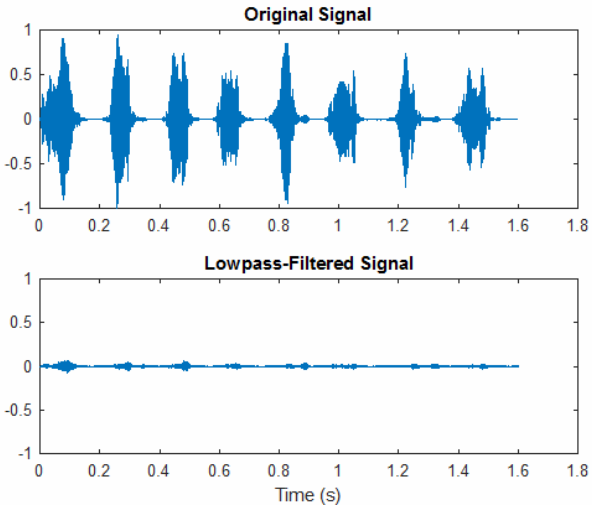
Przefiltrować sygnał `chirp` filtrem dolnoprzepustowym Butterwortha z zastosowaniem funkcji `butter(), sosfilt()`.

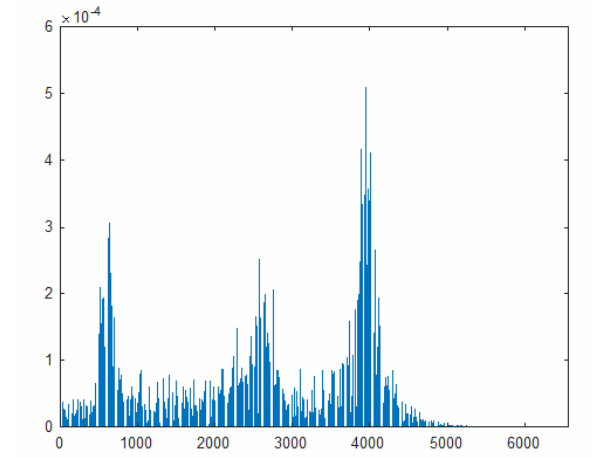
Program można rozpocząć od następującego kodu:

```
[zhi,phi,khi] = butter(7,0.48,'low');
soshi = zp2sos(zhi,phi,khi);
freqz(soshi)
```

Narysować charakterystykę częstotliwościową filtru.
Narysować przebieg czasowy i częstotliwościowy przefiltrowanego sygnału.





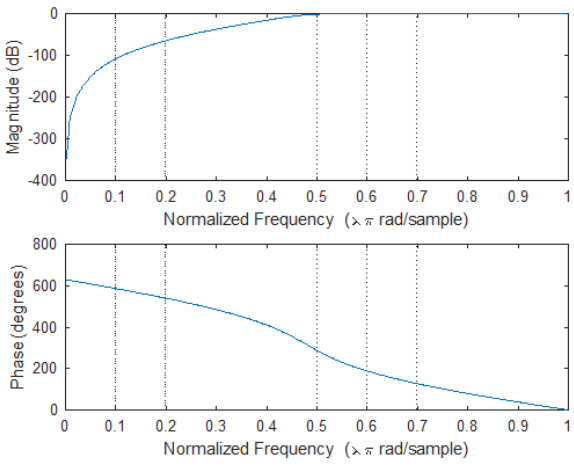


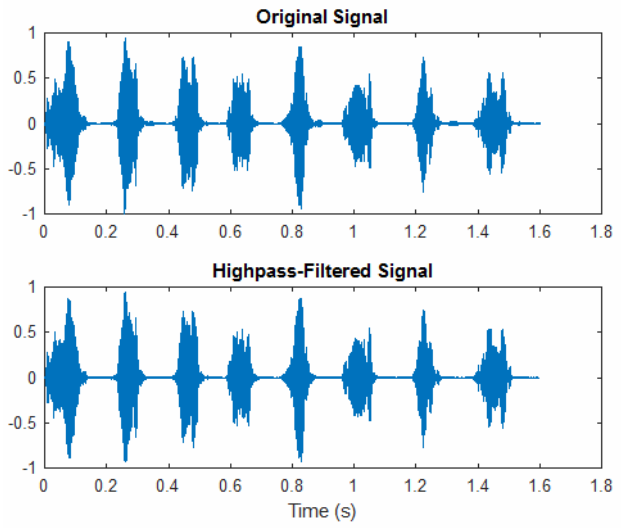
Zad 15

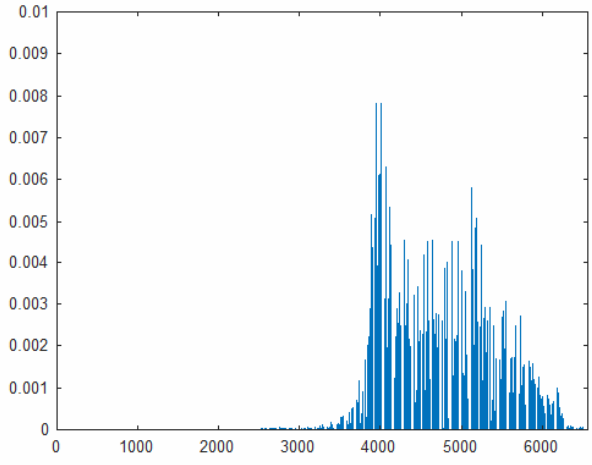
Zapoznać się z funkcją `sosfilt()`. Filtrowanie drugiego rzędu.

Przefiltrować sygnał `chirp` filtrem górnoprzepustowym Butterwortha z zastosowaniem funkcji `butter(), sosfilt()`.

Narysować charakterystykę częstotliwościową filtru.
Narysować przebieg czasowy i częstotliwościowy przefiltrowanego sygnału.







Pytania
1) Informacje na temat w jaki sposób projektujemy filtry IIR.

2) Różnice między filtrami FIR i IIR.
3) Kiedy bardziej wskazane jest użycie filtru FIR a kiedy filtru IIR?
4) Czy są jakieś różnice w budowie filtrów FIR i IIR.