

Przetwarzanie dźwięków i obrazów

Filtry cyfrowe

część 1:

FILTRY FIR

Opracowanie: Grzegorz Szwoch

Politechnika Gdańska, Katedra Systemów Multimedialnych

greg@multimed.org

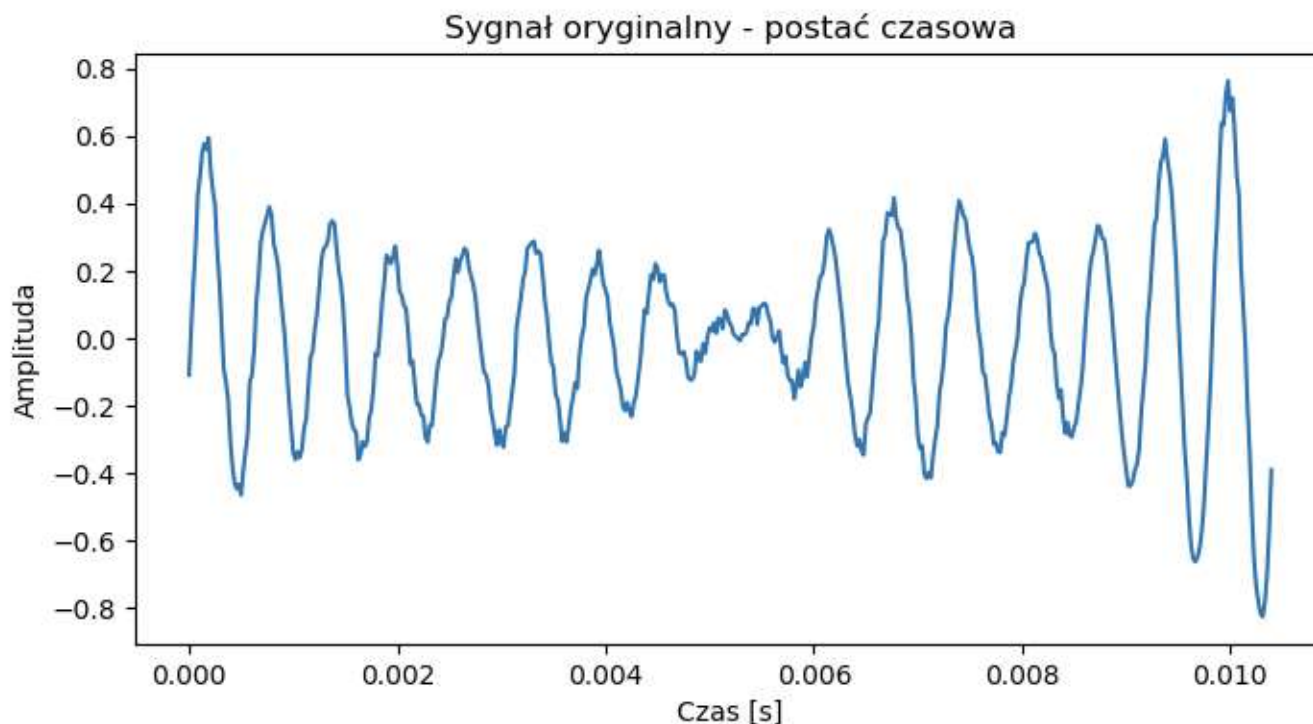
Zanim zaczniemy:

- Celem tego cyklu wykładów jest przedstawienie filtrów cyfrowych **od strony praktycznej**.
- Jeżeli pojawią się wzory i schematy, to służą one tylko ilustracji – nie uczyć się ich na pamięć!
- Projektowaniem filtrów zajmują się programy komputerowe.
- Ale musimy wiedzieć jakie dane im podać.
- Po zakończeniu cyklu wykładów, student powinien rozumieć: jak działają filtry cyfrowe i jak je zaprojektować.

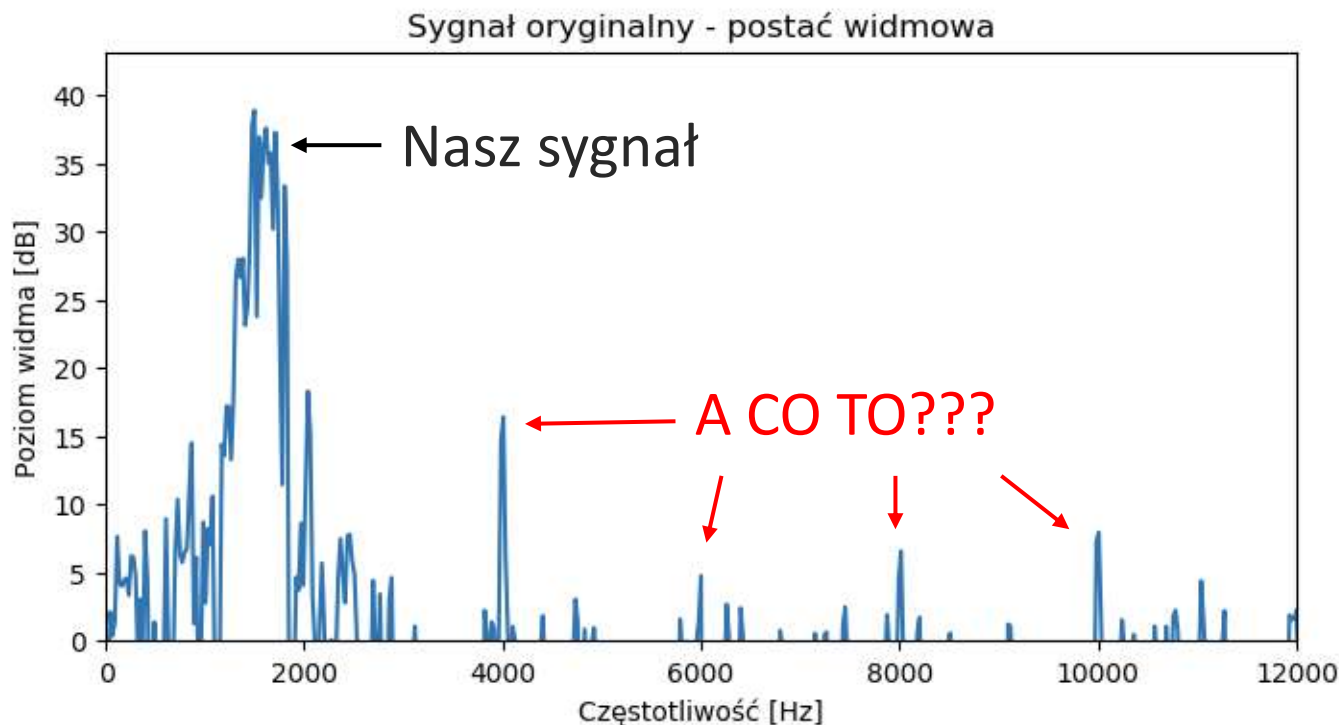
Przykład praktyczny nr 1.

Mamy sygnał z czujnika (może to być dźwięk).

Patrząc na wykres czasowy widzimy, że coś jest nie tak jak trzeba. Ale co?

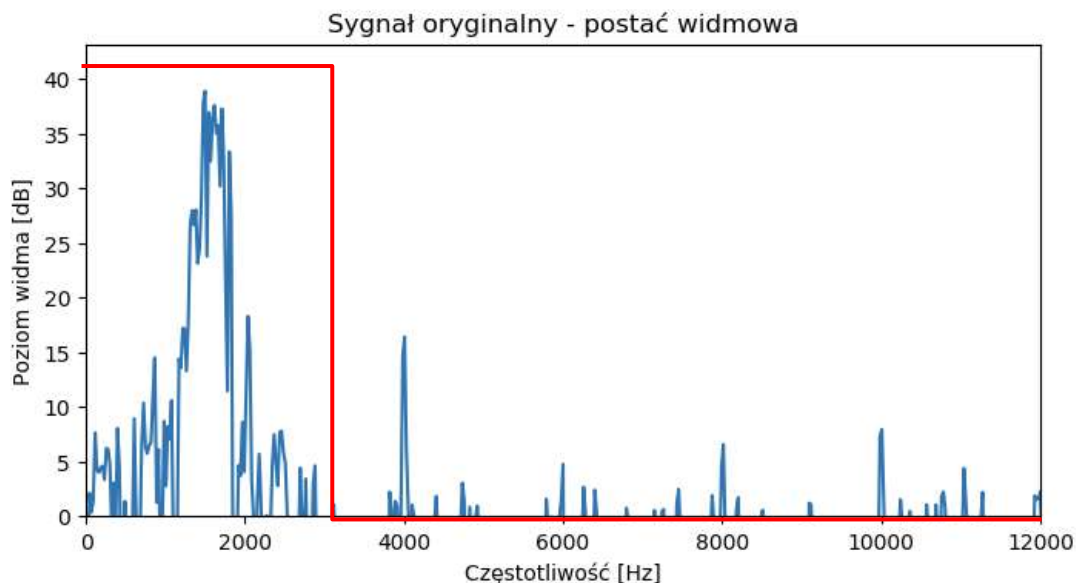


Musimy popatrzeć na widmo sygnału:



Zniekształcenia muszą zostać **odfiltrowane** z sygnału!

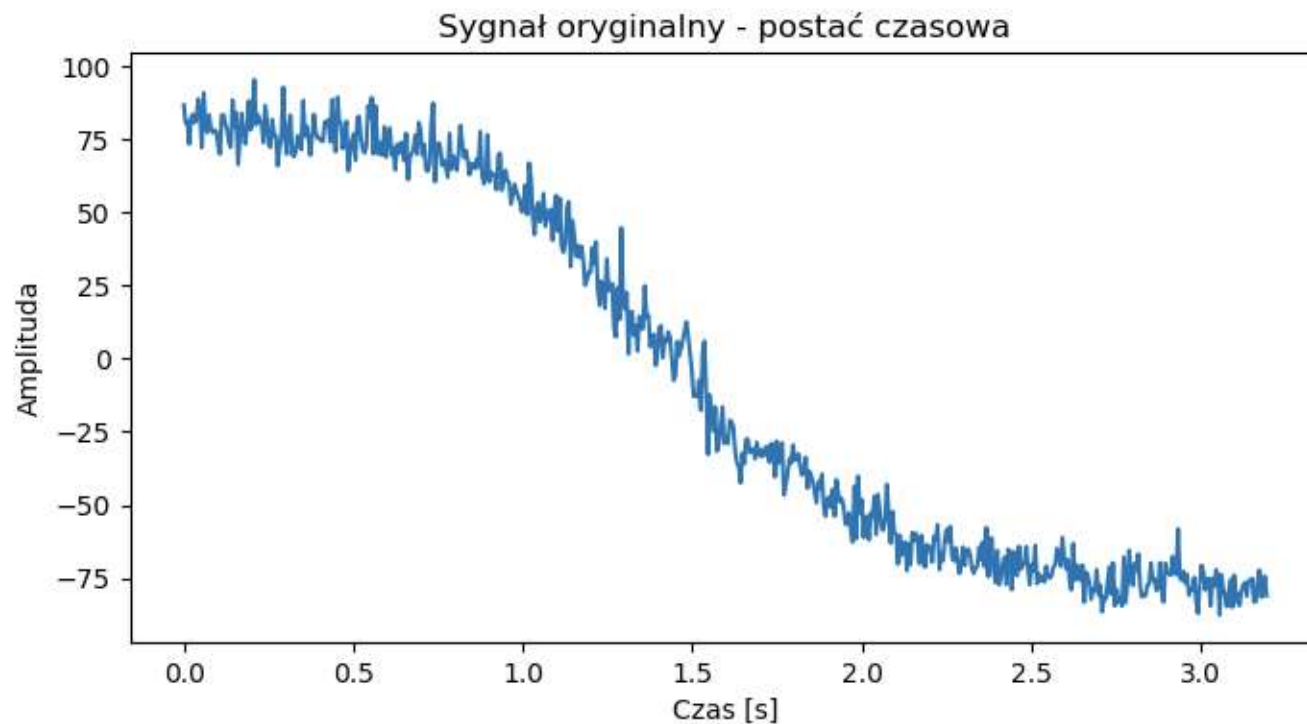
- **Filtr cyfrowy** jest algorytmem, który usuwa z sygnału niepożądane składowe widmowe.
- Najczęściej filtry działają w dziedzinie częstotliwości:
 - tłumią (filtrują) pewien zakres częstotliwości,
 - przepuszczają pozostałe częstotliwości bez zmian.



Do przykładu nr 1 wrócimy później.

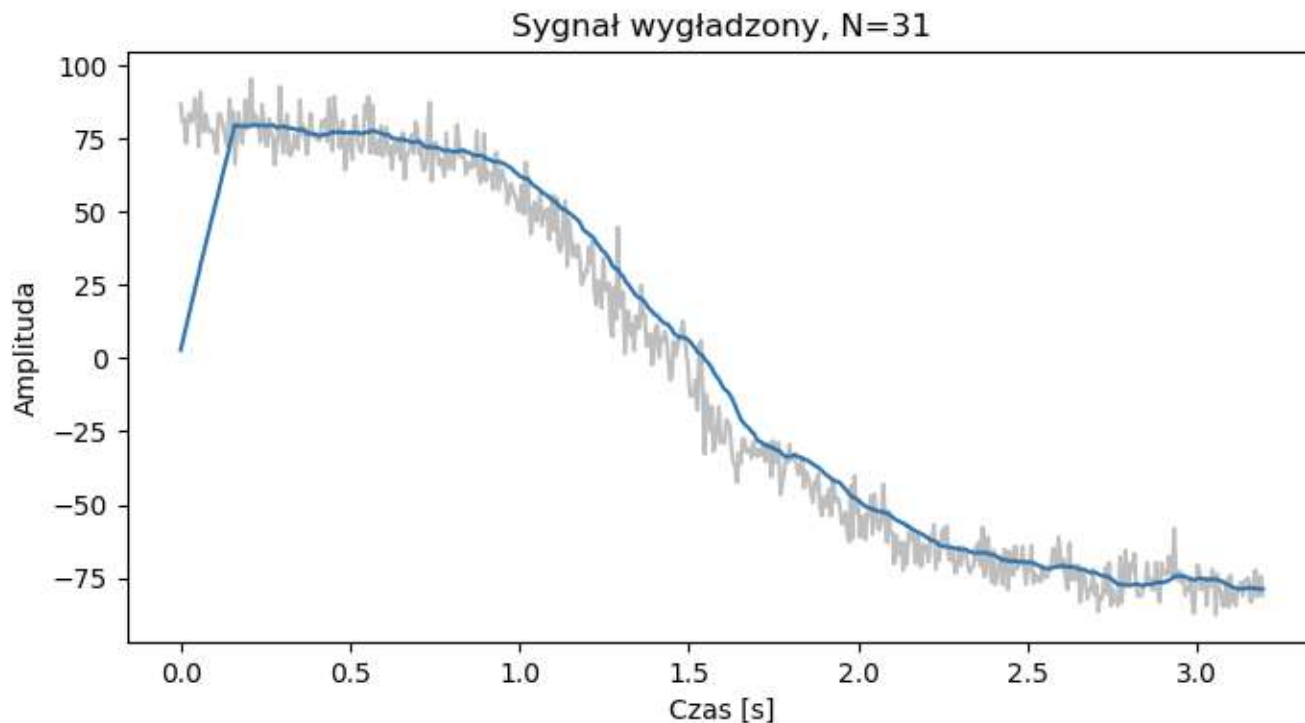
Przykład nr 2.

Zaszumiony sygnał – wymaga wygładzenia.

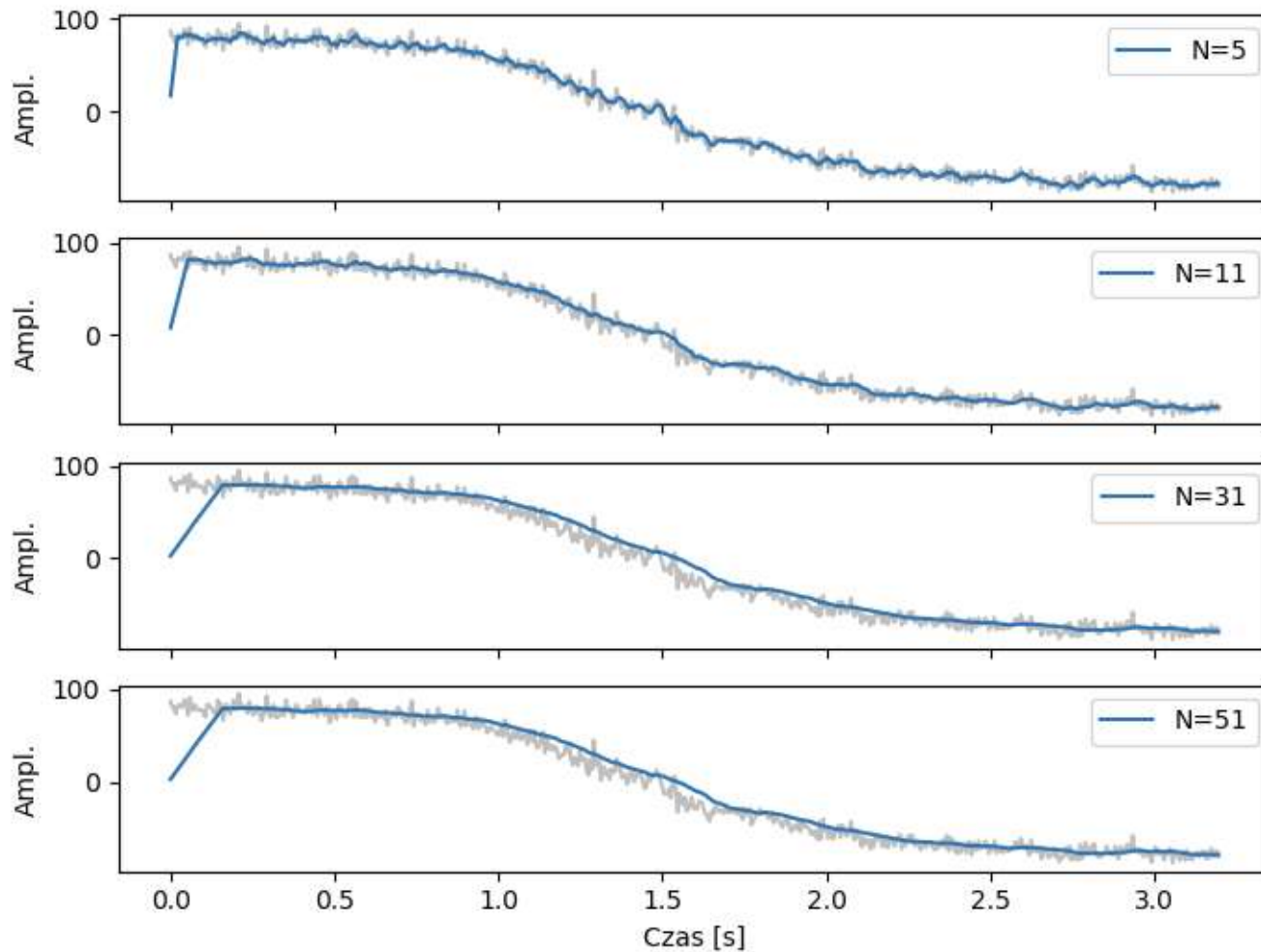


Aby wygładzić sygnał, każdą próbkę zastępujemy średnią z N ostatnich próbek sygnału (wliczając bieżącą)

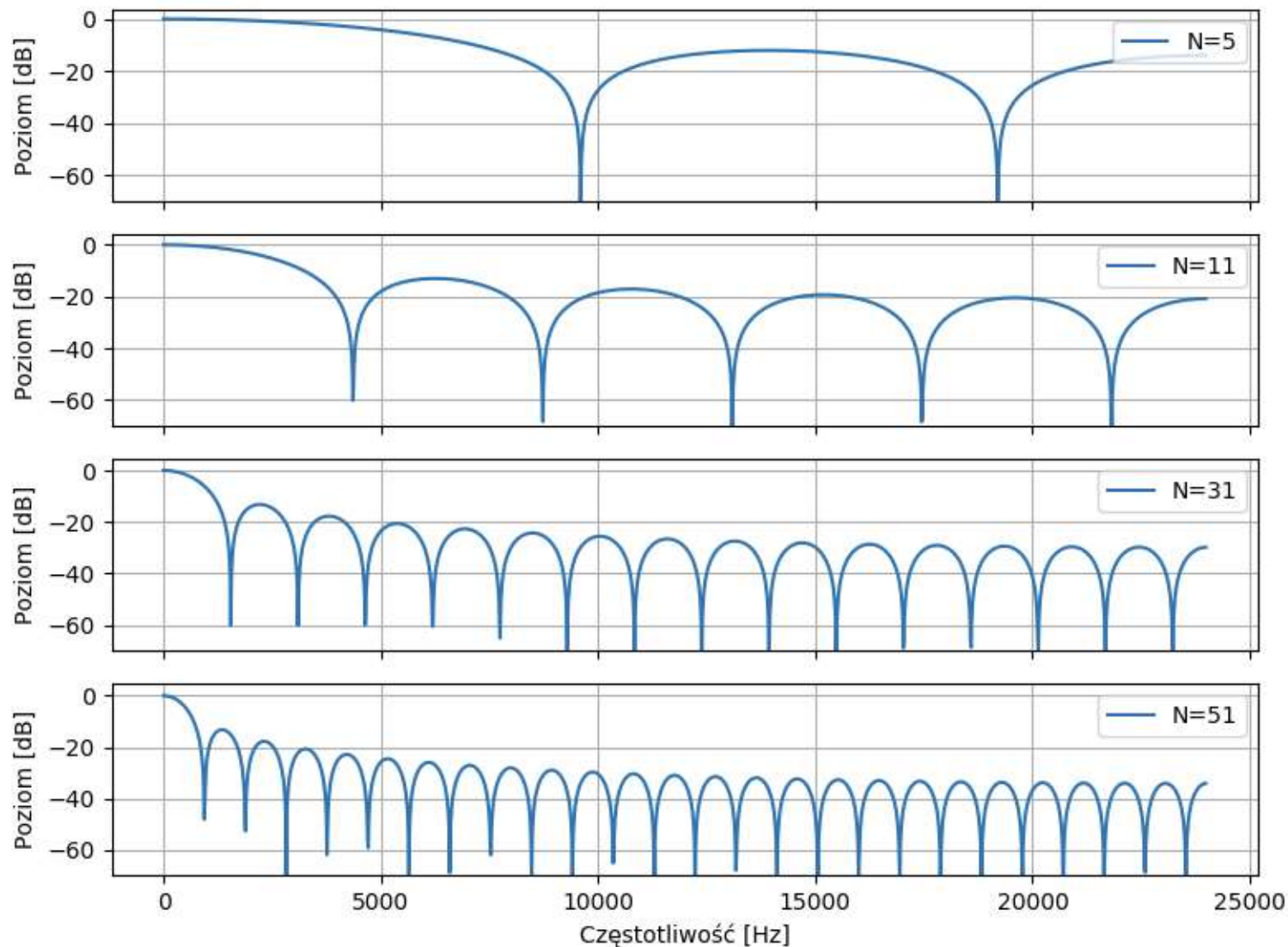
$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i]$$



Wynik wygładzania dla różnej liczby N :



Charakterystyki częstotliwościowe układów wygładzania:



Równanie algorytmu wygładzania:

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i]$$

Zapiszmy je inaczej:

$$y[n] = h \cdot x[n] + h \cdot x[n-1] + h \cdot x[n-2] + \dots + h \cdot x[n-(N-1)]$$

gdzie $h = 1 / N$.

Wartości N ostatnich próbek (w tym bieżącej)
mnożymy przez **współczynnik h** i dodajemy do siebie.

Jest to **filtr średniej ruchomej**
(*moving average filter, MA*).

Równanie filtru średniej ruchomej:

$$y[n] = h \cdot x[n] + h \cdot x[n-1] + h \cdot x[n-2] + \dots + h \cdot x[n-(N-1)]$$

Zapiszmy je w sposób bardziej ogólny:

$$y[n] = h_0 \cdot x[n] + h_1 \cdot x[n-1] + h_2 \cdot x[n-2] + \dots + h_{N-1} \cdot x[n-(N-1)]$$

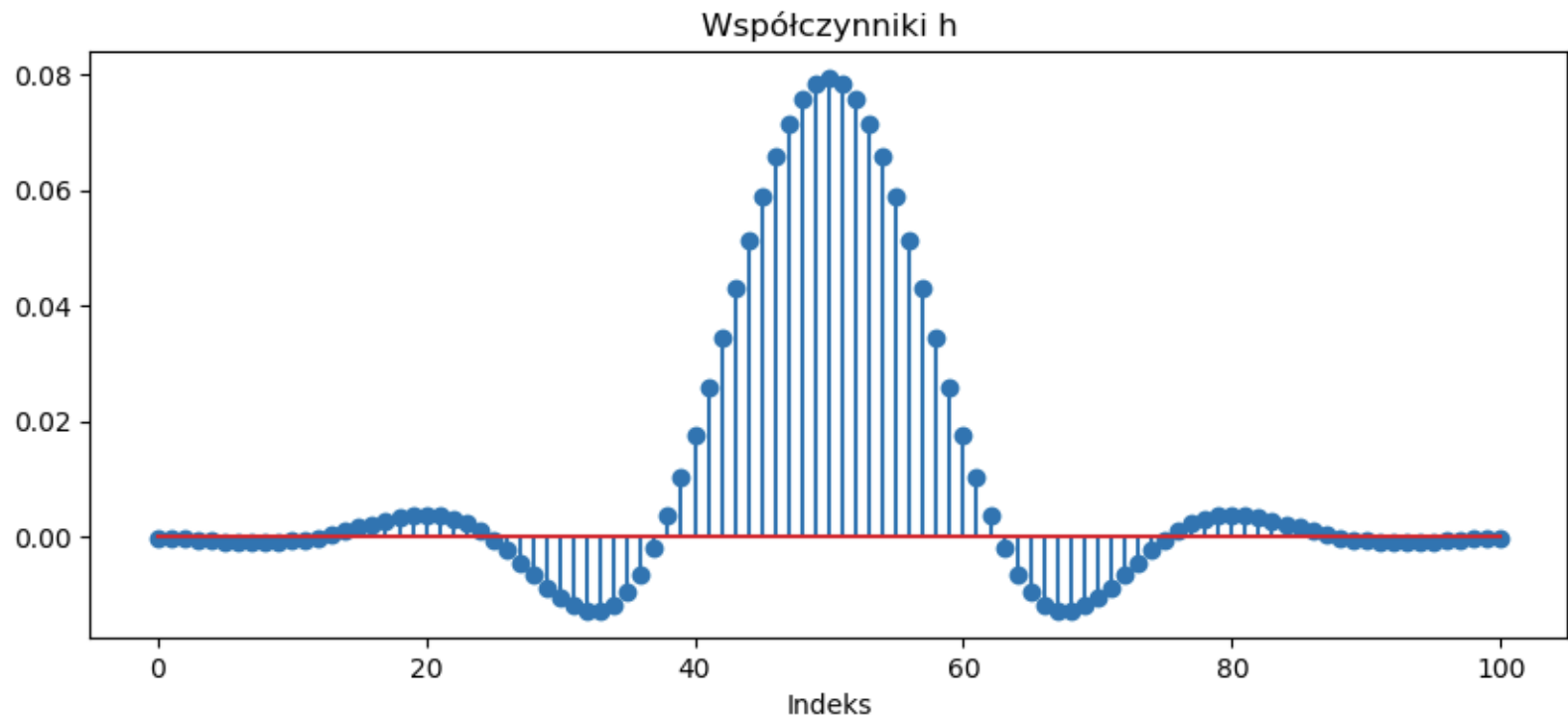
albo krócej:

$$y[n] = \sum_{i=0}^{N-1} (h_i \cdot x[n-i])$$

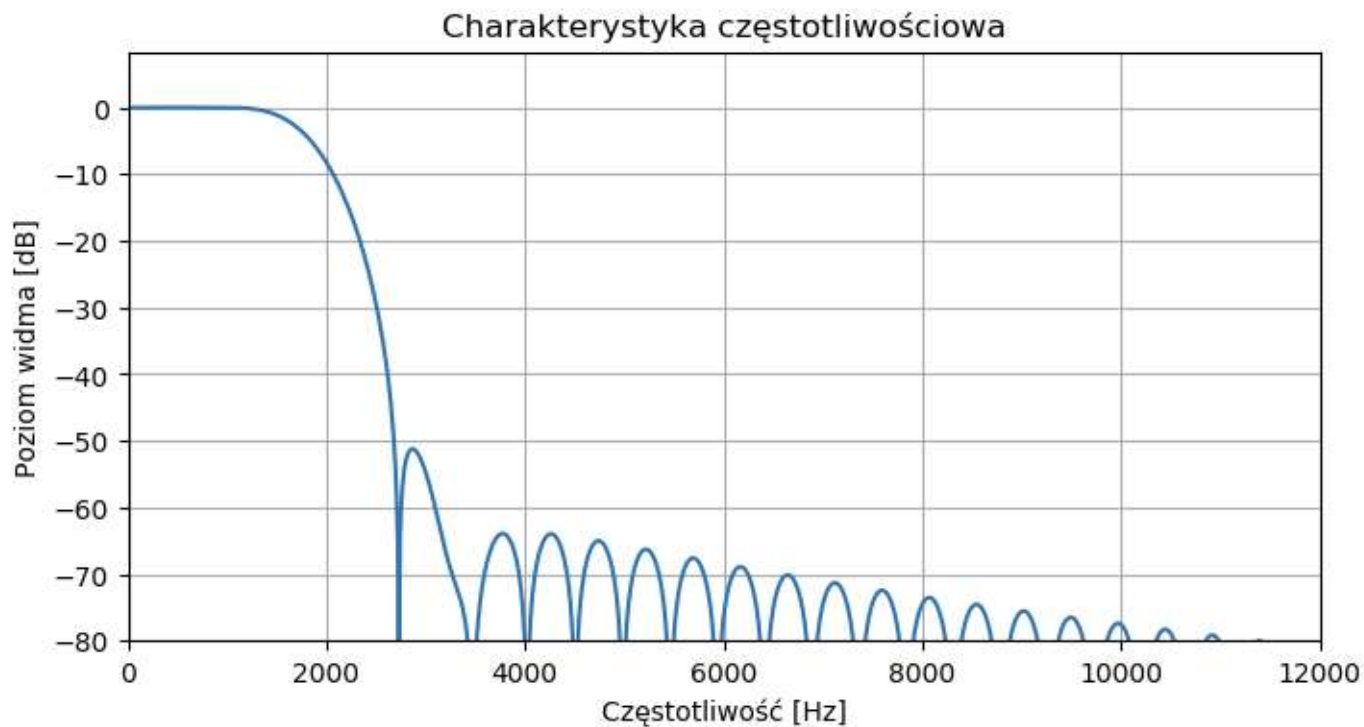
Dla filtru średniej ruchomej:

$$h_0 = h_1 = \dots = h_{N-1} = 1 / N.$$

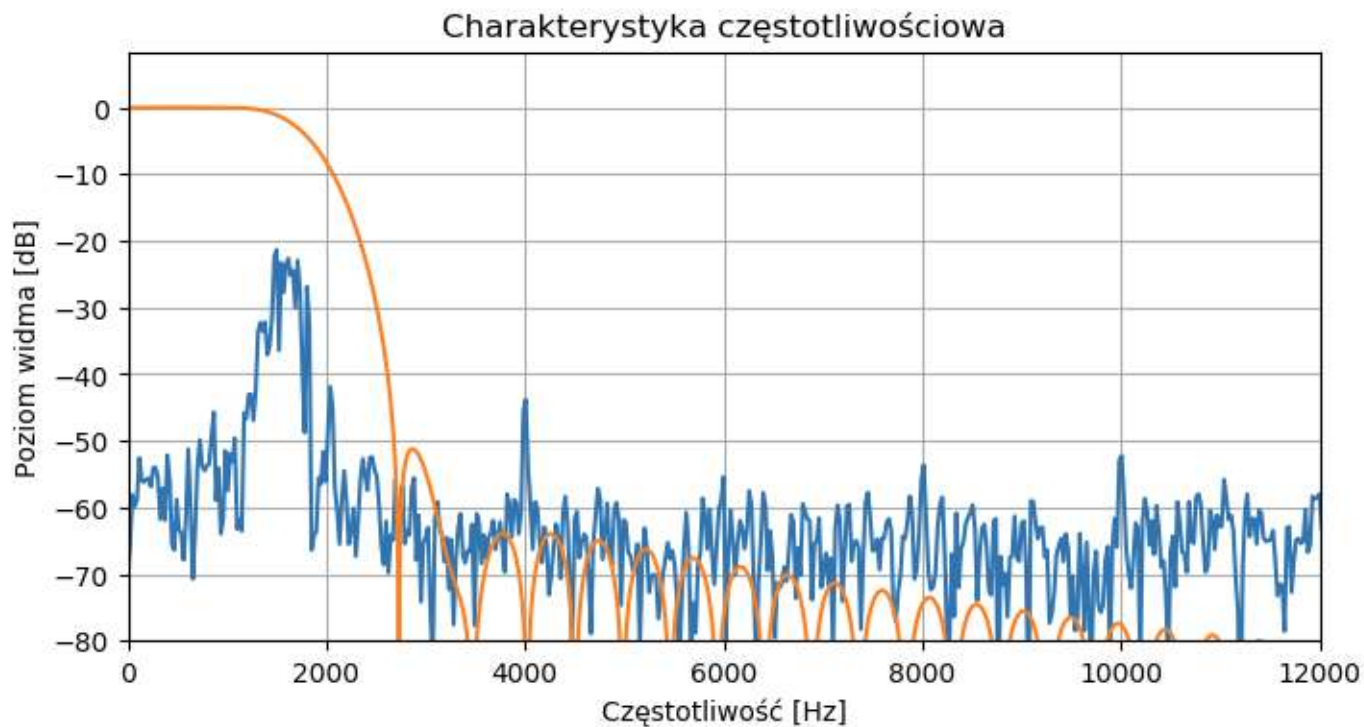
Czyli współczynniki h mogą być dowolne.
Na przykład takie ($N = 101$):



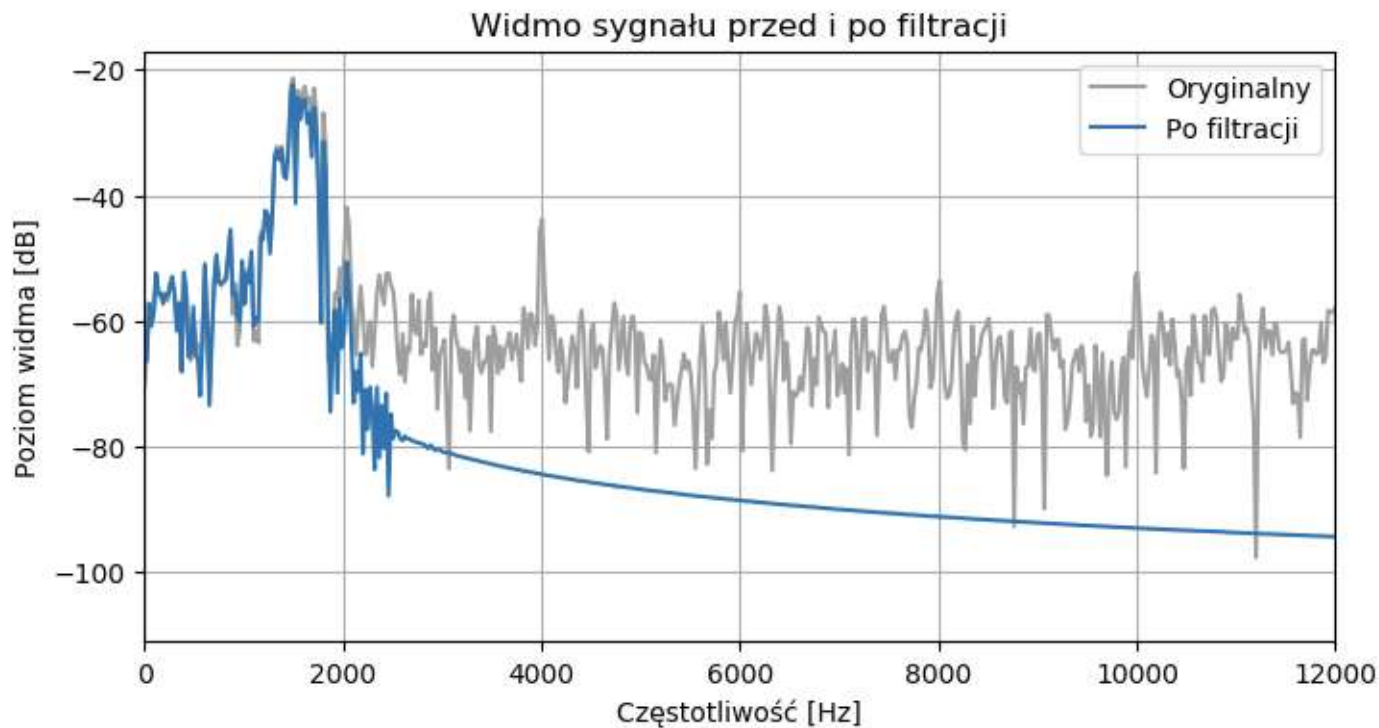
Jak wygląda charakterystyka częstotliwościowa przy takich współczynnikach?



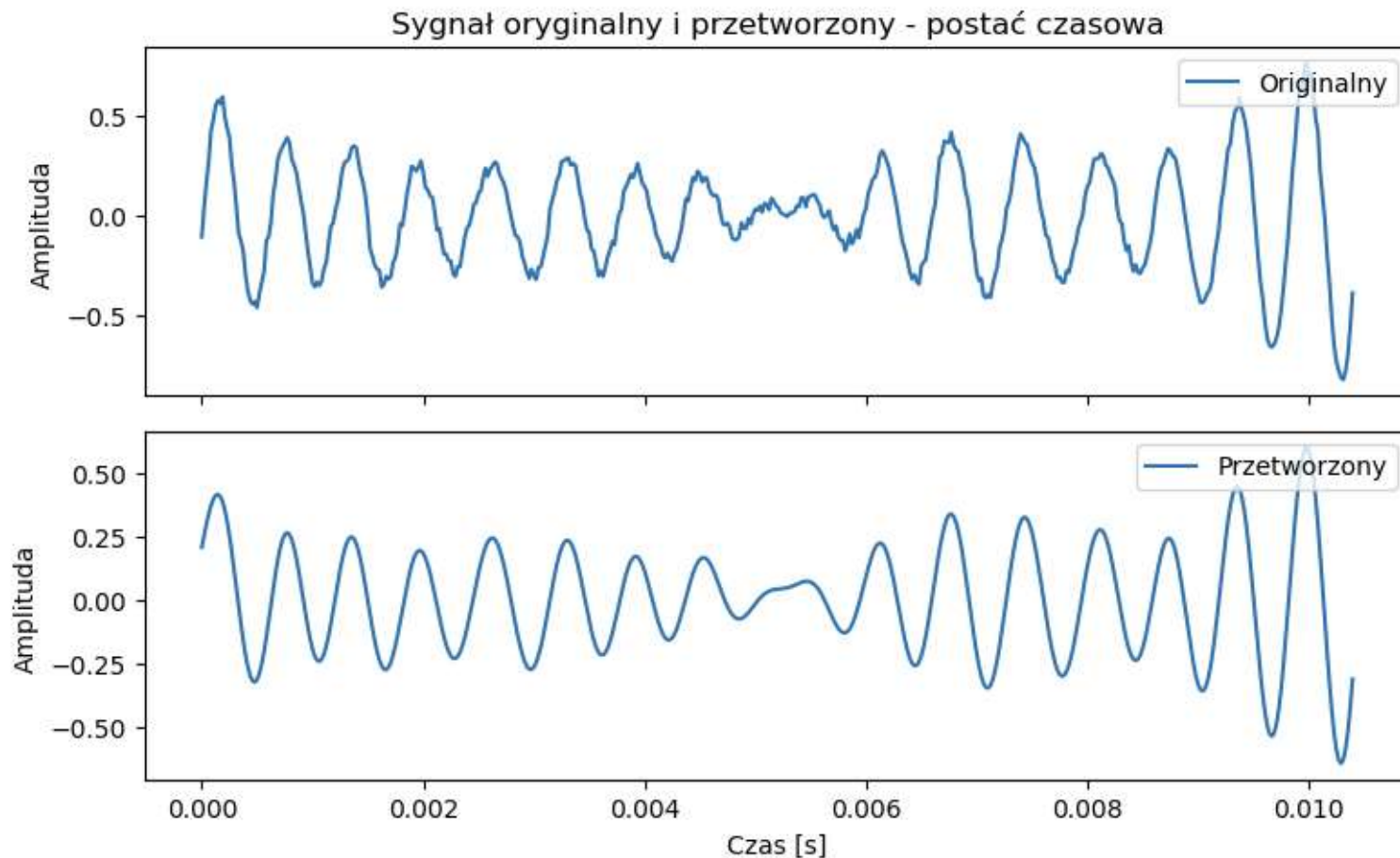
Wracamy do przykładu nr 1.
Czy to nam pomoże usunąć zniekształcenia?



Widmo sygnału oryginalnego oraz po przetworzeniu przez algorytm:



Postać czasowa – przed i po:



Sukces! Zniekształcenia zostały usunięte!

Przepis na obliczenie wyjściowych wartości próbek
- tzw. równanie różnicowe:

$$y[n] = h_0x[n] + h_1x[n-1] + h_2x[n-2] + \dots + h_{N-1}x[n-(N-1)]$$

$$y[n] = \sum_{i=0}^{N-1} (h_i x[n-i])$$

Algorytm realizujący obliczenia według tego równania nazywa się
filtrem cyfrowym o skończonej odpowiedzi impulsowej
(*FIR* – *finite impulse response filter*).

Co robi filtr FIR:

- bierze N ostatnich próbek,
- mnoży je przez współczynniki h ,
- sumuje wyniki mnożenia,
- wysyła wynik na wyjście.

I to naprawdę wszystko!

```
y = 0
```

```
FOR i = 0 TO N-1:
```

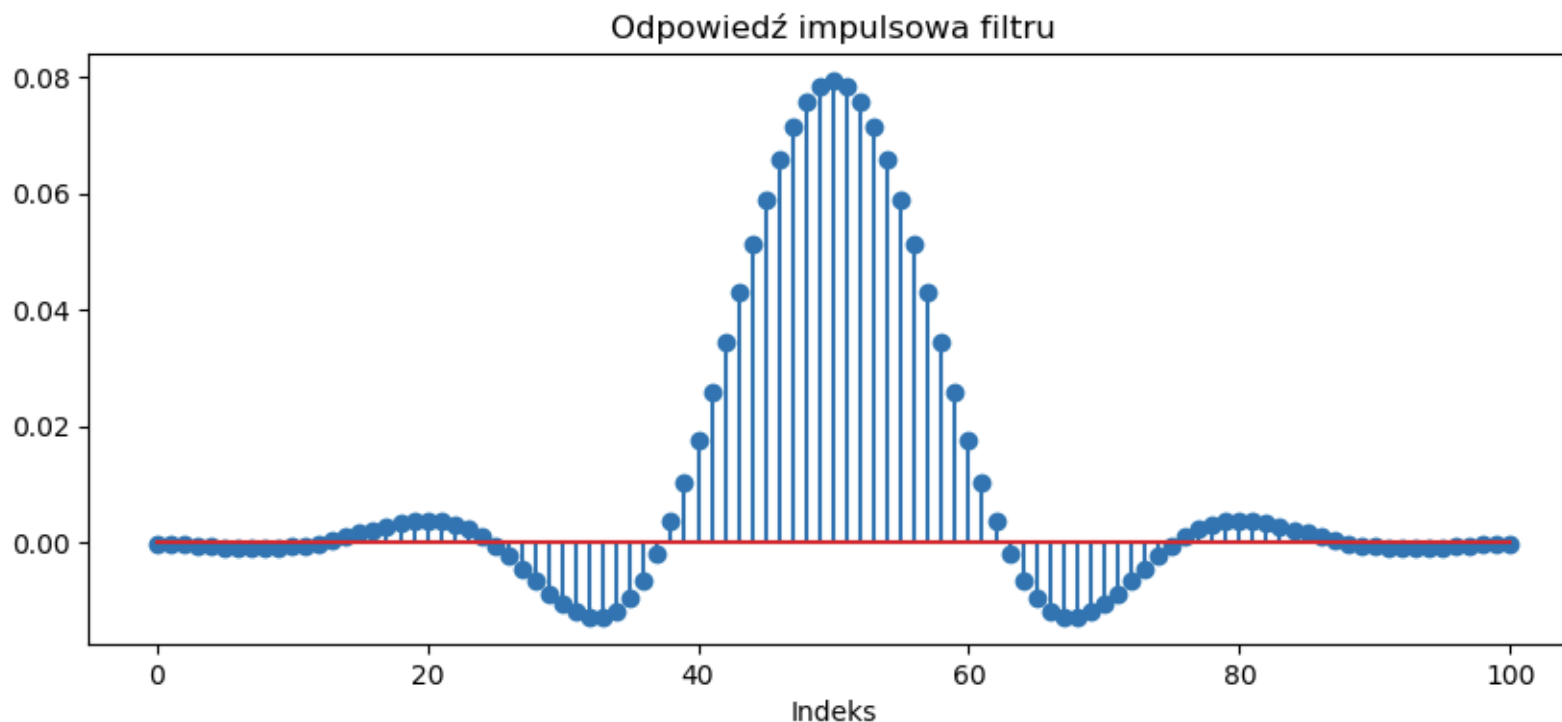
```
    y = y + x[i] * h[i]
```

```
RETURN y
```

Odpowiedź impulsowa

filtru FIR na pobudzenie impulsowe $\delta[n]$
jest równa zbiorowi współczynników filtru:

$$(h_0, h_1, h_2, \dots, h_{N-1})$$



W dziedzinie zmiennej zespolonej z , opóźnieniu sygnału o jedną próbkę odpowiada z^{-1} .

$$y[n] = h_0 x[n] + h_1 x[n-1] + h_2 x[n-2] + \dots + h_{N-1} x[n-(N-1)]$$

możemy zapisać jako:

$$H[z] = h_0 + h_1 z^{-1} + h_2 z^{-2} + \dots + h_{N-1} z^{-(N-1)}$$

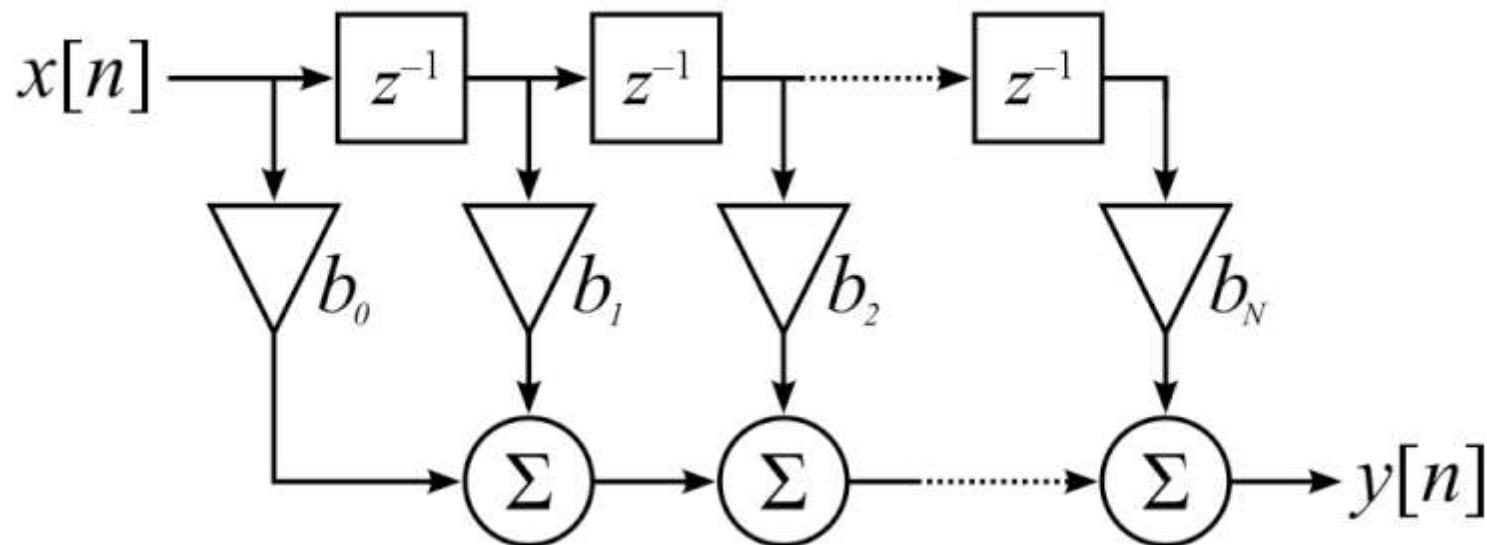
Jest to **transmitancja** filtru FIR.

Transmitancja jest transformatą Fouriera odpowiedzi impulsowej (czyli zbioru współczynników):

$$H[z] = \mathcal{F}(h[n])$$

Schemat filtru FIR

(współczynniki oznacza się czasami literą b):



Pobieramy sygnał po każdym z^{-1} i mnożymy przez b_i – miejsca te nazywa się „odczepami” (*tap*).

- Długość filtru FIR (*filter length*)
 - liczba współczynników filtru (N),
 - czyli długość odpowiedzi impulsowej.
- Rząd filtru FIR (*filter order*)
 - najwyższa potęga w transmitancji,
 - zawsze o 1 mniejsza niż długość filtru ($N-1$)
(ponieważ współczynniki numerujemy od zera).

Np. filtr FIR o 51 współczynnikach
ma długość 51 i rząd 50.

Transmitancję filtru można też zapisać w następujący sposób:

$$H(z) = k(1 - q_1 z^{-1})(1 - q_2 z^{-1})(1 - q_3 z^{-1}) \dots (1 - q_{N-1} z^{-1})$$

k – stałe wzmocnienie

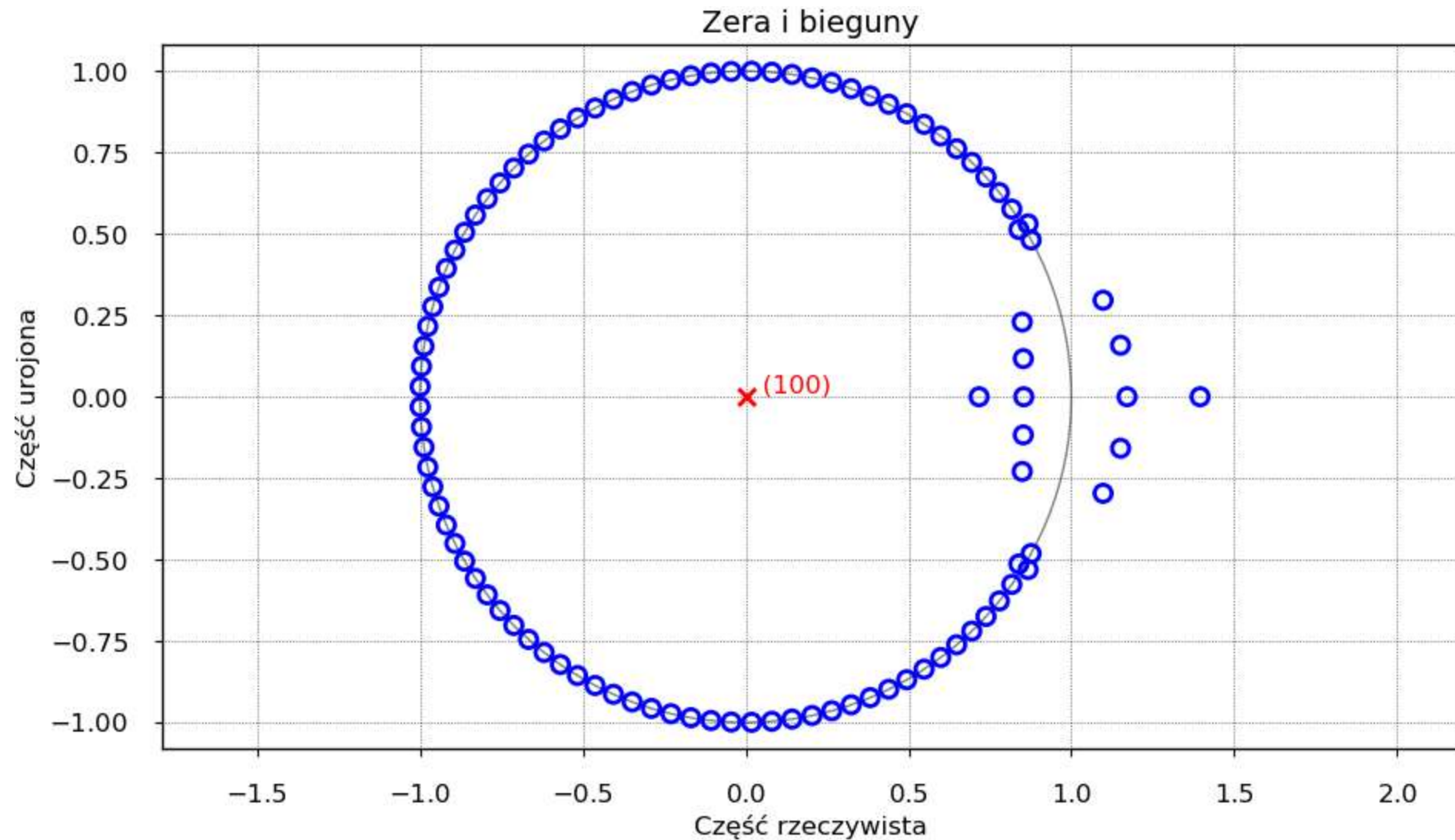
q_i – zera transmitancji

Transmitancja filtru FIR o długości N posiada:

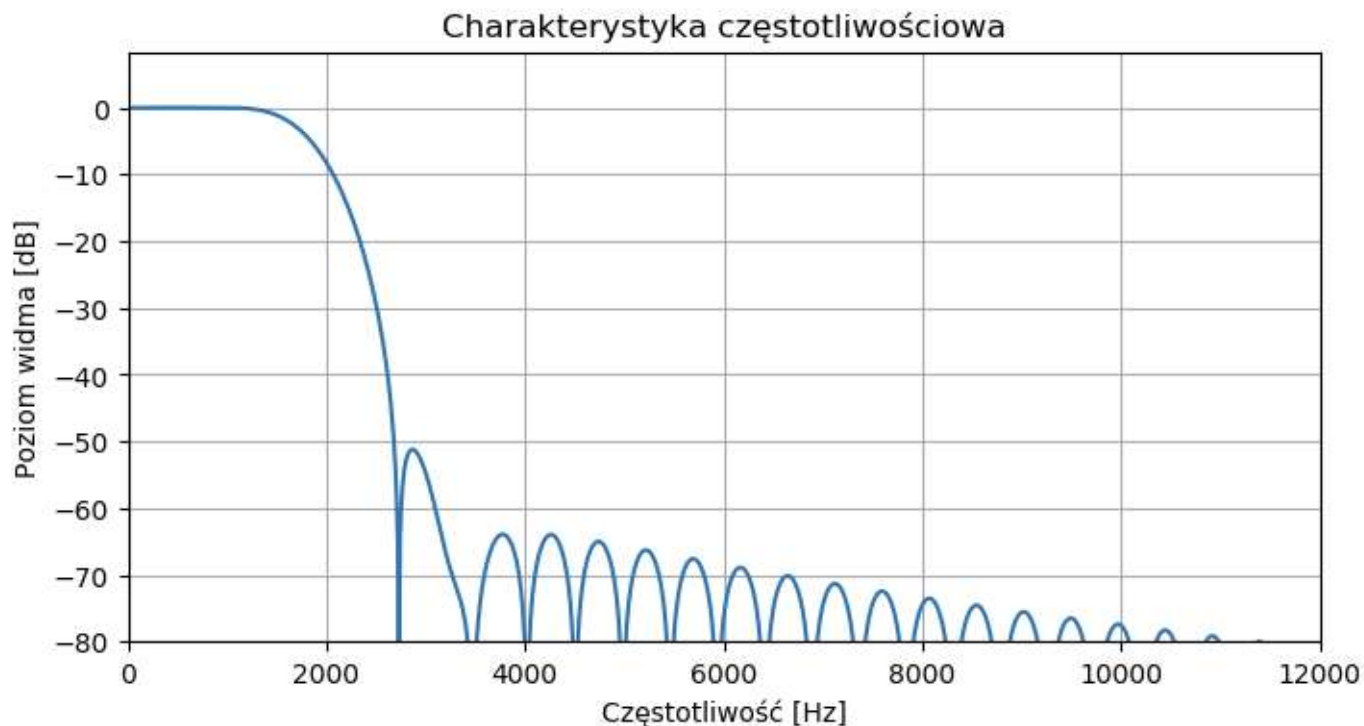
- $N-1$ zer (w parach zespolonych sprzężonych),
- $N-1$ biegunów położonych w punkcie zerowym.

Z tego względu, **filtry FIR są zawsze stabilne.**

Zera i bieguny transmitancji filtru FIR ($N = 101$):



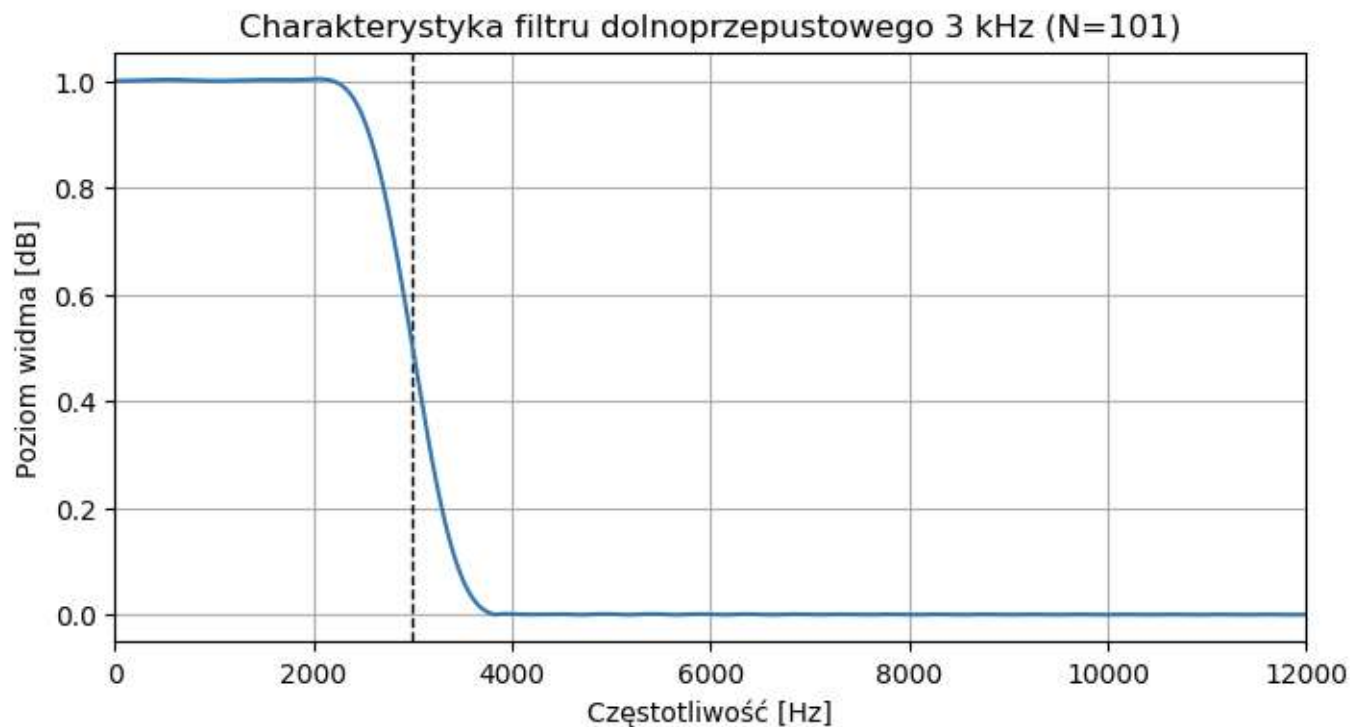
Moduł transmitancji $|H[z]|$ określa **charakterystykę amplitudową** (częstotliwościową)
– wzmacnienie filtru dla poszczególnych zakresów częstotliwości.



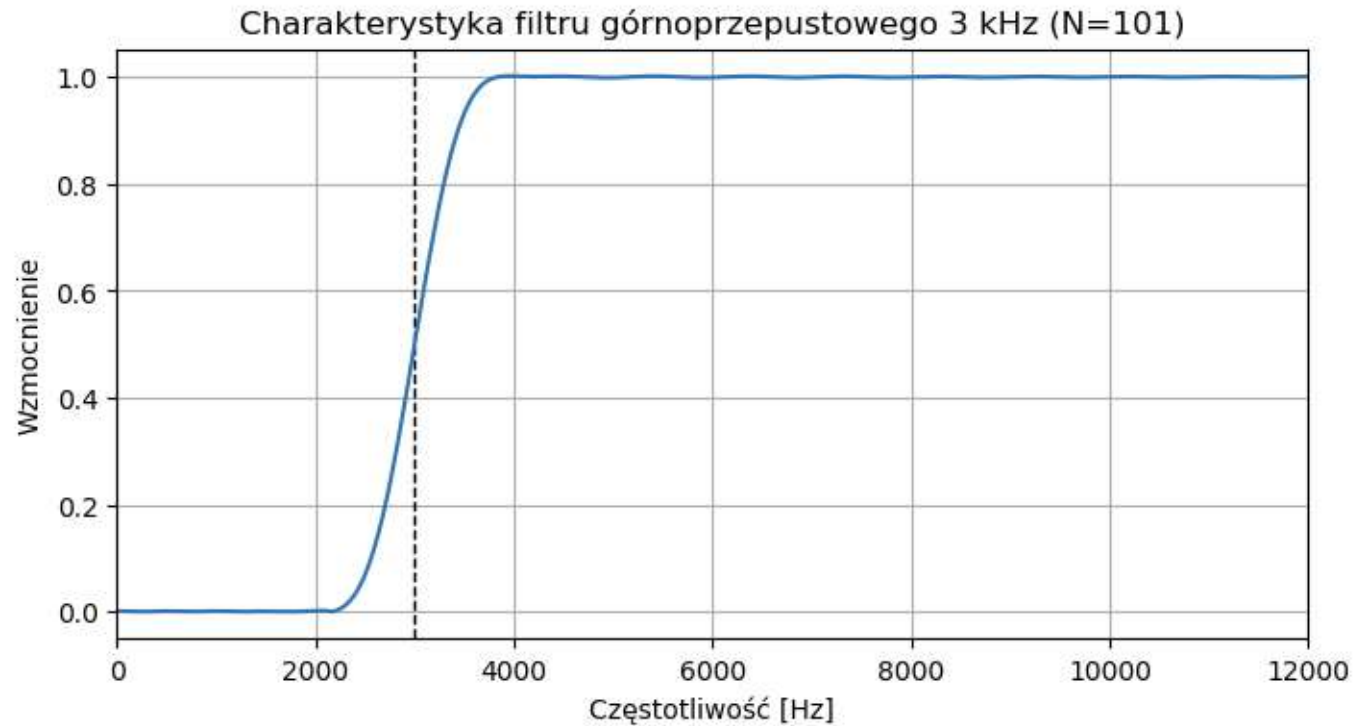
Najczęściej chcemy przepuścić wybrany zakres częstotliwości bez zmiany i stłumić resztę.

- **Pasmo przepustowe** (*pass band*)
 - wzmacnienie filtru powinno być równe 1, składowe przepuszczane bez zmiany.
- **Pasmo zaporowe** (*stop band*)
 - wzmacnienie filtru powinno być bliskie 0, składowe są tłumione.
- **Częstotliwość graniczna** lub odcięcia (*cut-off*)
 - granica między pasmem przepustowym i zaporowym.

Filtr dolnoprzepustowy (DP) – *low-pass* (LP)



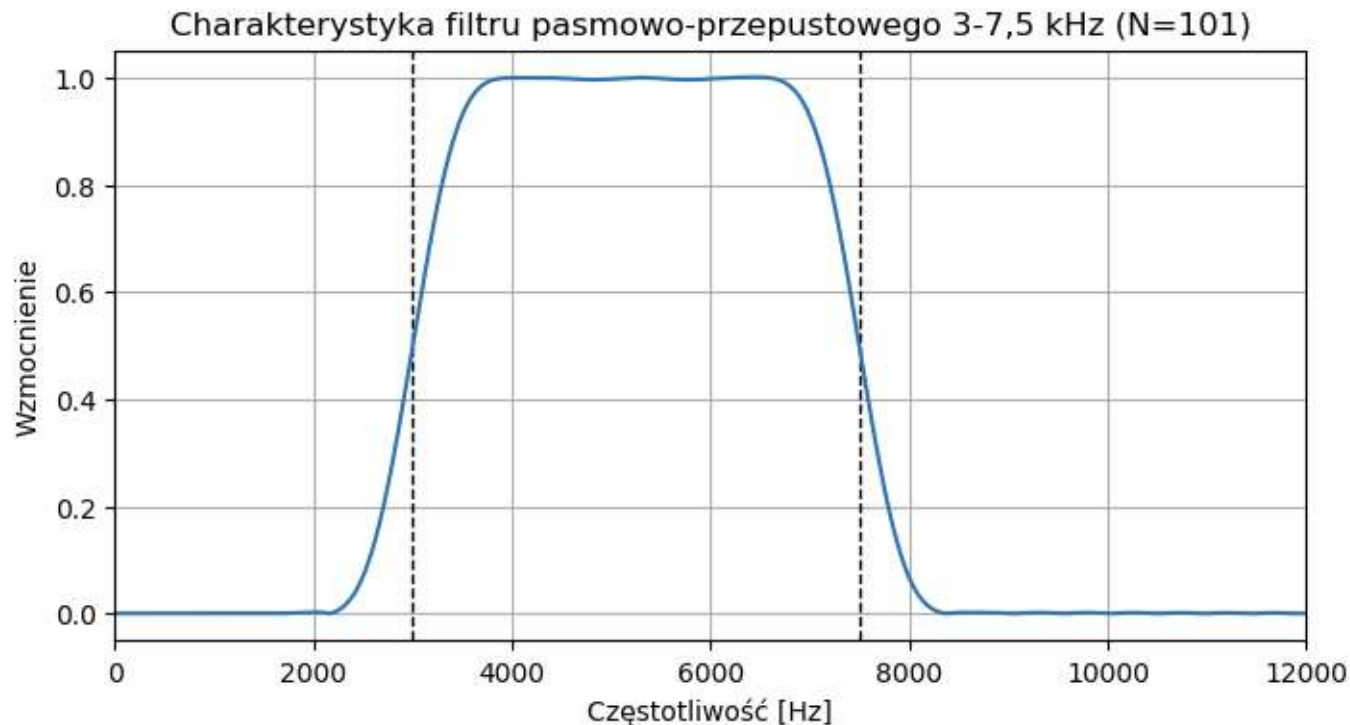
Filtr górnoprzepustowy (GP) – *high-pass* (HP)



W dziedzinie częstotliwości: $GP(f) = 1 - DP(f)$

Filtr pasmowo-przepustowy (PP) – *band-pass* (BP)

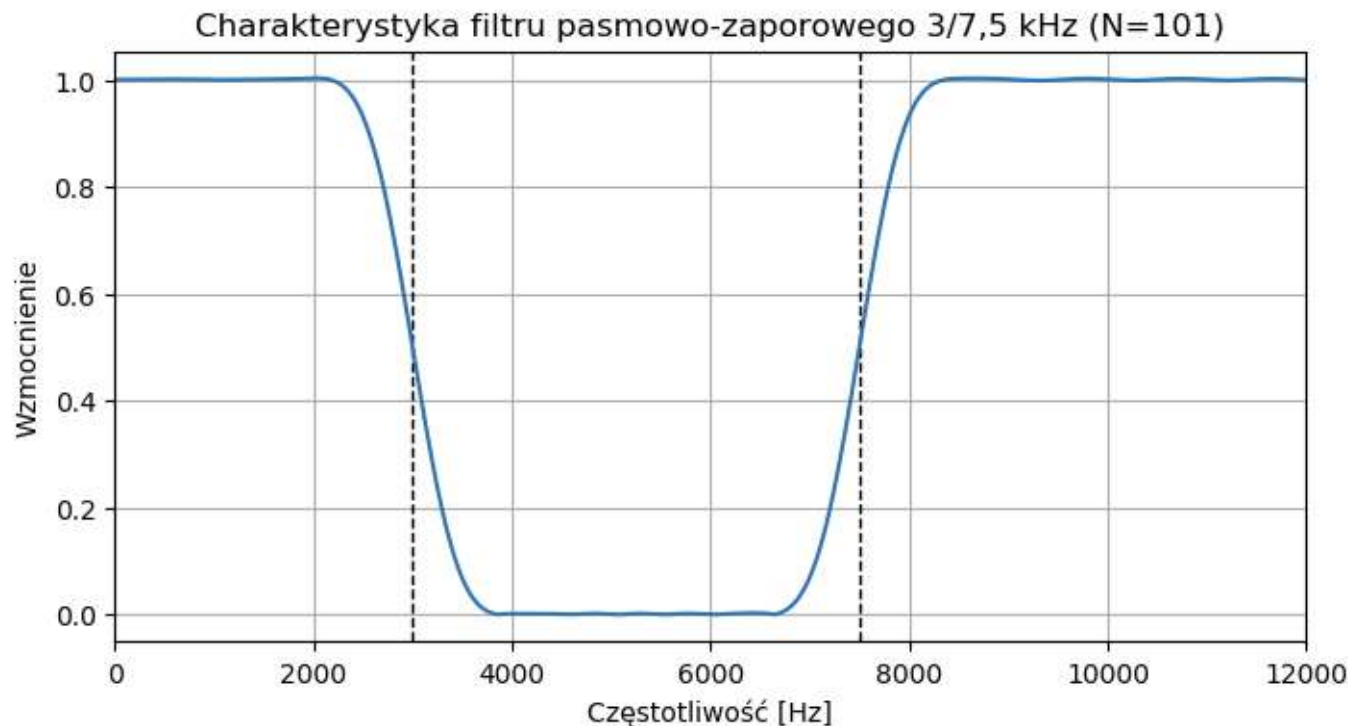
Dwie częstotliwości graniczne – górna i dolna



W dziedzinie częstotliwości: $PP(f) = DP(f) \cdot GP(f)$

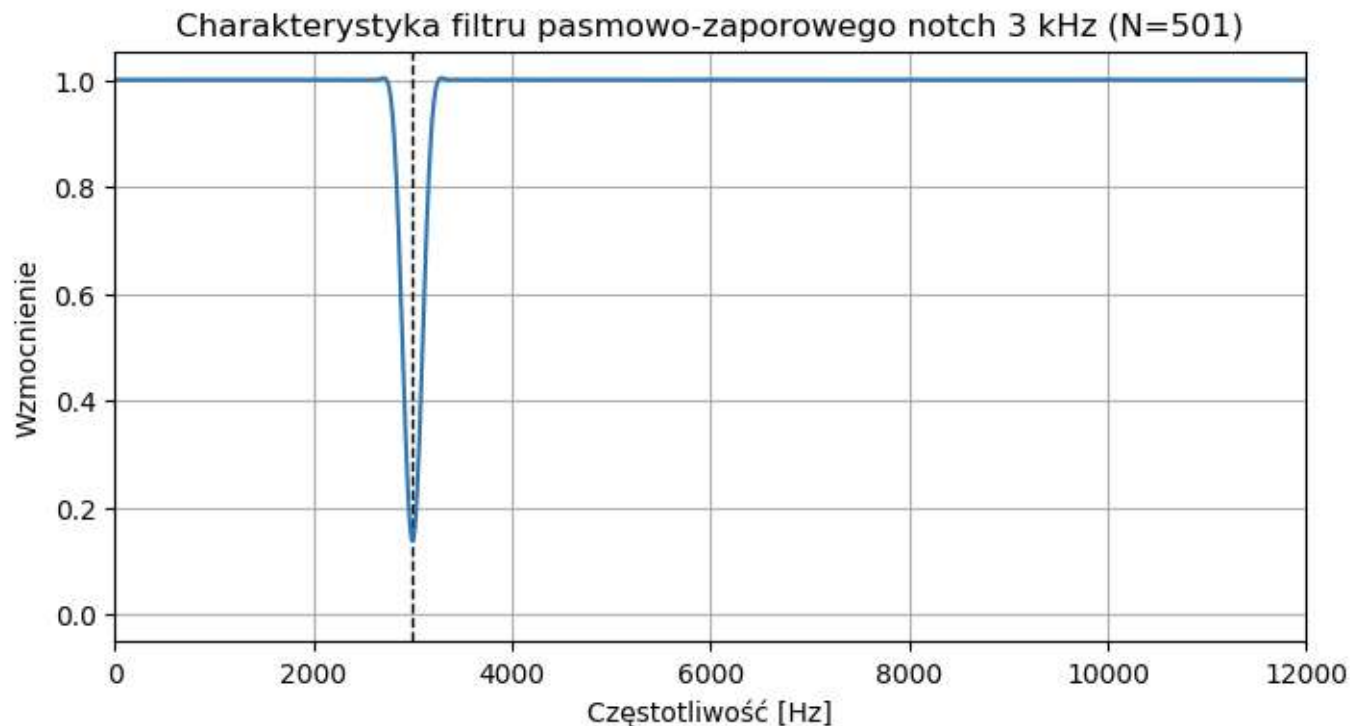
Filtr pasmowo-zaporowy (PZ) – *band-stop* (BS)

Dwie częstotliwości graniczne – górna i dolna



W dziedzinie częstotliwości: $PZ(f) = DP(f) + GP(f)$

Wąskopasmowy filtr pasmowo-zaporowy – – *notch filter*

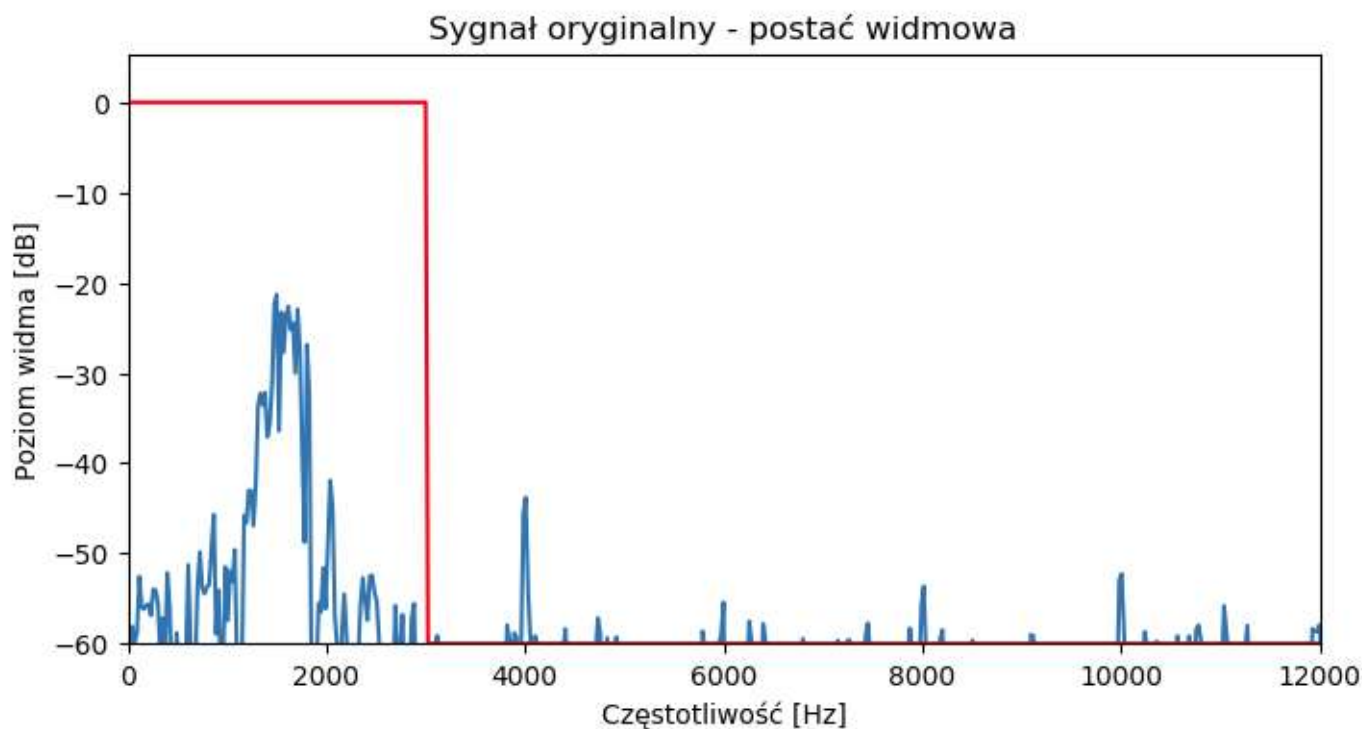


Służy do eliminacji jednej konkretnej częstotliwości.

Projektowanie filtru cyfrowego FIR

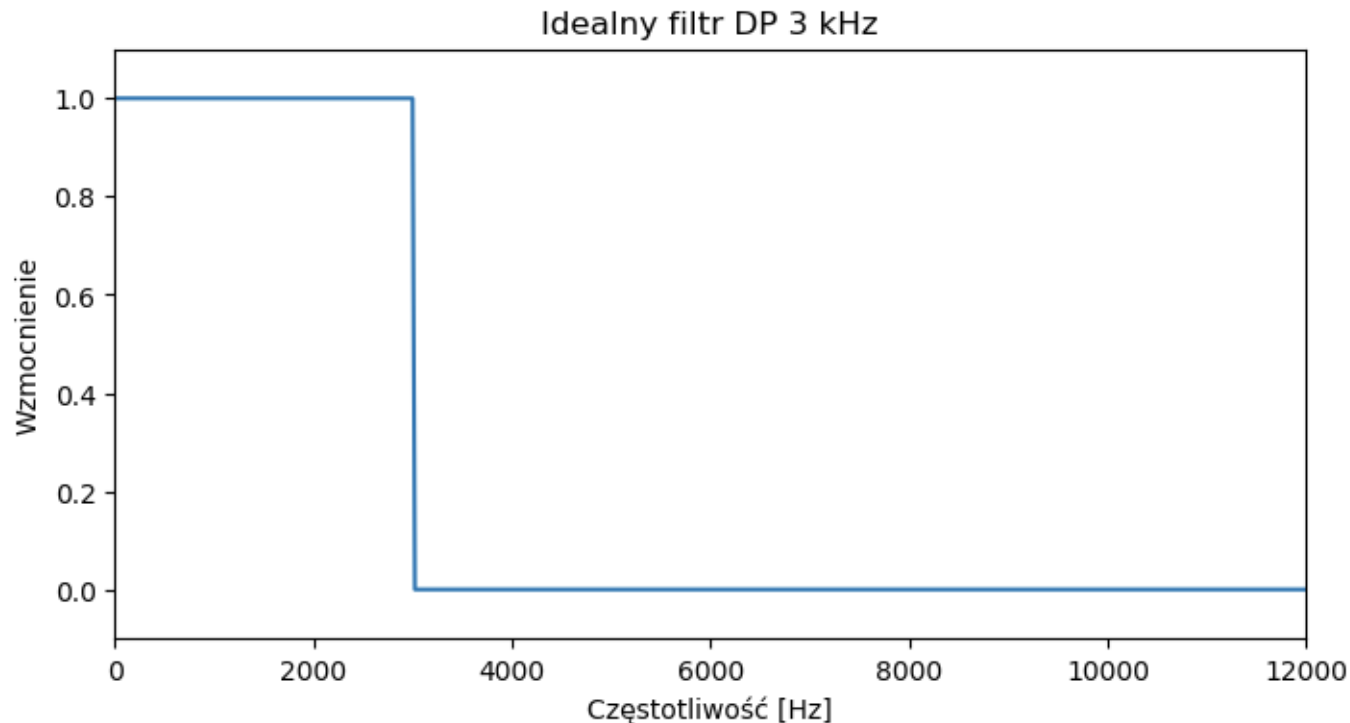
- Decyzja projektanta:
 - jaki typ charakterystyki?
 - jaka długość filtru (ile współczynników)?
 - która metoda projektowania?
- Obliczenie współczynników filtru na podstawie podanych danych – wykonuje program komputerowy.
- Sprawdzenie obliczonych charakterystyk filtru.
- Ocena efektów filtracji, ew. powtórzenie projektu.

Wracamy do przykładu nr 1.



Aby usunąć zniekształcenia, potrzebny jest filtr dolnoprzepustowy o częstotliwości granicznej ok. 3 kHz.

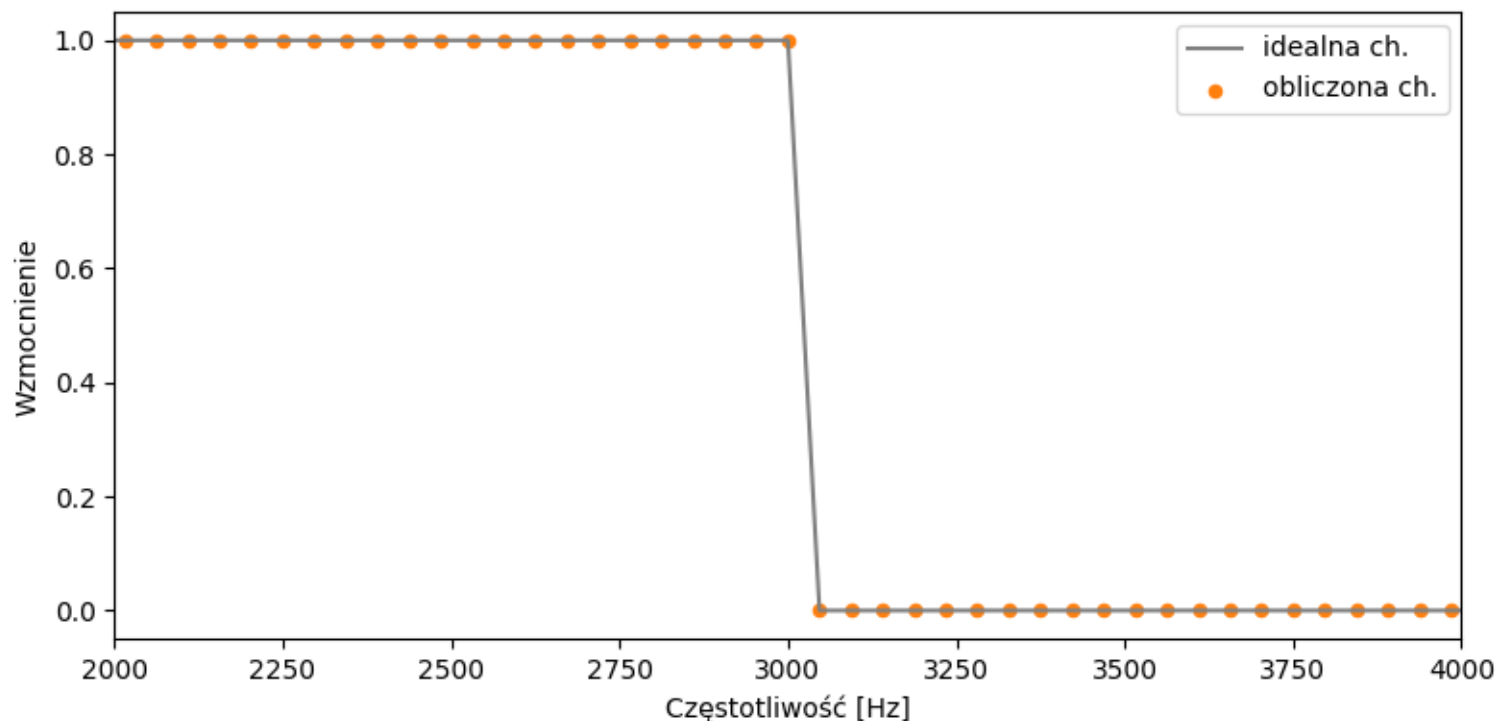
Charakterystyka idealnego filtra DP 3 kHz:



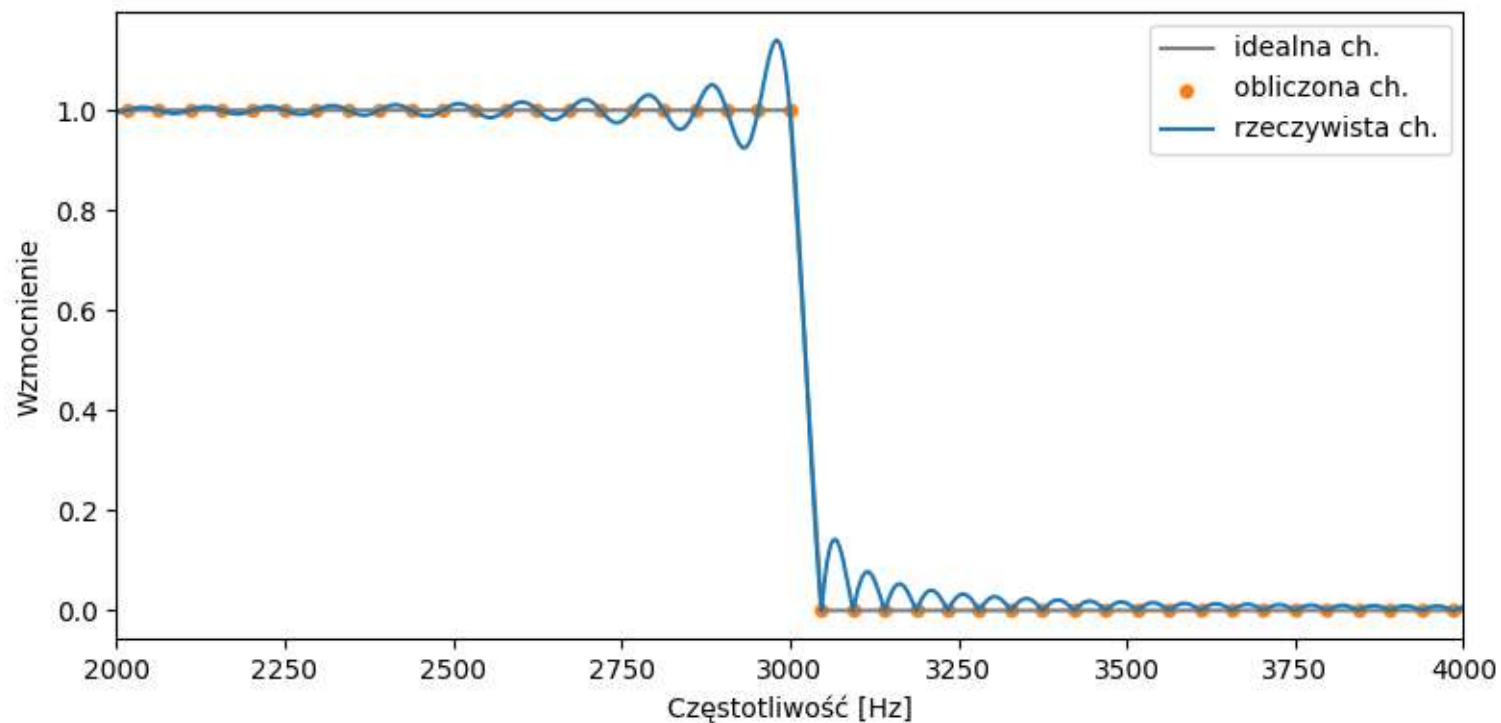
Teoretycznie, powinniśmy móc obliczyć odwrotną transformatę Fouriera, otrzymując współczynniki filtra. Problem rozwiązany?

*Dlaczego nie mogę mieć idealnego filtra FIR?
Przecież mam dokładnie taką charakterystykę,
jaką chciałem mieć?*

Tak, ale ta charakterystyka jest określona tylko
w wybranych punktach skali częstotliwości:



Charakterystyka pomiędzy punktami określonymi podczas obliczania filtru wcale nie jest taka, jakiej oczekujemy:

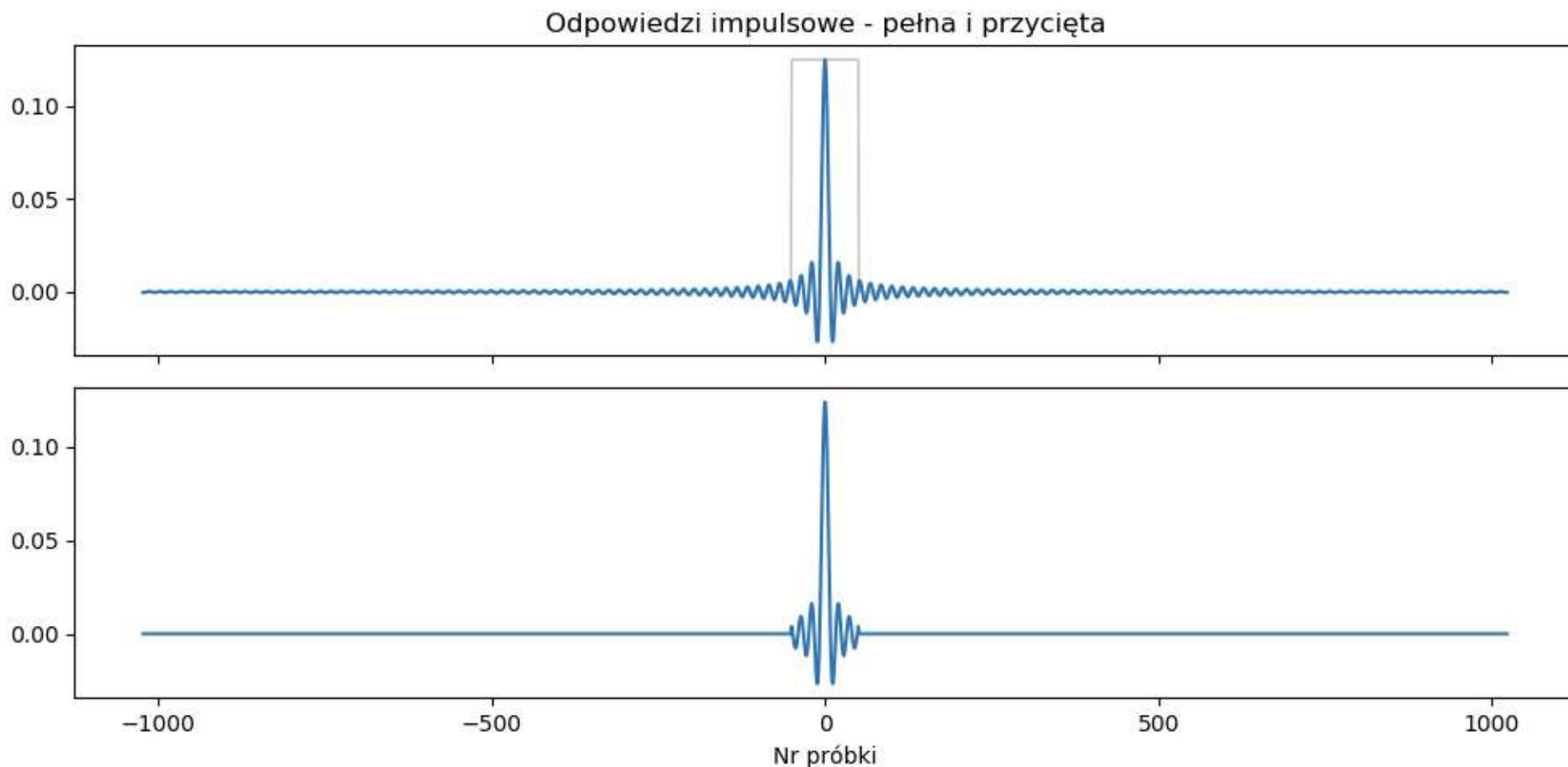


Dlaczego nie da się zrobić filtru FIR w ten sposób, licząc IFFT idealnej charakterystyki widmowej?

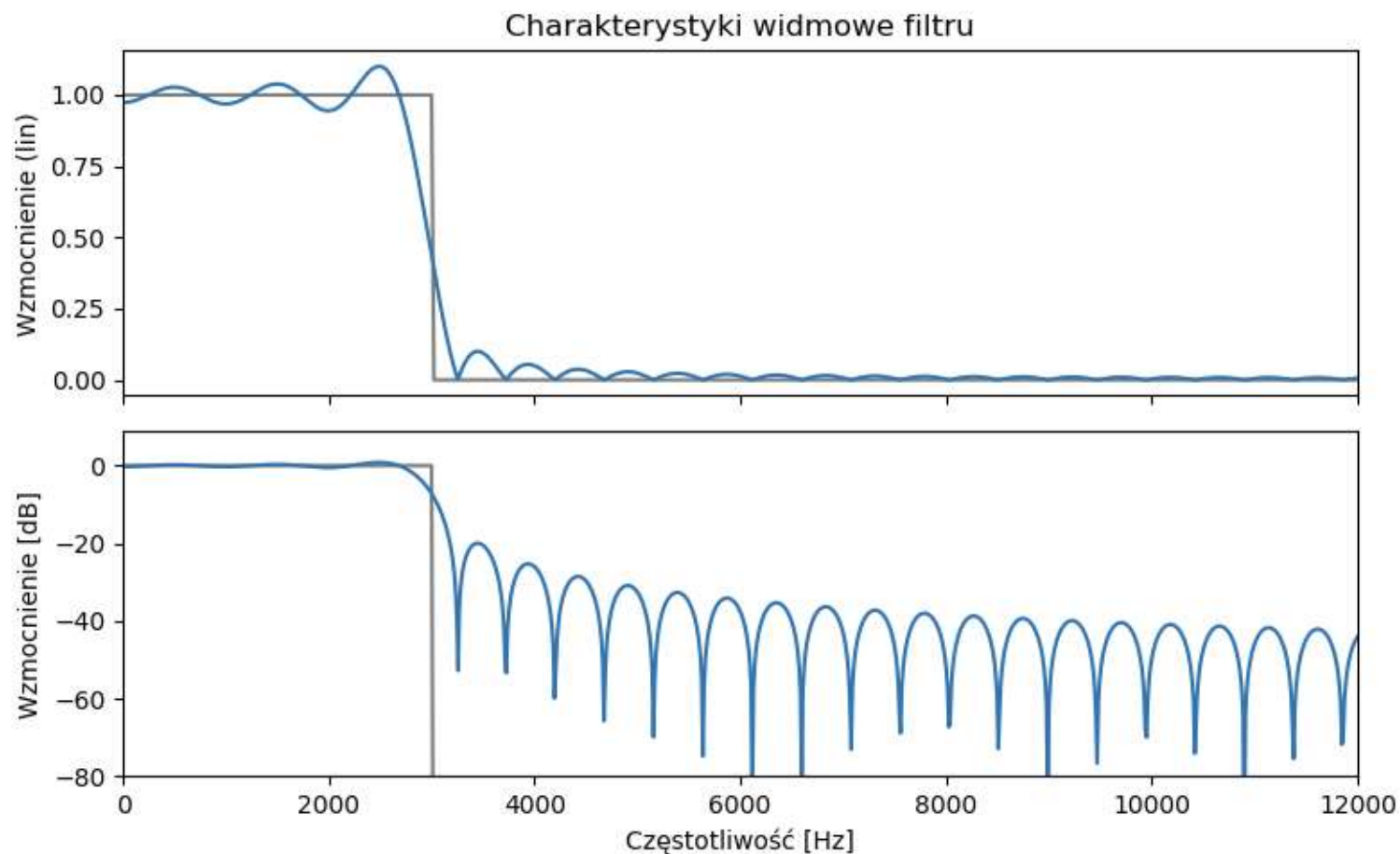
Ponieważ idealny filtr FIR:

1. ma nieskończoną odpowiedź impulsową, czyli $N = \infty$,
2. wymaga znajomości przyszłych próbek sygnału, czyli jest nieprzyczynowy.

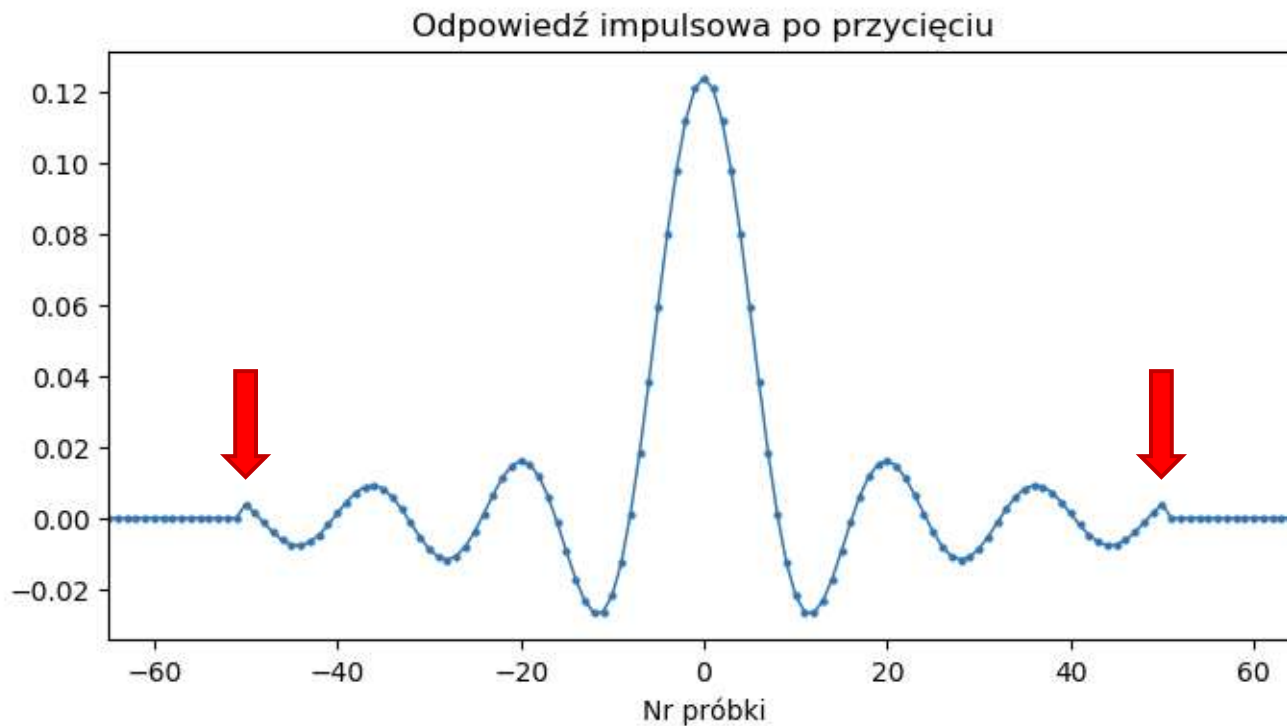
Pierwszy problem ($N = \infty$) możemy rozwiązać przycinając odpowiedź impulsową do ustalonej długości N .



Zniekształcenia na skutek przycięcia odp. impulsowej:



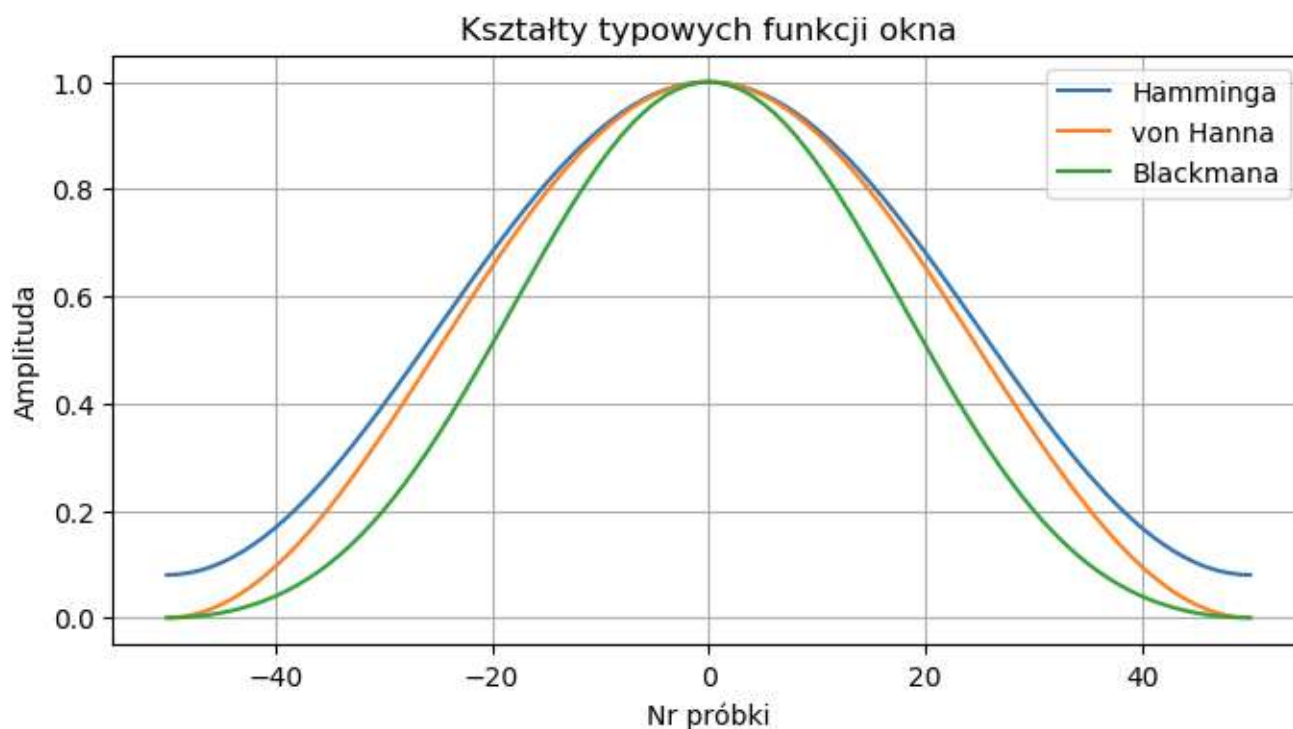
Powodem zniekształceń jest nieciągłość odpowiedzi impulsowej na granicach przyciętego fragmentu:



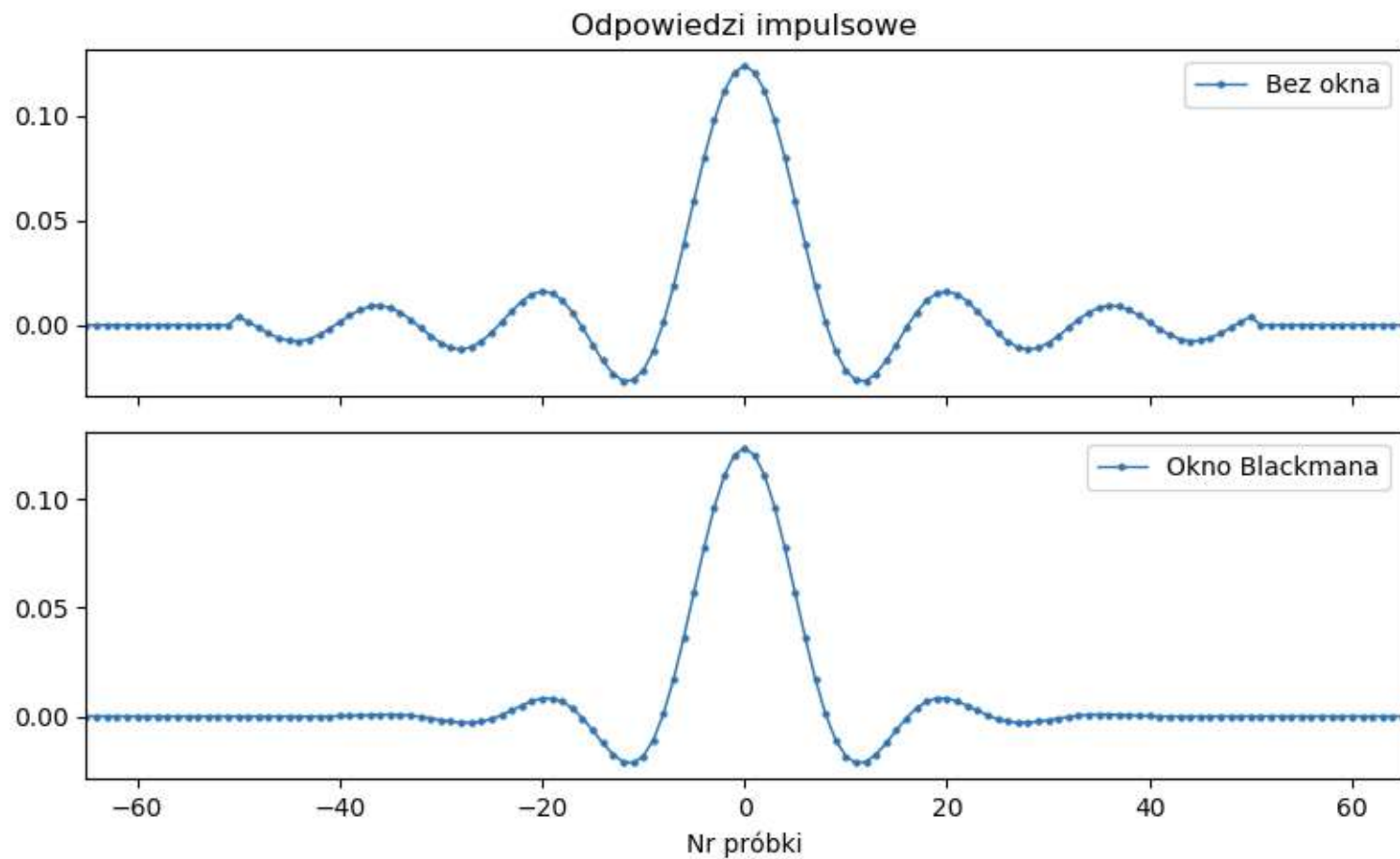
Rozwiązaniem jest przemnożenie przycinanej odpowiedzi impulsowej przez funkcję okna (*window*).

Typowe okna: Hamminga, von Hanna, Blackmana.

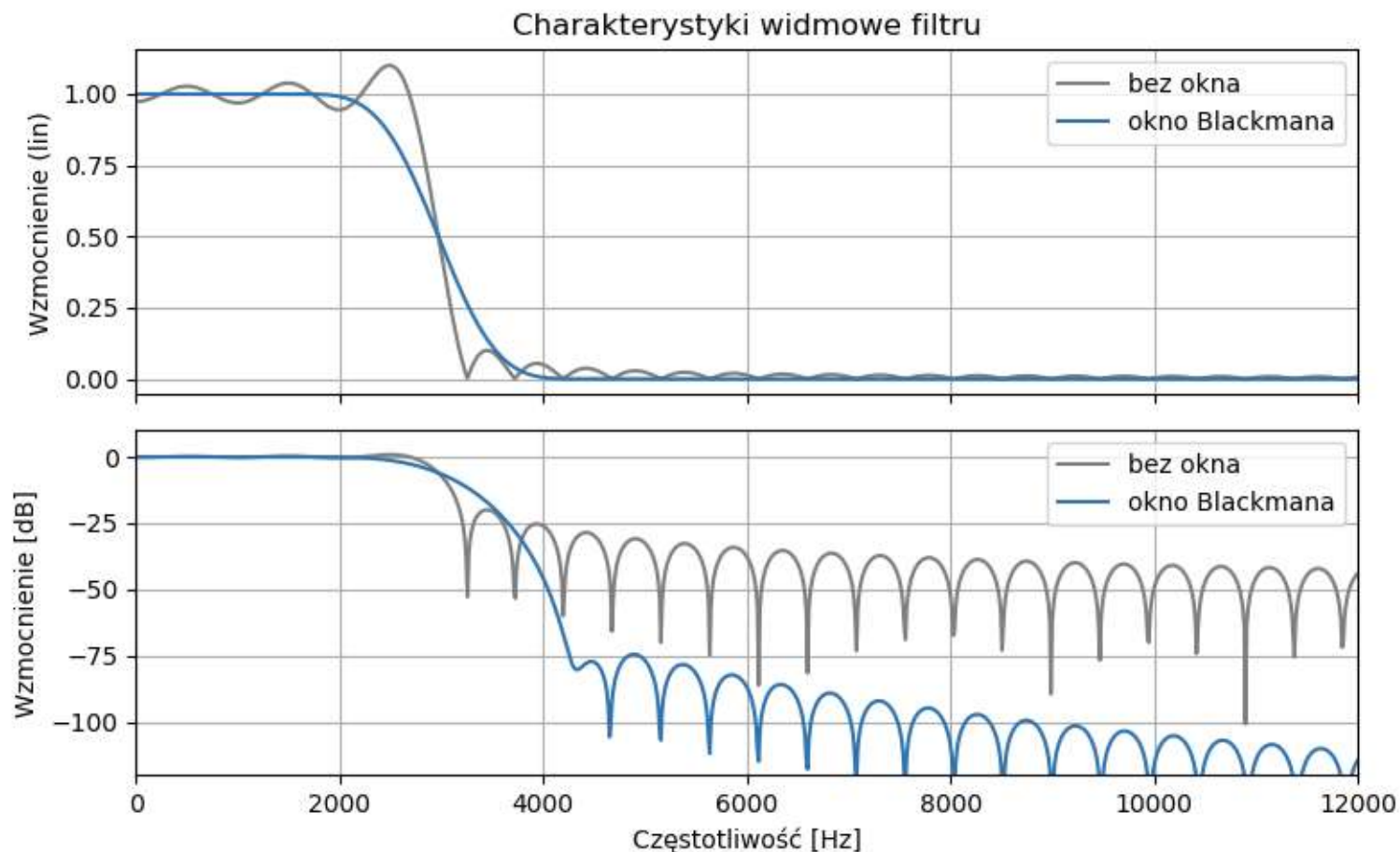
Wybór okna ma wpływ na kształt charakterystyki filtru.



Odpowiedź impulsowa przycięta bez okna
oraz oknem Blackmana:

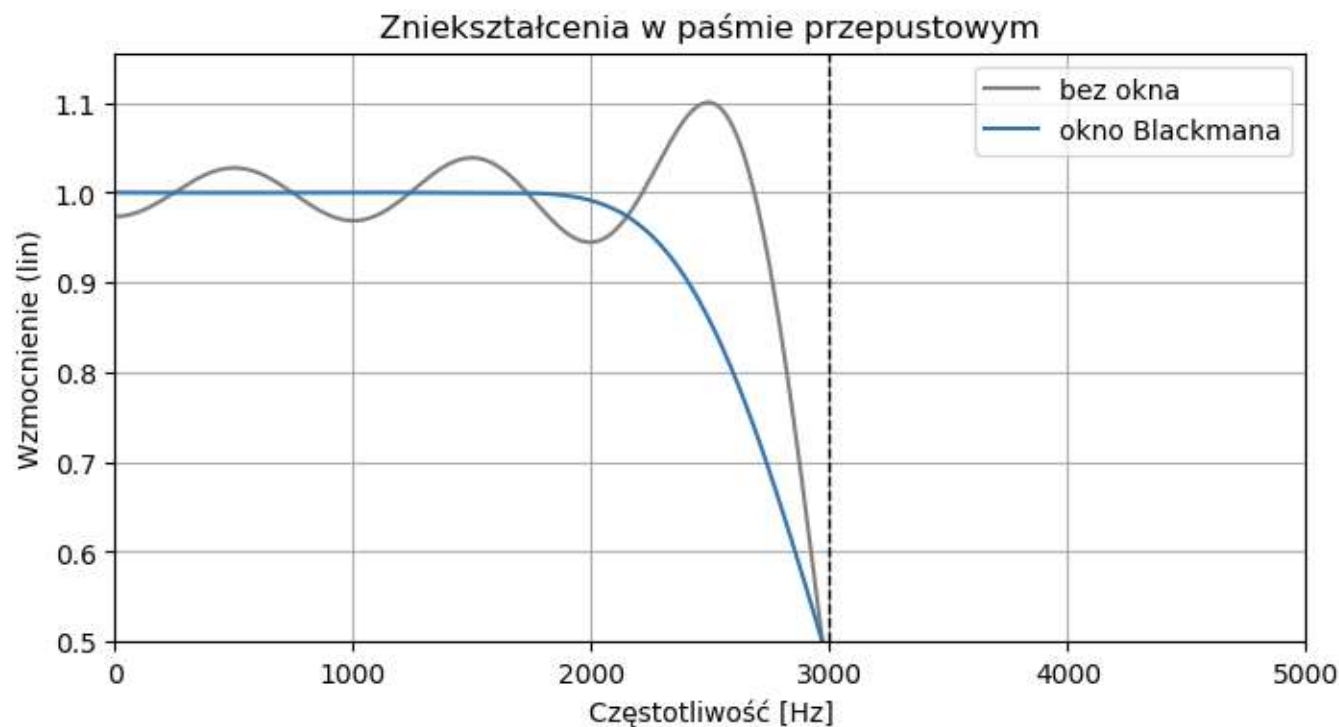


Porównanie charakterystyk widmowych bez okna i z oknem Blackmana ($N = 101$):



Zniekształcenia w wyniku przycięcia odp. imp.

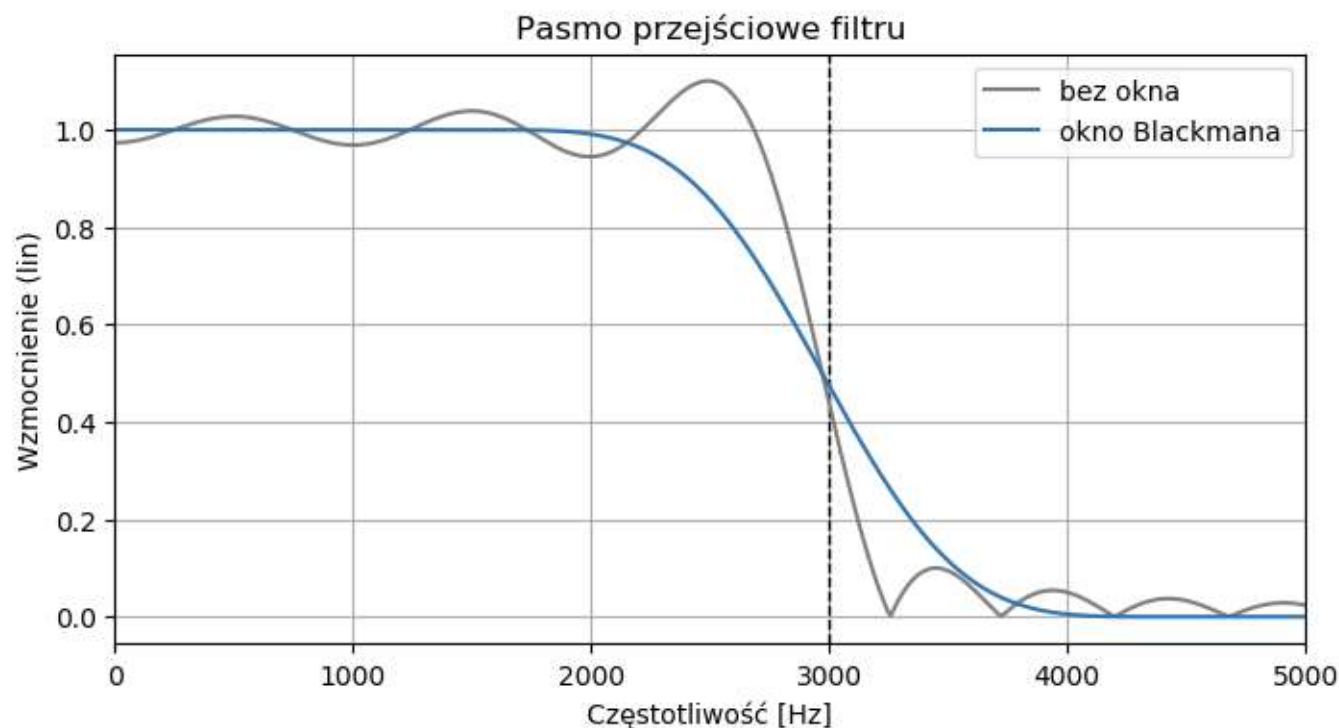
1. Zafalowania charakterystyki w paśmie przepustowym
- korygowane przez funkcję okna.



Zniekształcenia w wyniku przycięcia odp. imp.

2. **Pasmo przejściowe** – filtr zaczyna tłumić przed i kończy za częstotliwością graniczną.

Funkcje okna **poszerzają** pasmo przejściowe!

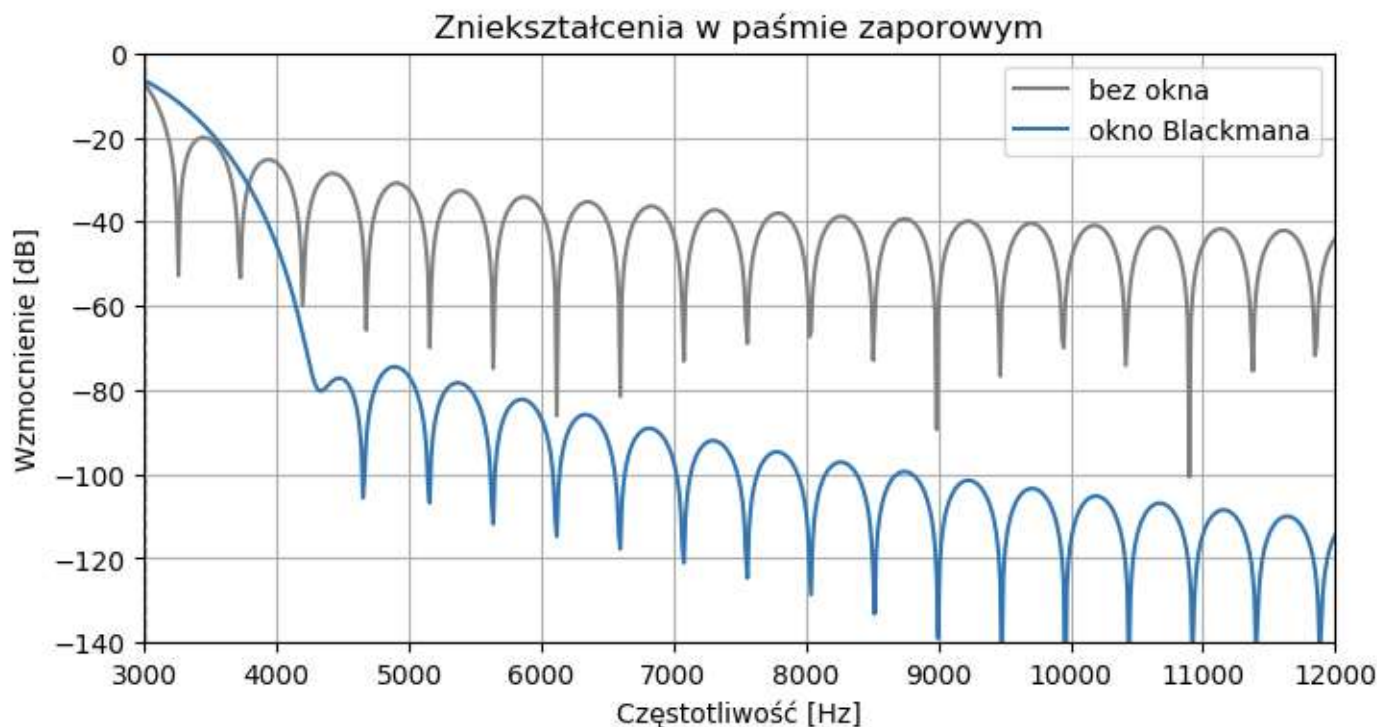


Zniekształcenia w wyniku przycięcia odp. impulsowej

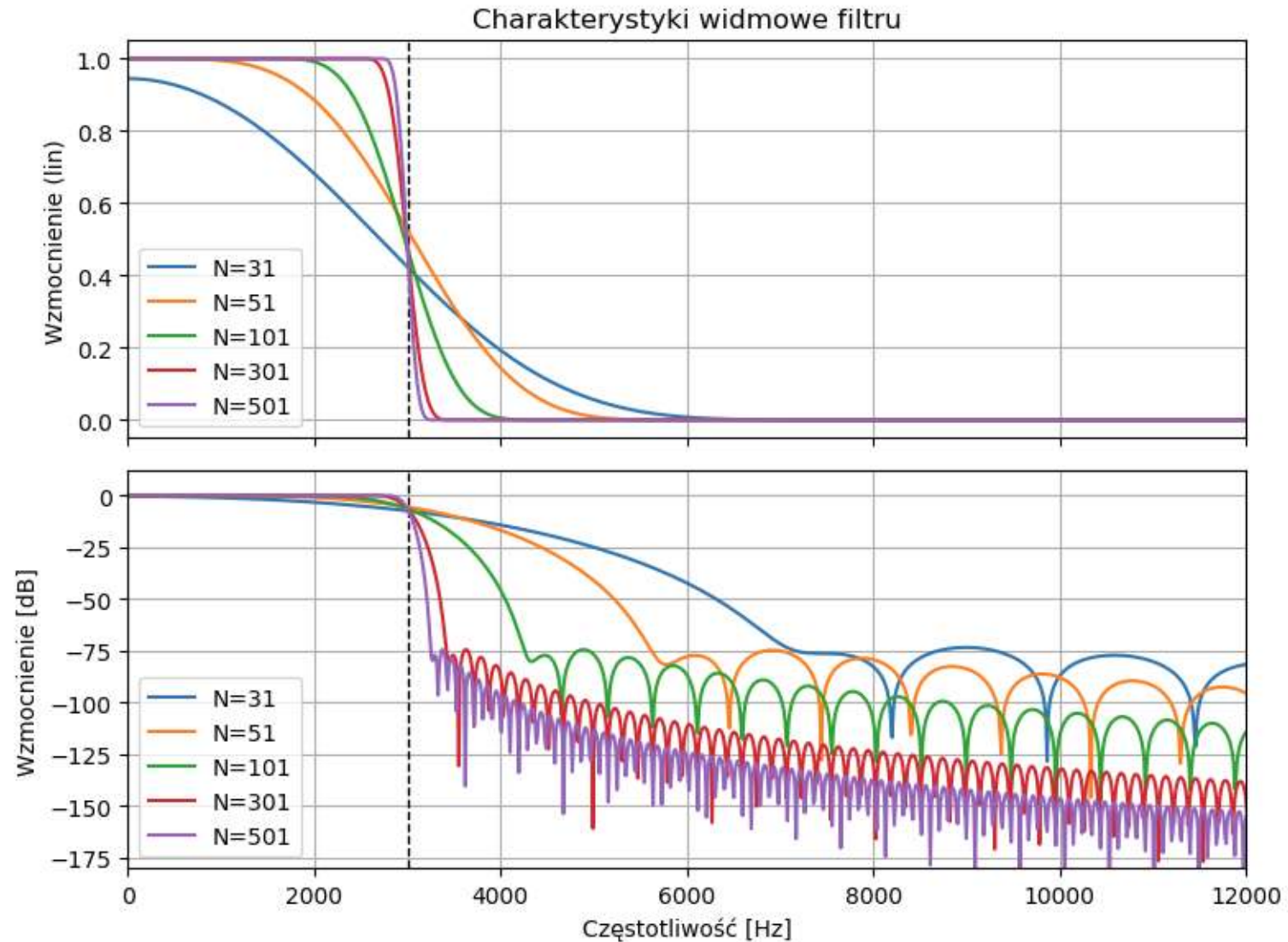
3. Zafalowania w paśmie zaporowym

(zmniejszone tłumienie sygnału w tym paśmie)

- redukowane przez funkcję okna.



Jaki wpływ na kształt charakterystyki ma długość filtru?



Jakie są zalety filtrów o większej długości?

Im większe N , tym bliżej jesteśmy charakterystyki idealnego filtra. A zatem:

- węższe pasmo przejściowe,
- mniejsze zafalowania w paśmie przepustowym,
- większe tłumienie w paśmie zaporowym,
- ogólnie bardziej skuteczna filtracja.

A więc: lepiej zawsze używać możliwie długich filtrów?

Jakie są wady filtrów o większej długości?

Większe N oznacza:

- więcej obliczeń (mnożenia, dodawania), dłuższy czas potrzebny na przeprowadzenie filtracji,
- większą zajętość pamięci (bufor próbek, tablica współczynników),
- **większe opóźnienie** między wejściem a wyjściem filtru – to jest największa wada.

Wpływ okna na charakterystyki filtru

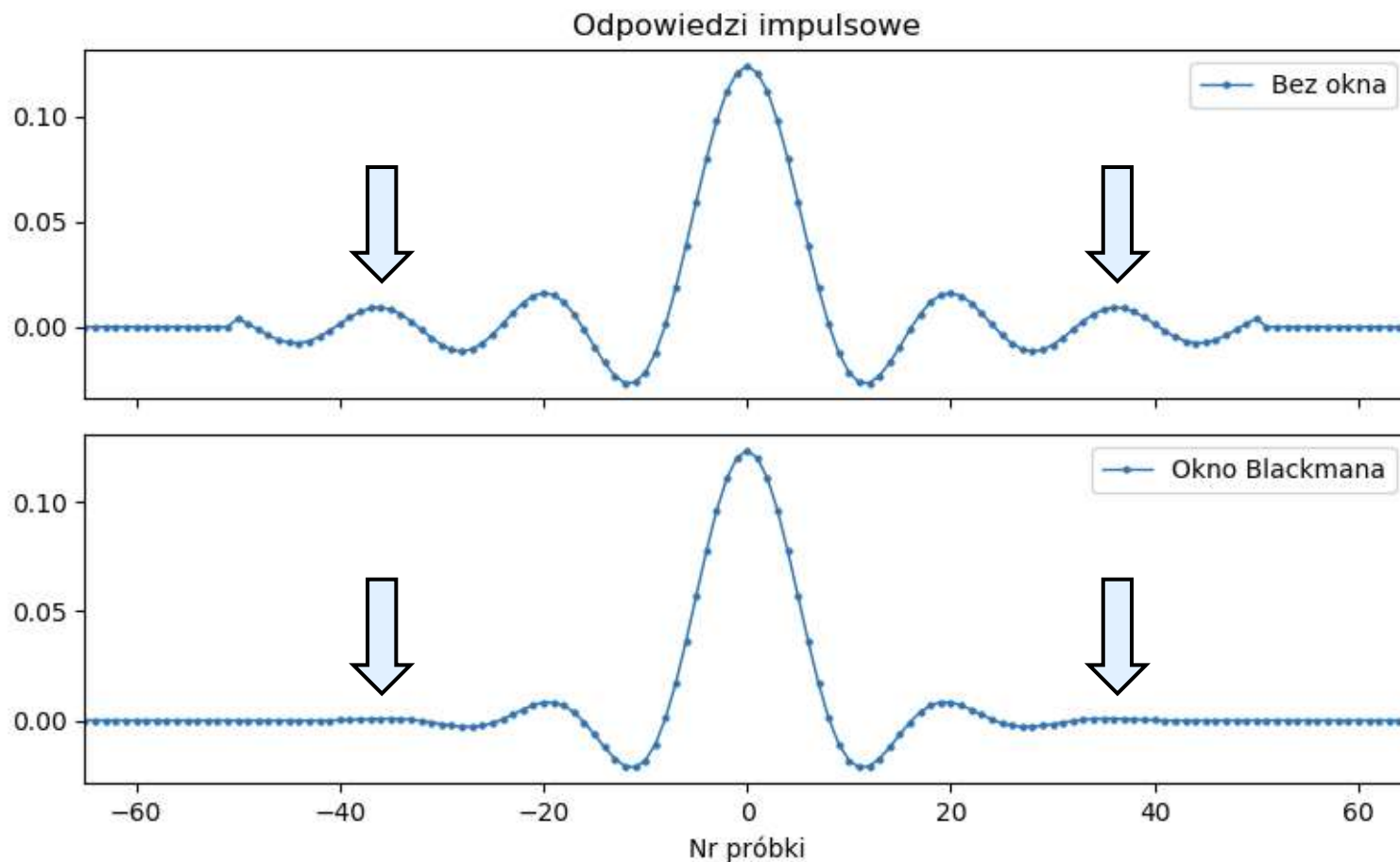
Przybliżona szerokość pasma przejściowego oraz minimalne tłumienie w paśmie zaporowym przy długości filtru N i cz. próbkowania f_s :

Okno	Szerokość	Tłumienie
brak	$0,9 f_s / N$	21 dB
von Hann	$3,1 f_s / N$	44 dB
Hamming	$3,3 f_s / N$	53 dB
Blackman	$5,5 f_s / N$	74 dB

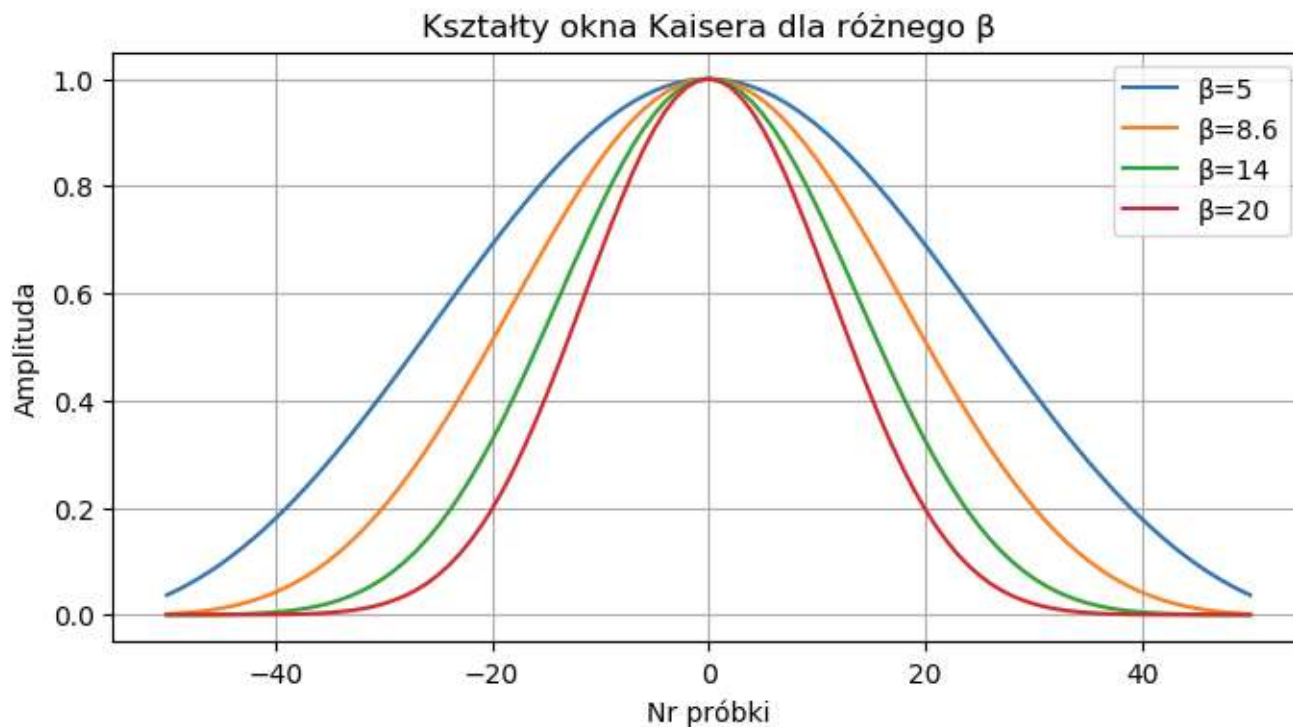
Wpływ okna na charakterystyki filtru

- Użycie okna powoduje zwiększenie (polepszenie) tłumienia w paśmie zaporowym.
- Zmniejszane są zafalowania w paśmie przepustowym.
- Wytłumienie współczynników na końcach okna powoduje zmniejszenie efektywnej długości filtru – poszerzenie pasma przejściowego.
- Trzeba to zrekompensować zwiększeniem długości filtru.

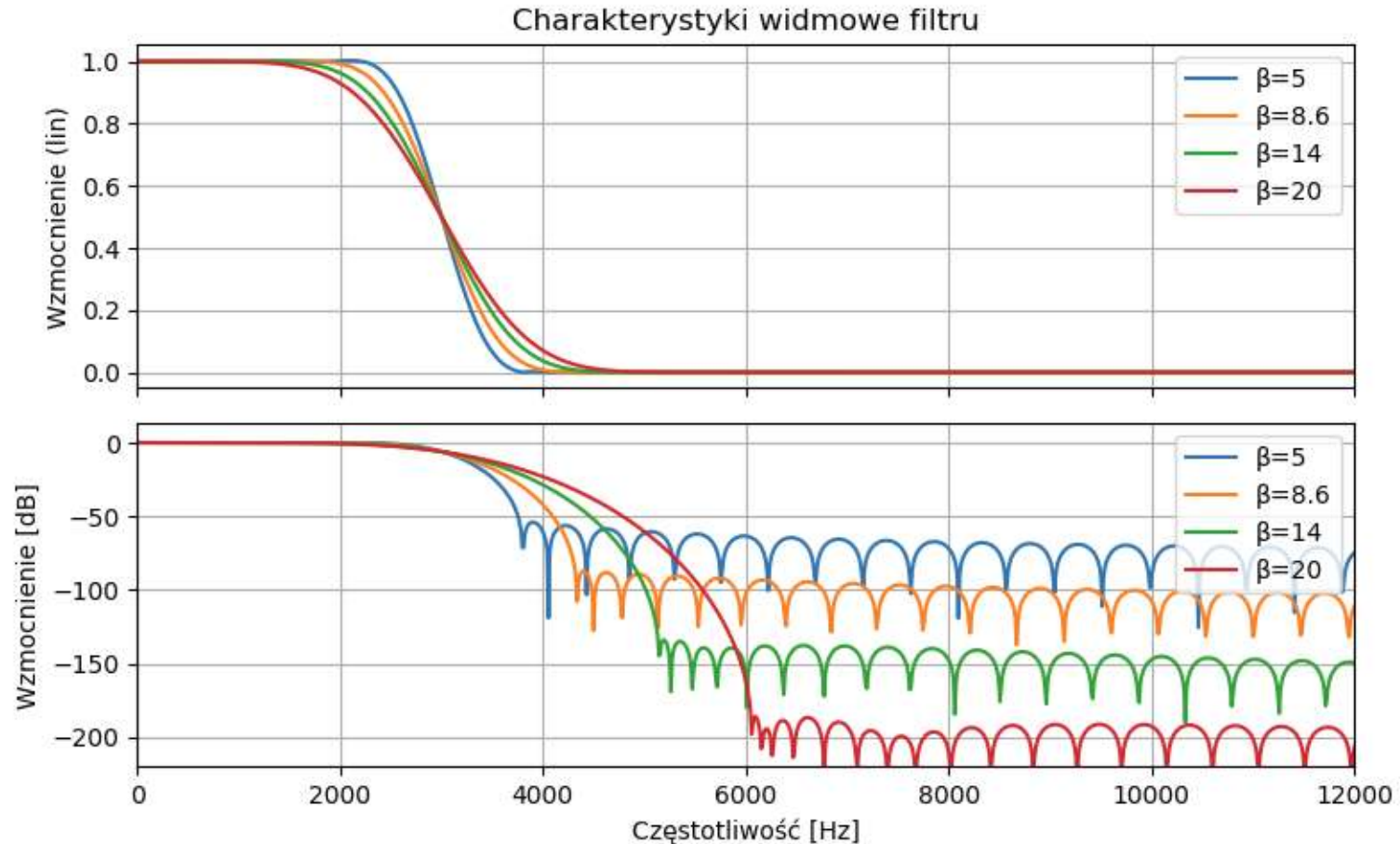
Okno tłumi skrajne wartości odp. imp. zatem efektywna długość odp. imp. jest mniejsza niż bez okna.



Okno Kaisera jest użyteczne w projektowaniu filtrów. Posiada parametr β , który decyduje o kształcie okna. Pozwala on regulować kształt charakterystyki filtru.



Charakterystyki widmowe filtru ($N=101$) dla okna Kaisera z różnymi wartościami β :



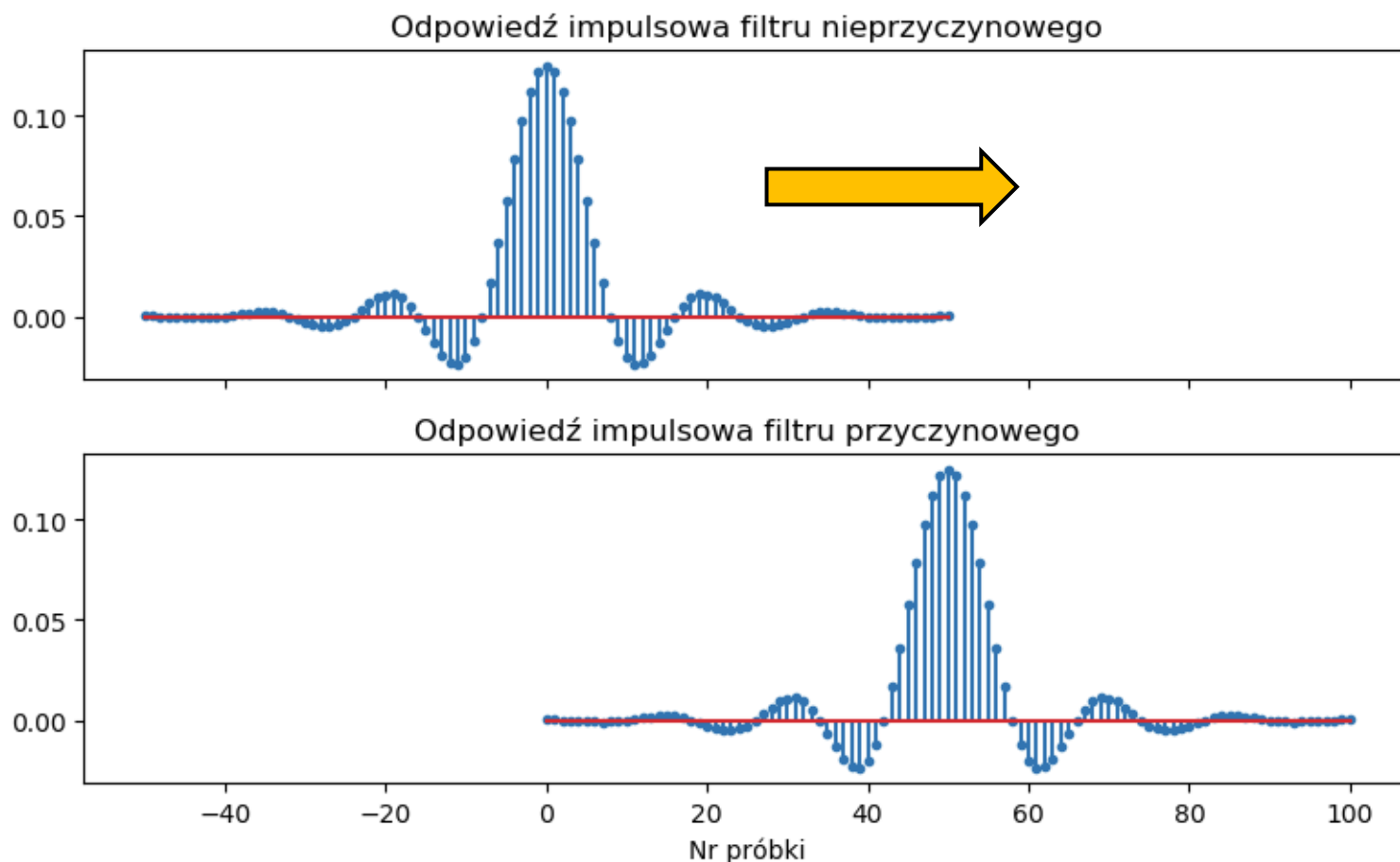
Zastosowanie okna Kaisera do projektowania filtru:

- zakładamy minimalne tłumienie w p. zaporowym i maksymalny poziom zafalowań w p. przepustowym – bierzemy większą z tych wartości (w dB),
- zakładamy szerokość pasma przejściowego,
- obliczamy β dające zadane tłumienie (ze wzoru),
- obliczamy długość filtru dającą zadaną szerokość.

Przykład: zakładamy min. tłumienie pz -80 dB, max. zafalowanie pp 0,005 (46 dB), szerokość p. przejściowego 100 Hz. Stąd obliczamy:

$$\beta = 7,8573 \quad N = 2410$$

Pozostaje drugi problem – nieprzyczynowy filtr.
Możemy go łatwo rozwiązać przesuwając przyciętą
odpowieź impulsową tak, aby zaczynała się w zerze.



Praktyczna interpretacja tego przesunięcia:

- Filtr nieprzyczynowy o długości $N=2M+1$ potrzebuje:
 - M poprzednich próbek sygnału,
 - bieżącą próbkę sygnału,
 - M przyszłych próbek sygnału.
- Nie mamy jeszcze M przyszłych próbek, musimy na nie poczekać M okresów próbkowania.
- Wynik filtracji bieżącej próbki pojawi się na wyjściu filtru po M okresach próbkowania (opóźnienie).

$$M = \frac{N-1}{2}$$

Charakterystyka częstotliwościowa filtru FIR:

$$H(f) = \operatorname{Re} H(f) + j \operatorname{Im} H(f)$$

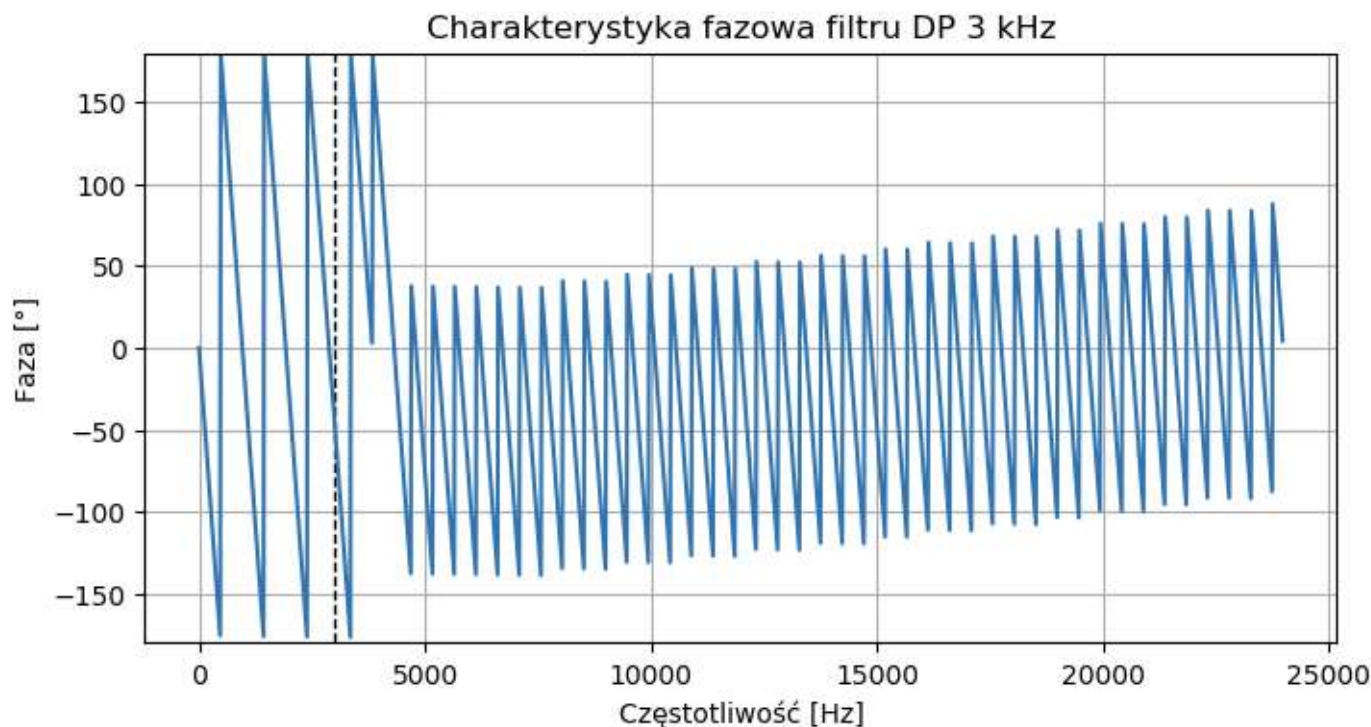
$$H(f) = |H(f)| \cdot e^{j\varphi(f)}$$

$|H(f)|$ – charakterystyka amplitudowa

$\varphi(f)$ – charakterystyka fazowa

$$\varphi(f) = \arg(H(f)) = \arctan\left(\frac{\operatorname{Im} H(f)}{\operatorname{Re} H(f)}\right)$$

Charakterystyka fazowa filtru FIR projektowana metodami opisanymi na wykładzie jest zawsze (odcinkami) liniowa. Nieciągłości fazy wynikają z jej cykliczności (zawijanie wokół 2π).



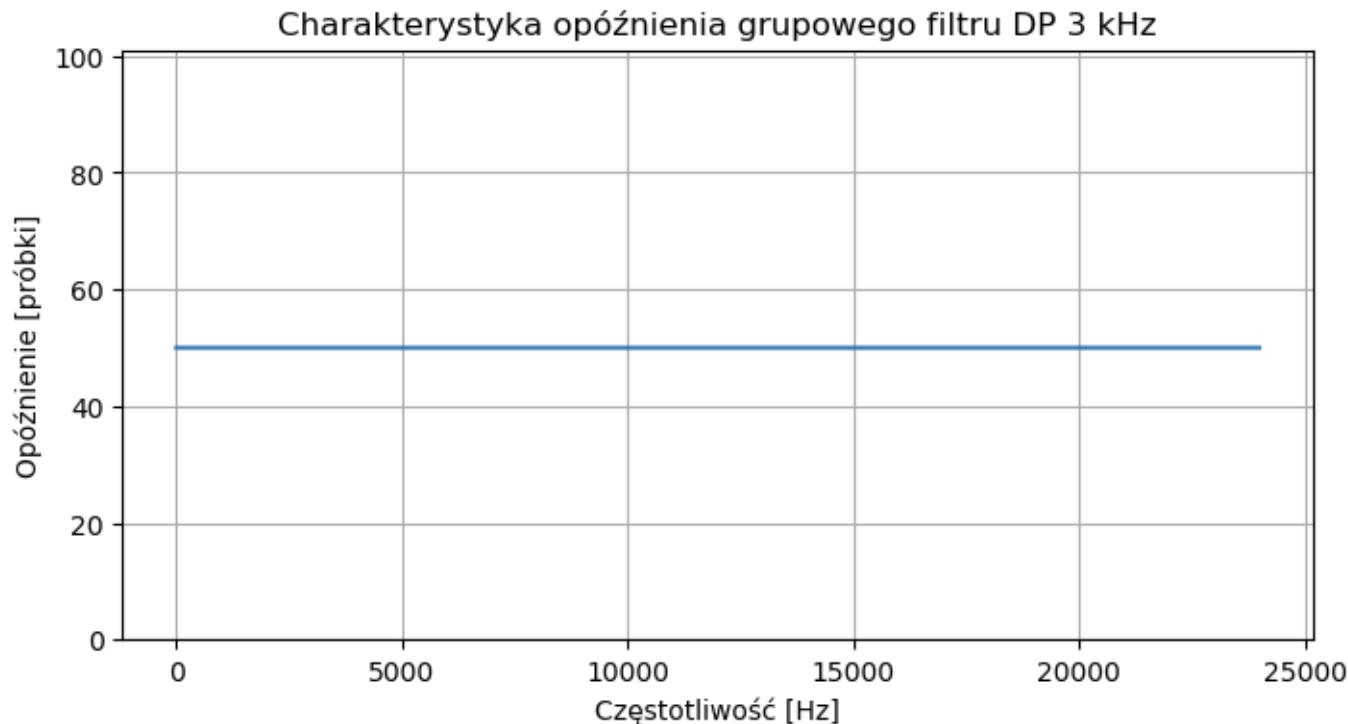
Co nam daje liniowość fazy filtru FIR?

Opóźnienie grupowe (*group delay*) jest równe ujemnej pochodnej charakterystyki fazowej:

$$D(f) = -\frac{d\varphi(f)}{df}$$

Opóźnienie grupowe dla częstotliwości f
= nachylenie charakterystyki fazowej
dla tej częstotliwości (z odwróconym znakiem).

Charakterystyka opóźnienia grupowego filtru FIR

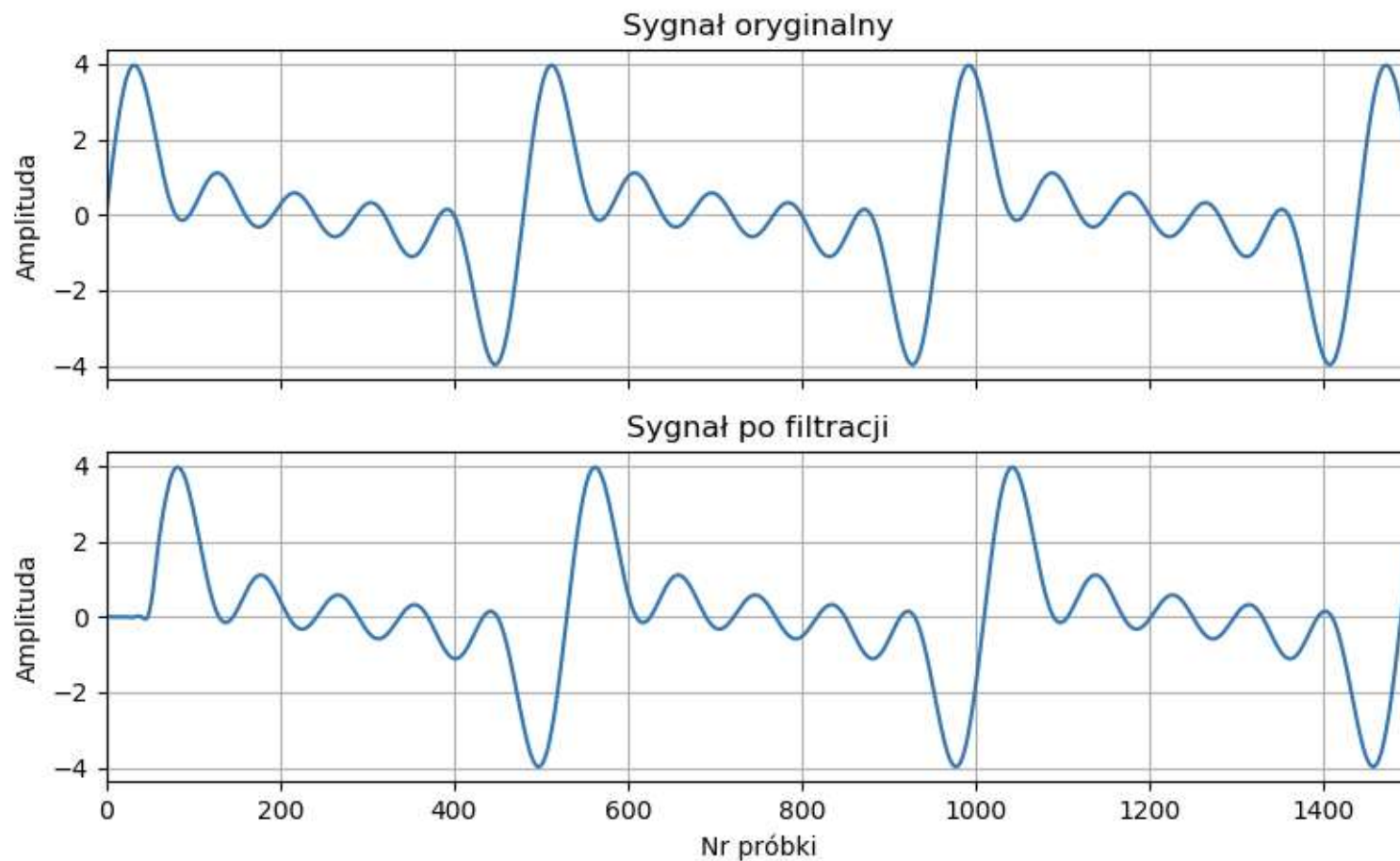


Liniowa faza = stała pochodna = **stałe opóźnienie grupowe**, równe $(N-1)/2$, czyli tyle, o ile przesunęliśmy odpowiedź impulsową.

Co nam daje stałe opóźnienie grupowe?

- Jeżeli mamy sygnał o złożonym widmie (np. mowa, muzyka), to wszystkie składowe widmowe są opóźniane przez filtr o tę samą liczbę próbek.
- Zależności fazowe między składowymi widmowymi na wyjściu filtru są takie same, jak na wejściu.
- Liniowofazowy filtr FIR **nie wprowadza zniekształceń fazowych** – jest to ważna cecha tych filtrów.
- (Można projektować filtry FIR, które celowo nie mają liniowej fazy.)

Ilustracja opóźnienia grupowego
– widoczne opóźnienie sygnału po filtracji.



Opisana metoda projektowania filtrów FIR nosi nazwę **metody okienkowania** (*windowing*).

Podsumowując:

- projektujemy idealną charakterystykę w dziedzinie częstotliwości,
- obliczamy odpowiedź impulsową (IFFT),
- przycinamy funkcją okna do żądanej długości,
- przesuwamy na osi czasu tak aby zaczynała się w 0.

I gotowe, mamy współczynniki filtra.

Dla typowych charakterystyk (DP, GP, PP, PZ) zwykle obliczamy odpowiedź impulsową bezpośrednio, bez konieczności liczenia IFFT.

Dla filtru **dolnoprzepustowego** o częstotliwości granicznej f_c i cz. próbkowania f_s :

(nie uczyć się wzorów na pamięć!)

$$h[n] = \frac{\sin(2\pi n f_c / f_s)}{n\pi} = 2\pi \frac{f_c}{f_s} \operatorname{sinc}\left(2 \frac{f_c}{f_s} n\right)$$

$$\operatorname{sinc}(x) = \sin(\pi x)/(\pi x)$$

Obliczoną odpowiedź trzeba przemnożyć przez okno i przesunąć o $(N-1)/2$.

Filtr **górnoprzepustowy** można otrzymać z filtru DP.

W dziedzinie widma: $GP(f) = 1 - DP(f)$.

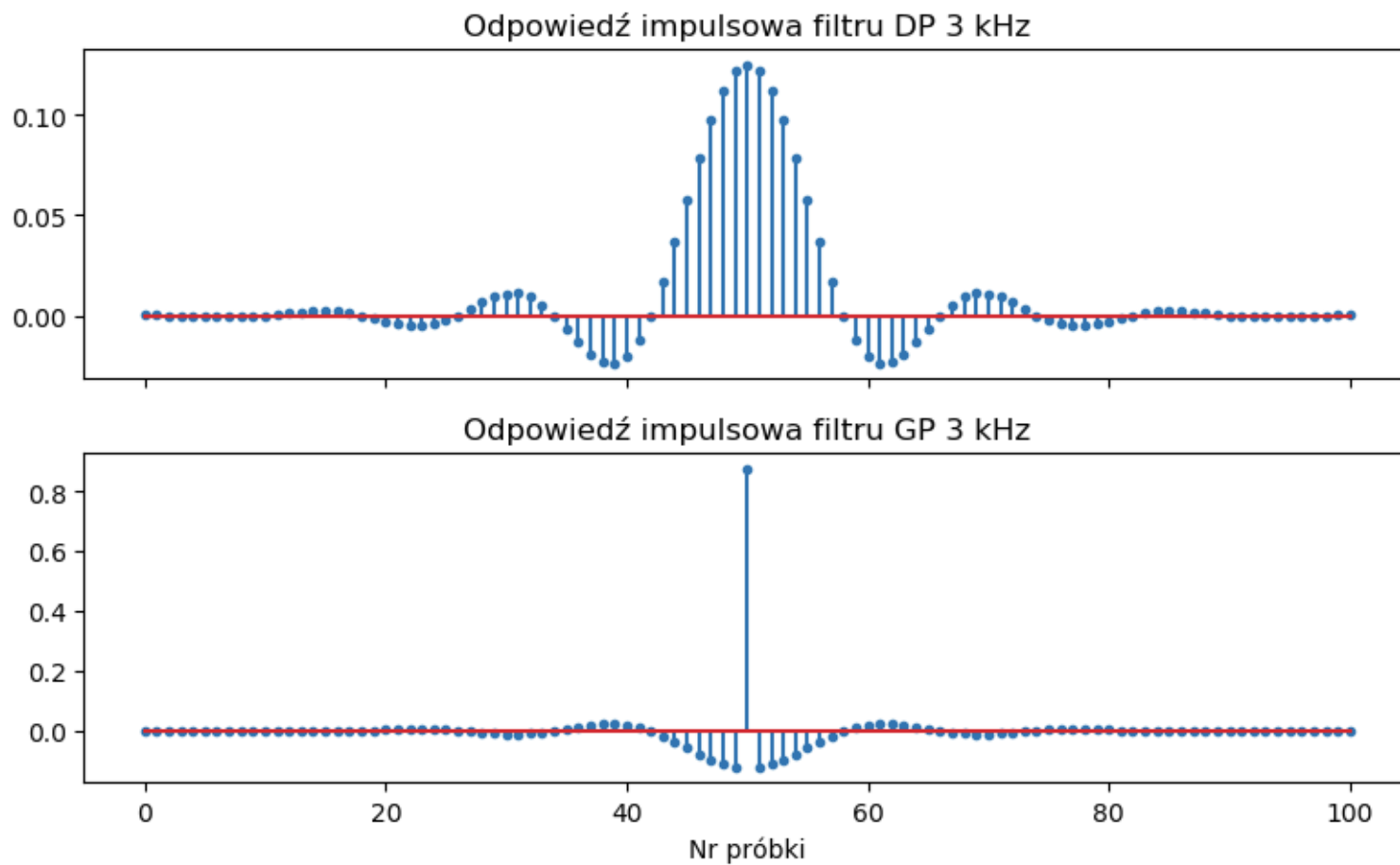
W dziedzinie czasu:

$$h_{GP}[n] = \delta[n] - h_{DP}[n]$$

W praktyce:

- obliczamy odpowiedź dla filtru DP o tej samej częstotliwości granicznej,
- odwracamy znak każdej wartości,
- dodajemy 1 do wartości dla czasu zerowego,
- mnożymy przez okno i przesuwamy.

Porównanie odpowiedzi impulsowych DP i GP



Filtr **pasmowo-przepustowy**:

- widma: $PP(f) = DP(f) \cdot GP(f) = DP_g(f) - DP_d(f)$
- w dziedzinie czasu – dwie metody:

$$h_{PP}[n] = h_{DP}[n] * h_{GP}[n]$$

$$h_{PP}[n] = h_{DP_g}[n] - h_{DP_d}[n]$$

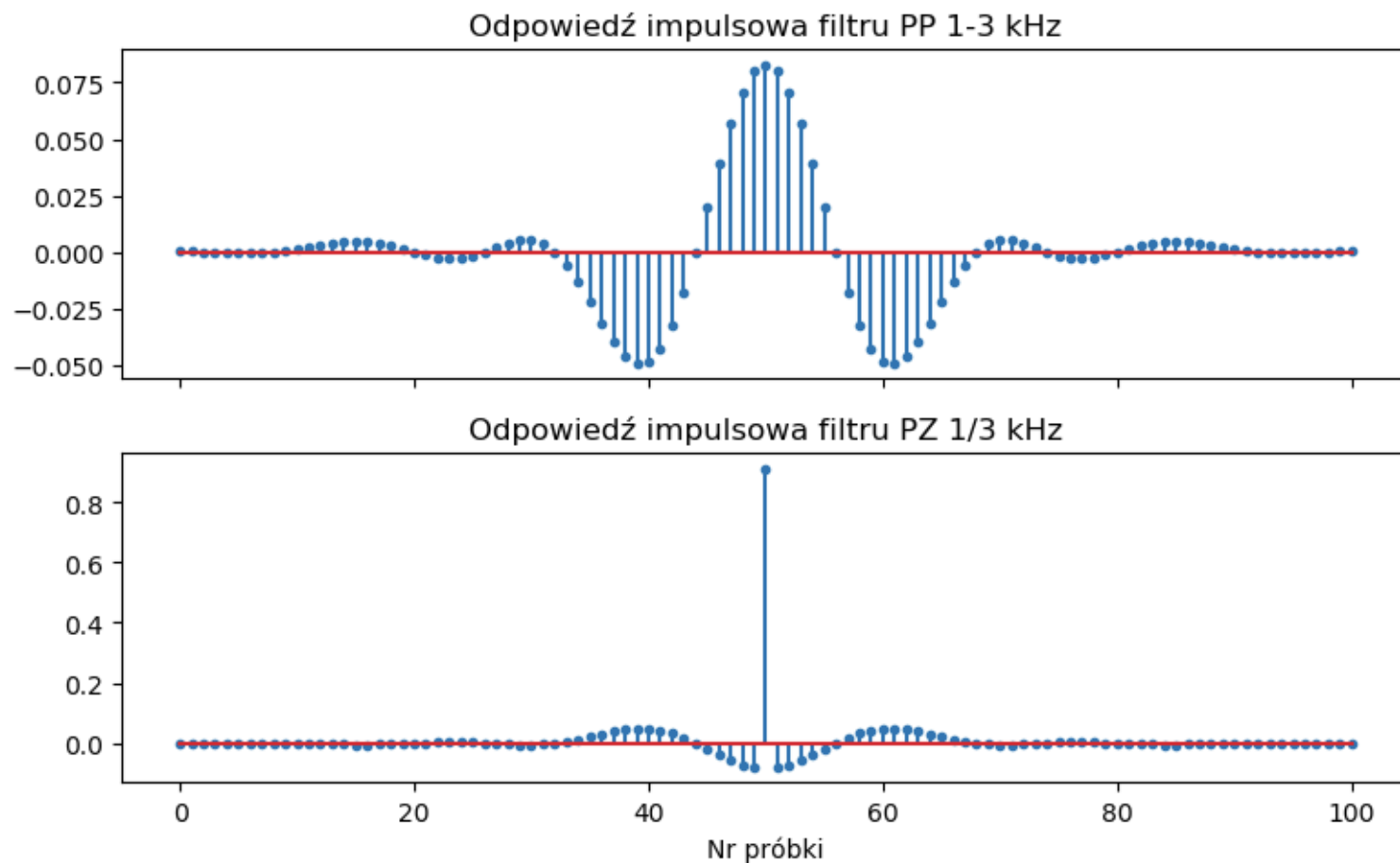
Filtr **pasmowo-zaporowy**:

- widma: $PP(f) = DP(f) + GP(f) = 1 - PP(f)$
- w dziedzinie czasu – dwie metody:

$$h_{PZ}[n] = h_{DP}[n] + h_{GP}[n]$$

$$h_{PZ}[n] = \delta[n] - h_{PP}[n]$$

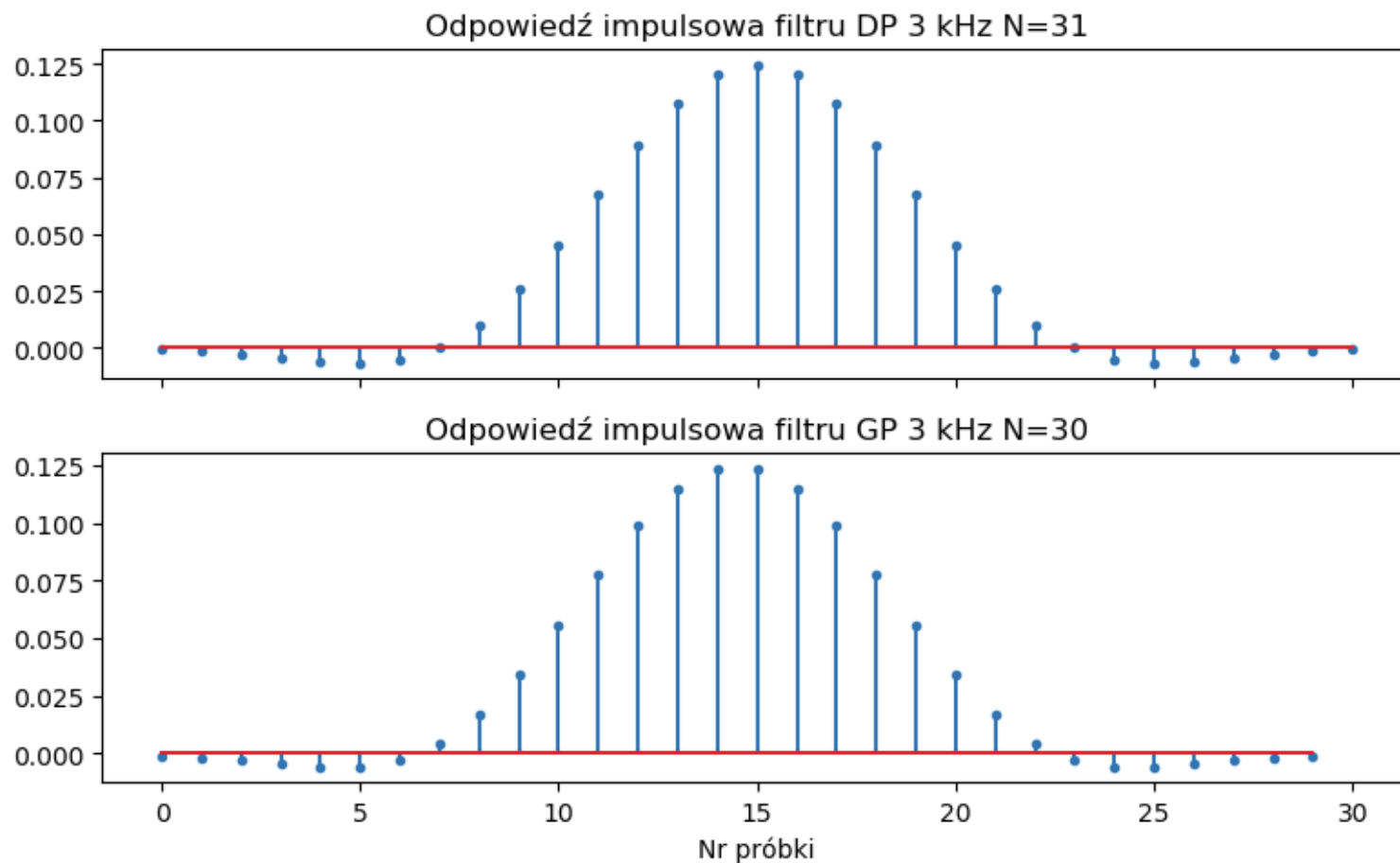
Porównanie odpowiedzi impulsowych PP i PZ



Projektując filtr w ten sposób możemy różnie dobrać długość filtru N .

- Nieparzysta długość filtru N
 - filtr FIR typu pierwszego (I)
 - środkowa próbka wyznacza symetrię
 - możliwe wszystkie charakterystyki: DP, GP, PP, PZ.
- Parzysta długość filtru N
 - filtr FIR typu drugiego (II)
 - dwie symetryczne połowy
 - wzmacnienie dla cz. Nyquista musi być zerowe!
 - możliwe więc tylko filtry: DP i PP.

Odpowiedź impulsowa filtru typu I i II



Możliwe jest również odwrócenie znaku „lewej” części odpowiedzi impulsowej – powstają w ten sposób filtry **antysymetryczne**:

- typ **III** – nieparzysty
 - zerowe wzmocnienie dla cz. 0 i Nyquista
 - tylko filtry **PP**
- typ **IV** – parzysty
 - zerowe wzmocnienie dla częstotliwości 0
 - tylko filtry **GP i PP**

Typy III i IV są używane tylko do specjalnych przypadków (np. filtr Hilberta).

Typy filtrów FIR zebrane razem:

Typ	Symetria	Długość	DP	GP	PP	PZ
I	symetr.	nieparzysta	+	+	+	+
II	symetr.	parzysta	+	−	+	−
III	asym.	nieparzysta	−	−	+	−
IV	asym.	parzysta	−	+	+	−

- Najczęściej korzysta się z typu I (uniwersalny).
- Filtr typu II jest trochę prostszy w implementacji (dwie symetryczne połowy) – wykorzystuje się czasem do filtrów DP i PP.
- Z typów III i IV nie korzysta się do typowych filtrów.

Normalizacja wzmocnienia filtru

- Zwykle chcemy aby wzmocnienie filtru w paśmie przepustowym było równe 1.
- Wymagana jest więc normalizacja wzmocnienia.
- Dla filtru DP sprawa jest prosta:
 - możemy wybrać do normalizacji częstotliwość 0,
 - wzmocnienie na cz. 0 jest równe sumie współczynników odpowiedzi impulsowej,
 - zatem dzielimy każdy współczynnik przez sumę wszystkich współczynników.

Normalizacja wzmacnienia filtru

- Dla filtru GP wybieramy częstotliwość Nyquista:

$$s = \sum_{n=-M}^M (h[n] \cos(\pi n))$$

- Dla filtrów PP i PZ wybieramy częstotliwość środkową f – średnią dolnej i górnej cz. granicznej.

$$s = \sum_{n=-M}^M \left(h[n] \exp \left(j 2 \pi n \frac{f}{f_s} \right) \right)$$

- Dzielimy współczynniki przez s .

Optymalne metody projektowania filtrów

- Metoda okienkowa jest prostą i działającą metodą projektowania filtrów, ale nie jest optymalna.
- „Optymalny” oznacza „najlepszy z możliwych”.
- W naszym przypadku „optymalny” oznacza zaprojektowaną charakterystykę $H(f)$, która jest możliwie najbliższa idealnej charakterystyce $D(f)$.
- Optymalizacja jest przeprowadzana za pomocą skomplikowanych algorytmów matematycznych.

Metoda najmniejszych kwadratów

(*least squares*, LS, LSQ)

- Dla zaprojektowanej charakterystyki $H(f)$ i idealnej charakterystyki $D(f)$, mamy funkcję błędu:

$$E = \sum_{i=0}^n w_i \left(|D(f_i)| - |H(f_i)| \right)^2$$

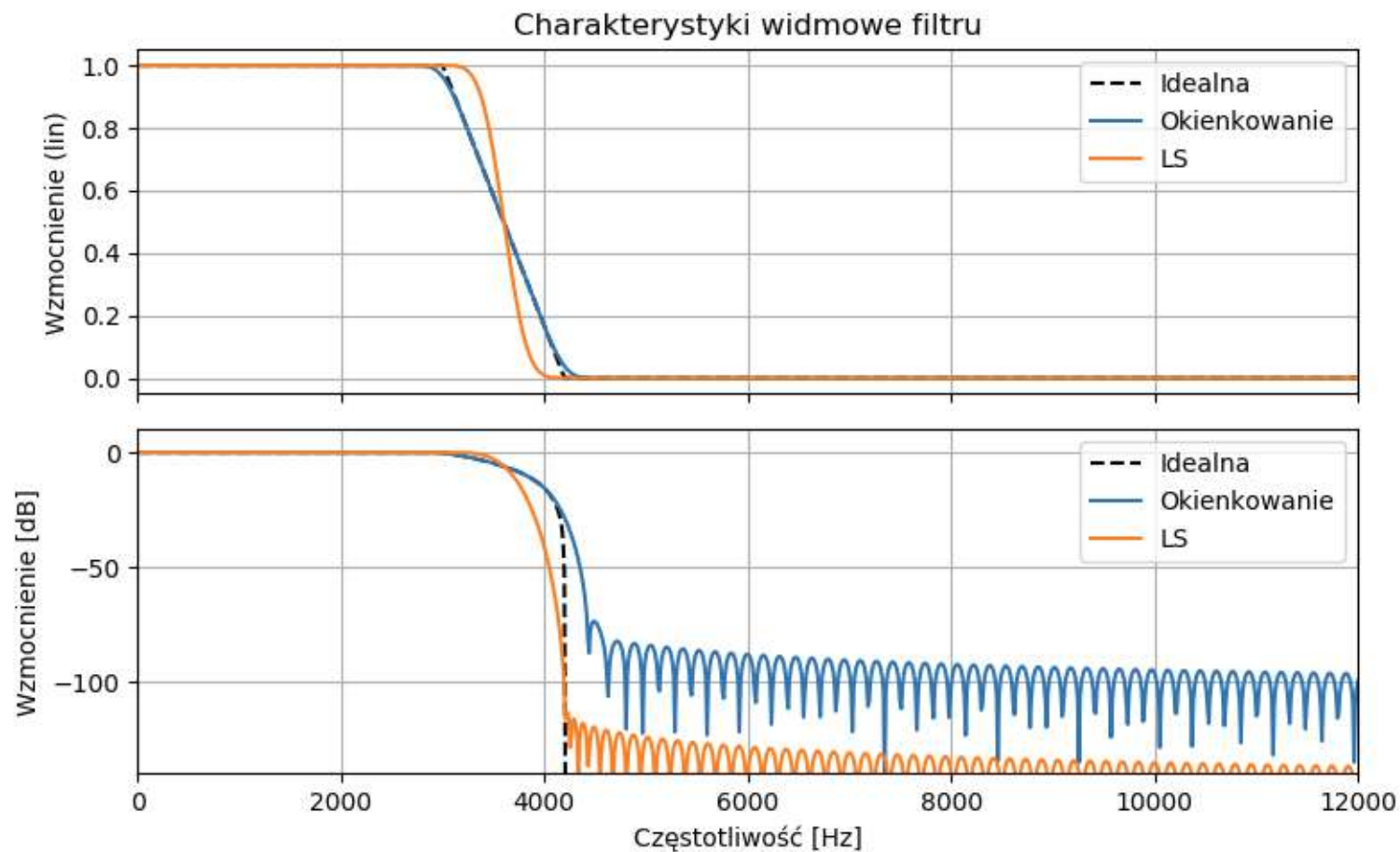
- Algorytm minimalizuje funkcję E – znajduje zbiór współczynników dający jak najmniejszy błąd.
- Obliczenia są wykonywane za pomocą operacji na macierzach (nie iteracyjnie).

Metoda najmniejszych kwadratów

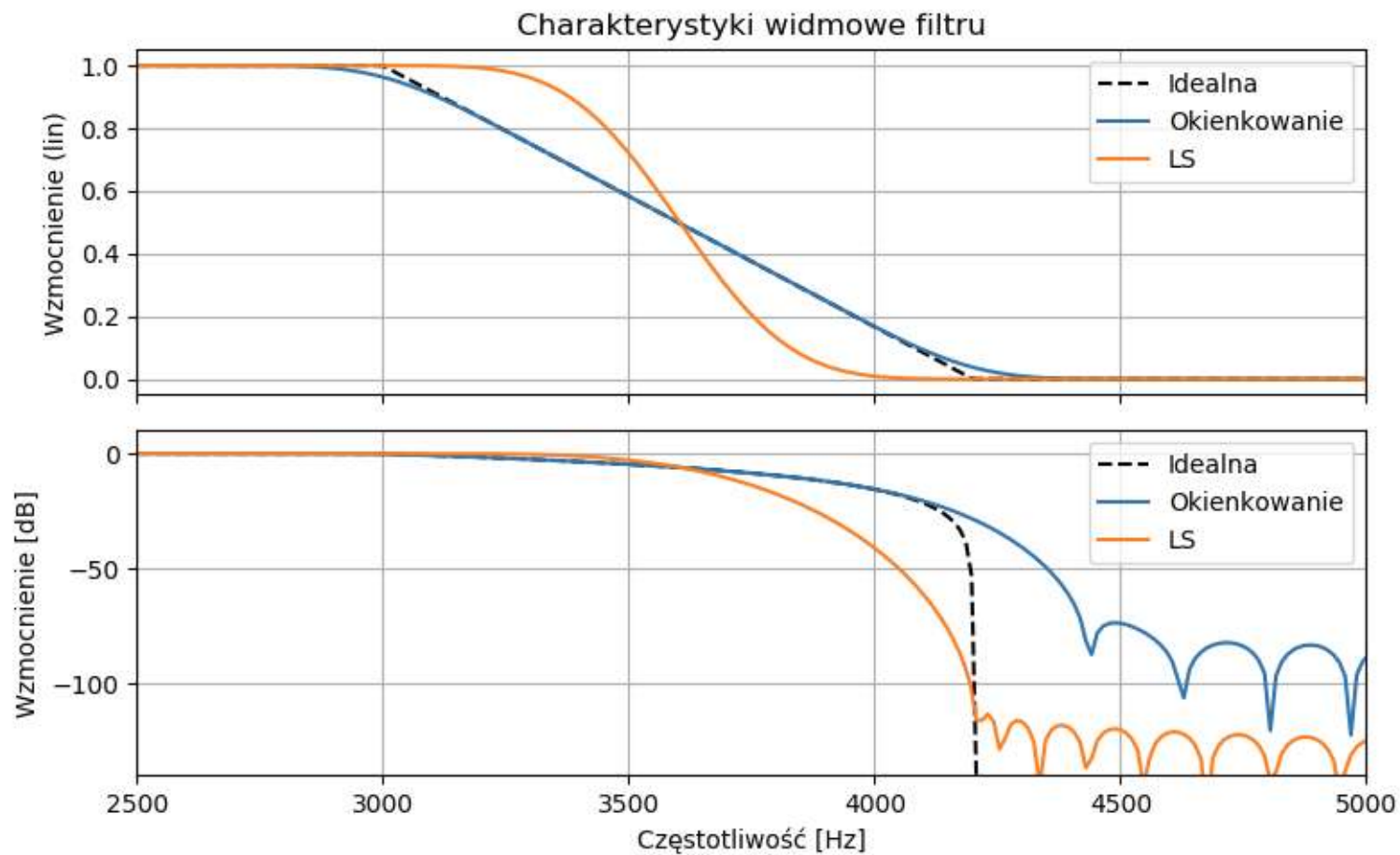
- Podajemy skrajne częstotliwości pasm przepustowych i zaporowych oraz wzmocnienie w każdym z pasm.
- Możliwe jest nadanie wag poszczególnym zakresom częstotliwości. Np. większa waga dla pasma zaporowego = dopuszczamy mniejsze błędy w tym zakresie.
- Algorytm dopasowuje charakterystykę do zadanych parametrów.
- W pasmach przejściowych kształt charakterystyki jest nieokreślony („jak wyjdzie”).

Porównanie metody LS i okienkowania

filtr DP $N=301$, pasmo przejściowe 3-4,2 kHz



Zmiana pasma przejściowego wokół cz. granicznej



- Metoda LS stara się objąć pełne pasma przepustowe i zaporowe.
- Efektem jest „przeciąganie” charakterystyki na pasmo przejściowe.
- Większa długość filtru daje większe tłumienie w paśmie zaporowym i bardziej strome opadanie charakterystyki w paśmie przejściowym.
- Wadą filtrów LS są nierównomierne zafalowania w pasmach zaporowych i przepustowych.

Metoda Parksa-McClellana

nazywana też *Remez* (niepoprawnie, ale prościej zapamiętać)

- Algorytm iteracyjny.
- Także stara się dopasować do idealnej charakteryst.
- Oblicza współczynniki stosując aproksymację Czebyszewa (algorytm Remeza).
- W kolejnych iteracjach koryguje współczynniki tak, aby zminimalizować maksymalny błąd.

$$\min_{h(n)} E = \min \left\{ \max_i \left[w_i (D(f_i) - H(f_i)) \right] \right\}$$

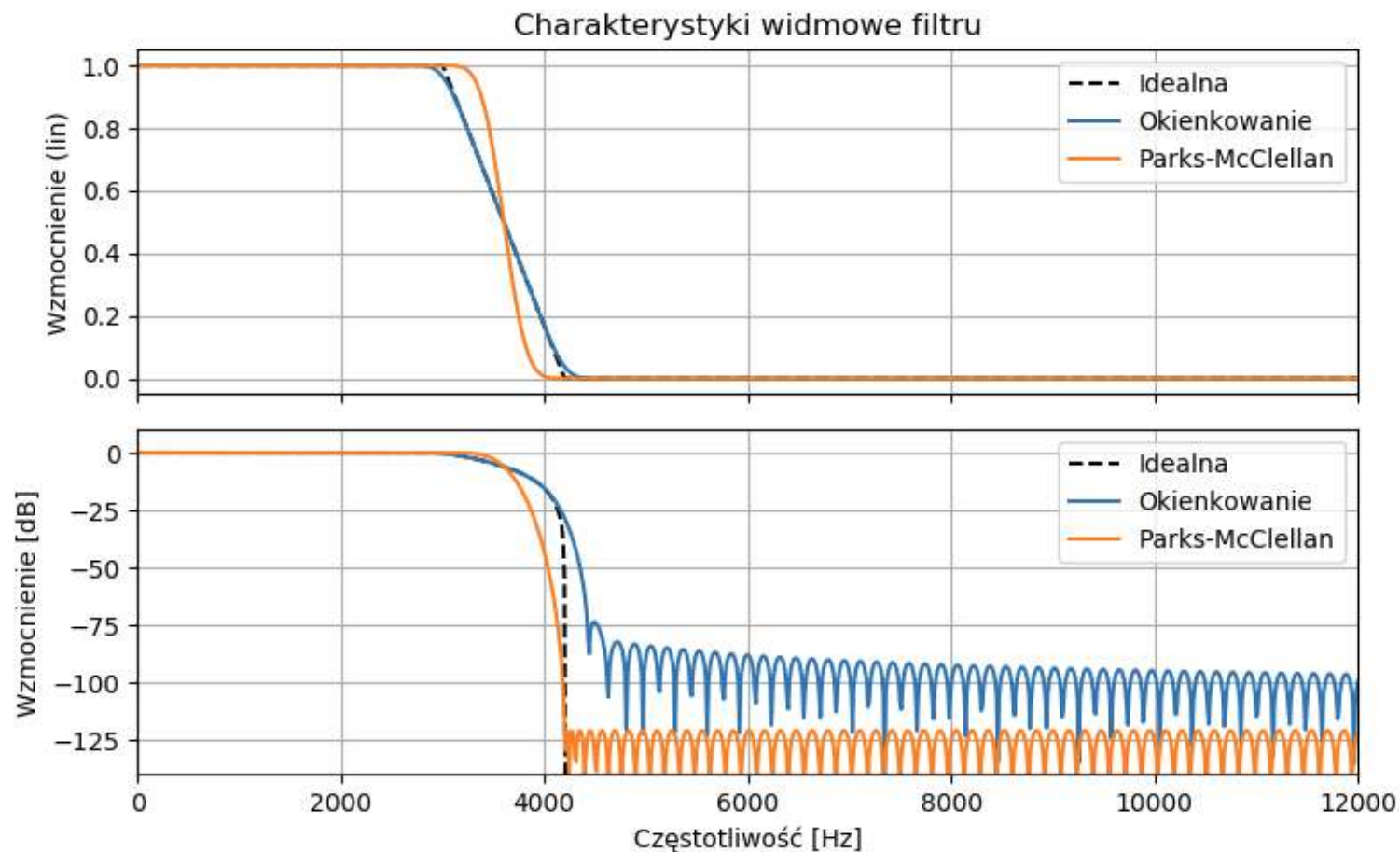
- Algorytm kończy działanie gdy kolejne iteracje nie zmniejszają już błędu E .

Metoda Parksa-McClellana

- Założenia projektowe są identyczne, jak dla metody LS – częstotliwości graniczne i wzmocnienia pasm, ewentualnie wagi dla poszczególnych pasm.
- Algorytm jest wrażliwy na dobór parametrów początkowych. W niektórych przypadkach nie jest w stanie uzyskać zbieżności i otrzymujemy błąd zamiast wyniku. Trzeba skorygować parametry.
- Algorytm opracowany w 1972 r. stał się standardem w projektowaniu filtrów FIR za pomocą programów komputerowych.
- Jest wolny (co nie ma praktycznego znaczenia).

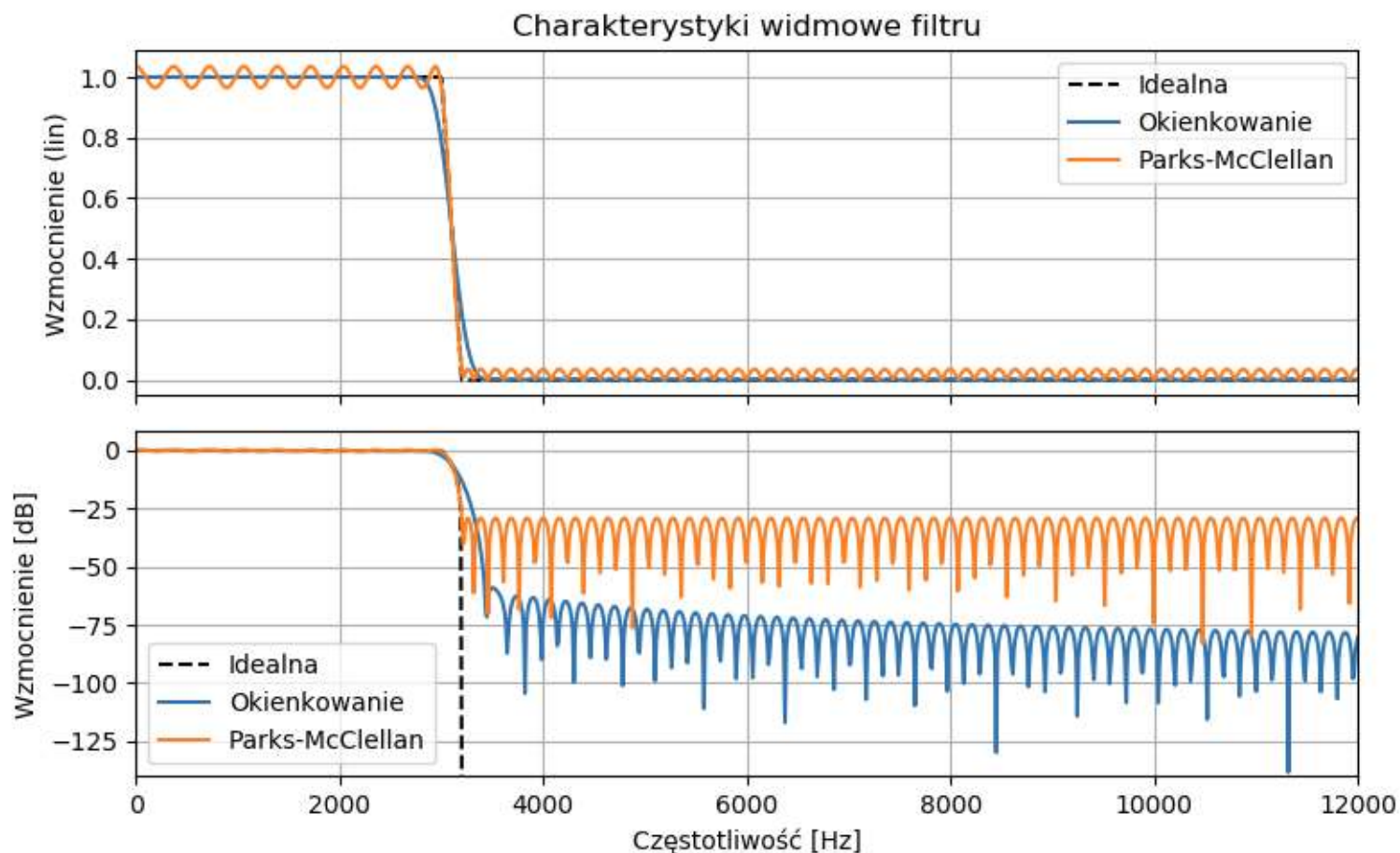
Porównanie metody P-M i okienkowania

filtr DP $N=301$, pasmo przejściowe 3-4,2 kHz



- Podobnie jak w metodzie LS, metoda P-M dopasowuje się do pasm przepustowych i zaporowych. Pasma przejściowe są nieokreślone.
- Również „przeciągane” są pasma przejściowe.
- Tłumienie w paśmie zaporowym wynika z długości filtru i szerokości pasma przejściowego.
- Różnica w porównaniu z LS: **poziom zafalowań jest stały** dla wszystkich częstotliwości, we wszystkich pasmach przepustowych i zaporowych.
- Z tego względu, metodę P-M często nazywa się metodą projektowania filtrów o równomiernych zafalowaniach – *equiripple*.

Przykład dla zbyt wąskiego pasma przejściowego (3-3,2 kHz). Filtr $N=301$ nie jest w stanie zapewnić dobrego tłumienia. Trzeba zwiększyć długość filtra.



Podsumowanie metod optymalizacyjnych:

- Metoda okienkowania zwykle zaczyna tłumić „za wcześnie” i osiąga pasmo zaporowe „za późno”.
- Metody optymalizacyjne (LS, P-M) zapewniają, że pasma przepustowe i zaporowe są zgodne z zadanymi częstotliwościami.
- Charakterystyka w paśmie przejściowym nie jest określona – nie mamy na nią wpływu.
- W metodach optymalizacyjnych musimy zapewnić dostatecznie szerokie pasmo przejściowe w odniesieniu do długości filtru, aby uzyskać dobre tłumienie w paśmie zaporowym.

Uwagi o częstotliwościach

- Projektując filtry posługujemy się zwykle częstotliwościami w hercach [Hz].
- Dla filtru nie ma znaczenia częstotliwość f_c w Hz. Istotna jest **pulsacja** (od 0 do 2π), liczona względem częstotliwości próbkowania f_s (odpowiada 2π).

$$\omega = 2\pi \frac{f_c}{f_s}$$

- Często podaje się pulsację w formie $(x \cdot \pi)$.
- Czasami pomija się π , podając tylko liczbę od 0 do 1, gdzie 1 oznacza częstotliwość Nyquista.

Uwagi o częstotliwościach

- Inną konwencją jest **częstotliwość unormowana**:

$$f_{cn} = \frac{f_c}{f_s}$$

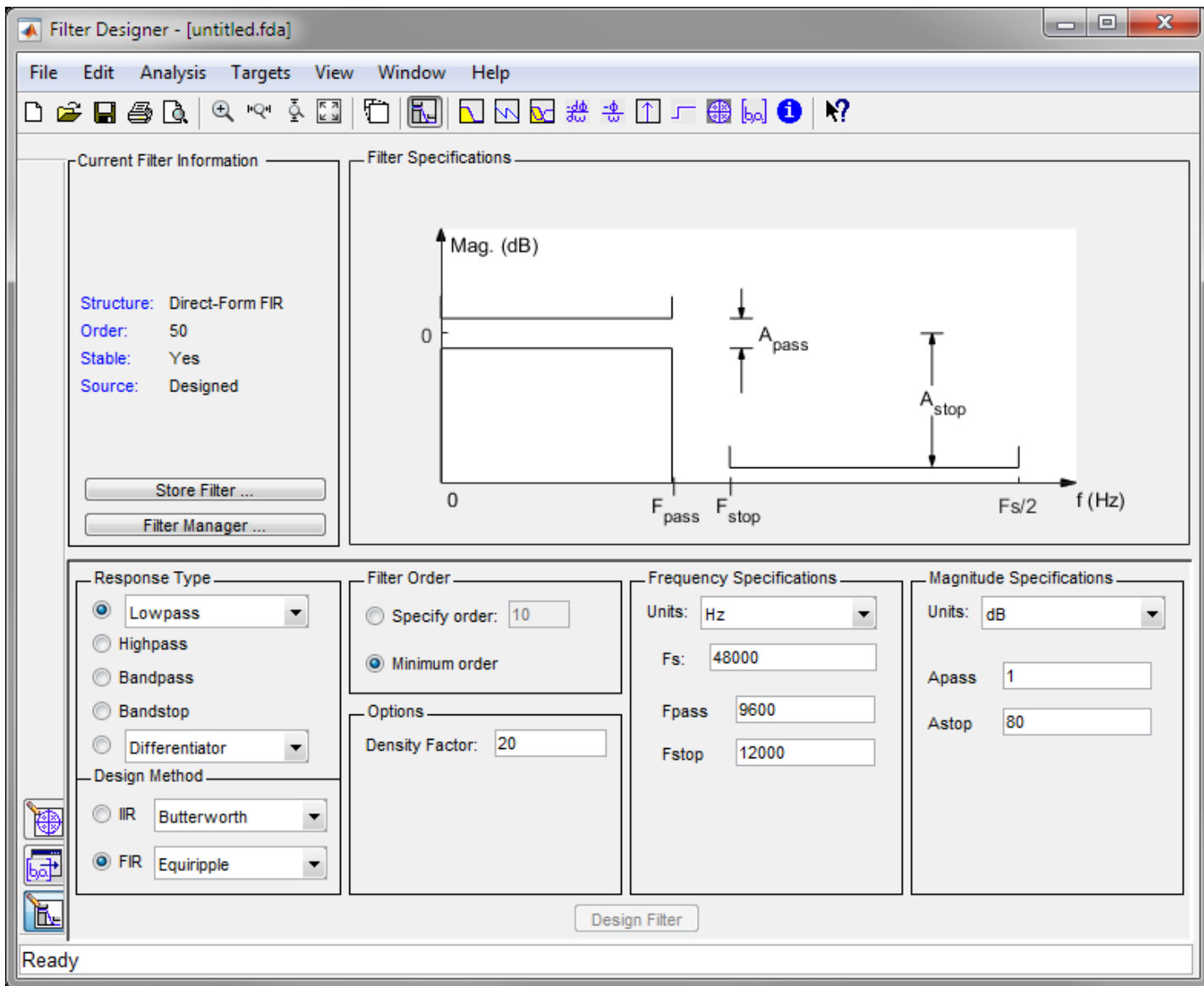
- Częstotliwość unormowana (bez jednostki) przyjmuje wartości od 0 do 1, gdzie 1 oznacza częstotliwość próbkowania.
- Możemy stosować wartości od 0 do 0,5, 0,5 oznacza częstotliwość Nyquista.

Komputerowe projektowanie filtrów FIR

Narzędzia do projektowania filtrów:

- programy z interfejsem GUI – najczęściej komercyjne, wygodne projektowanie,
- funkcje dostępne w językach programowania – duża elastyczność,
- oprogramowanie online w Internecie – zwykle kiepska jakość i skomplikowane interfejsy,
- metodę okienkowania można zaimplementować samodzielnie.

filterDesigner: Matlab + Signal Processing Toolbox



Pokażemy przykłady projektowania filtrów FIR
za pomocą darmowych narzędzi
– języka *Python* z modułami *SciPy* + *NumPy*.

Dokumentacja:

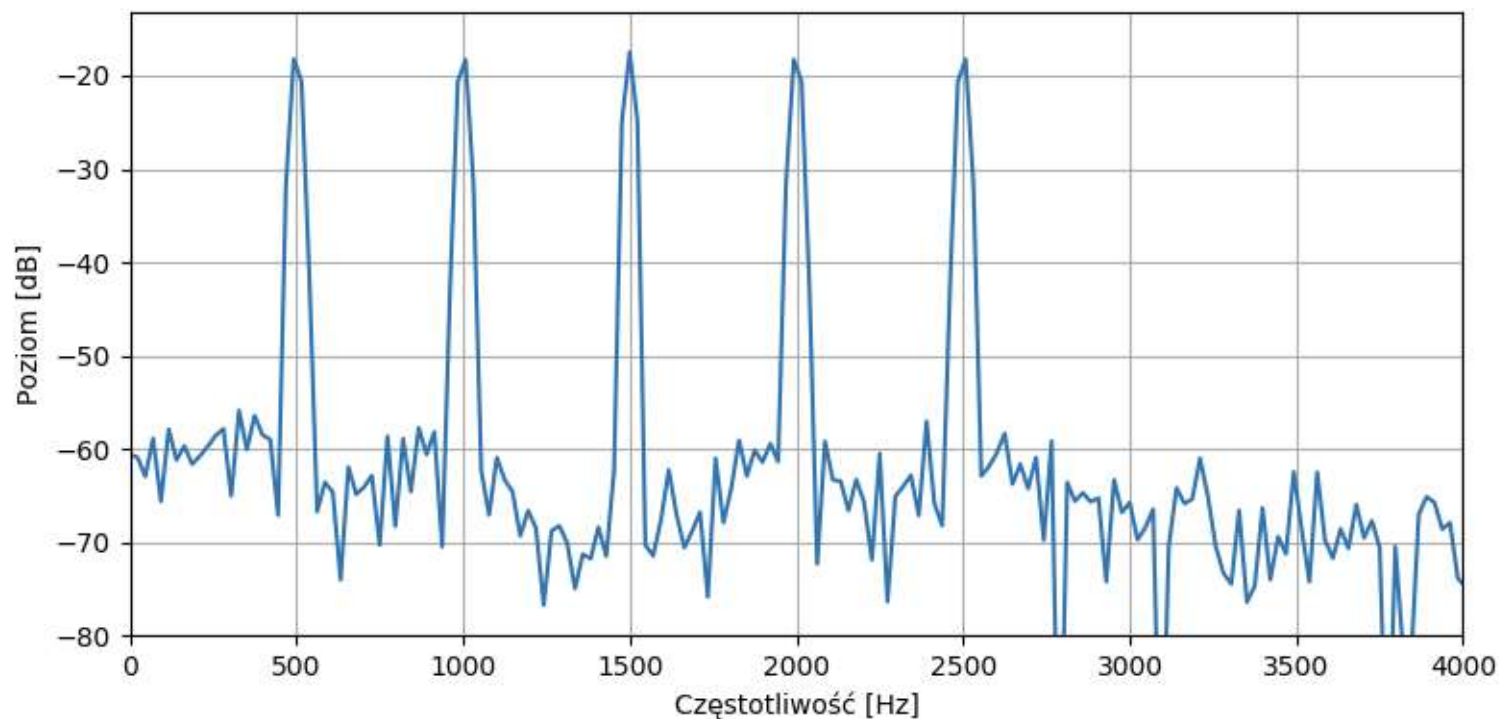
<https://docs.scipy.org/doc/scipy/reference/signal.html>

Wszystkie przykłady zakładają, że wcześniej wykonano instrukcje:

```
import numpy as np
import scipy.signal as sig
```

Częstotliwość próbkowania = 48 kHz

Sygnał testowy – suma pięciu „sinusów” + szum,
częstotliwości: 500, 1000, 1500, 2000, 2500 Hz



Projekt 1. Chcemy usunąć pierwsze dwa sinusy.

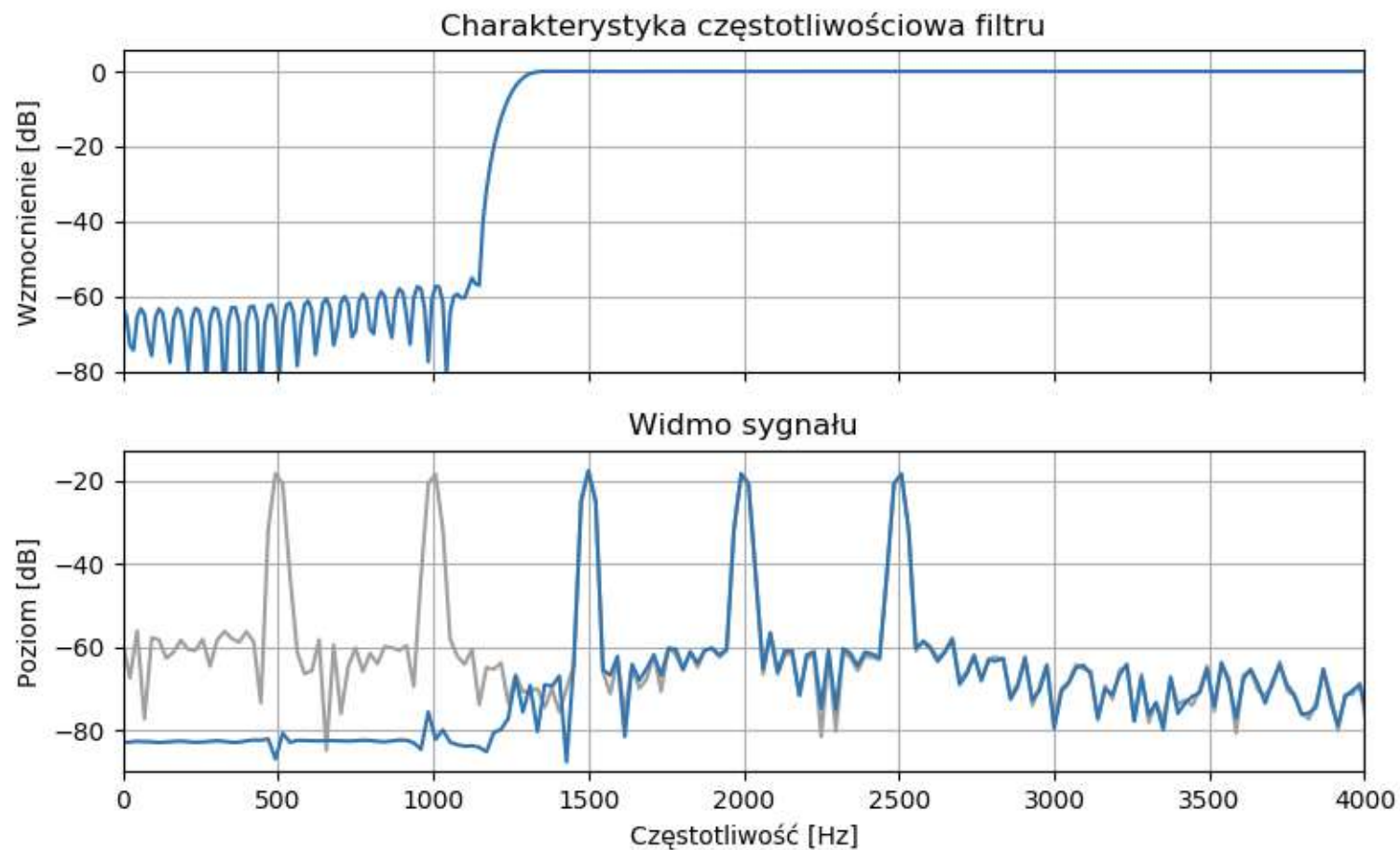
- filtr górnoprzepustowy
- częstotliwość graniczna = 1250 Hz
- zakładamy szerokość pasma przejściowego = 200 Hz
- stosujemy okno Hamminga – szacunkowa minimalna długość = $3,3 * (48000 / 200) = 792$
- wybieramy długość $N = 801$ (z zapasem)
(pamiętajmy: dla GP musi być nieparzysta)

Stosujemy procedurę okienkowania, z obliczaniem odpowiedzi impulsowej w dziedzinie czasu.

```
h1 = sig.firwin(801, 1250, window='hamming',  
               pass_zero='highpass', fs=48000)
```

- długość filtru (801)
- częstotliwość graniczna (1250)
- typ okna ('hamming')
- typ charakterystyki ('highpass' = GP)
- częstotliwość próbkowania (48000)

Projekt 1 – wynik



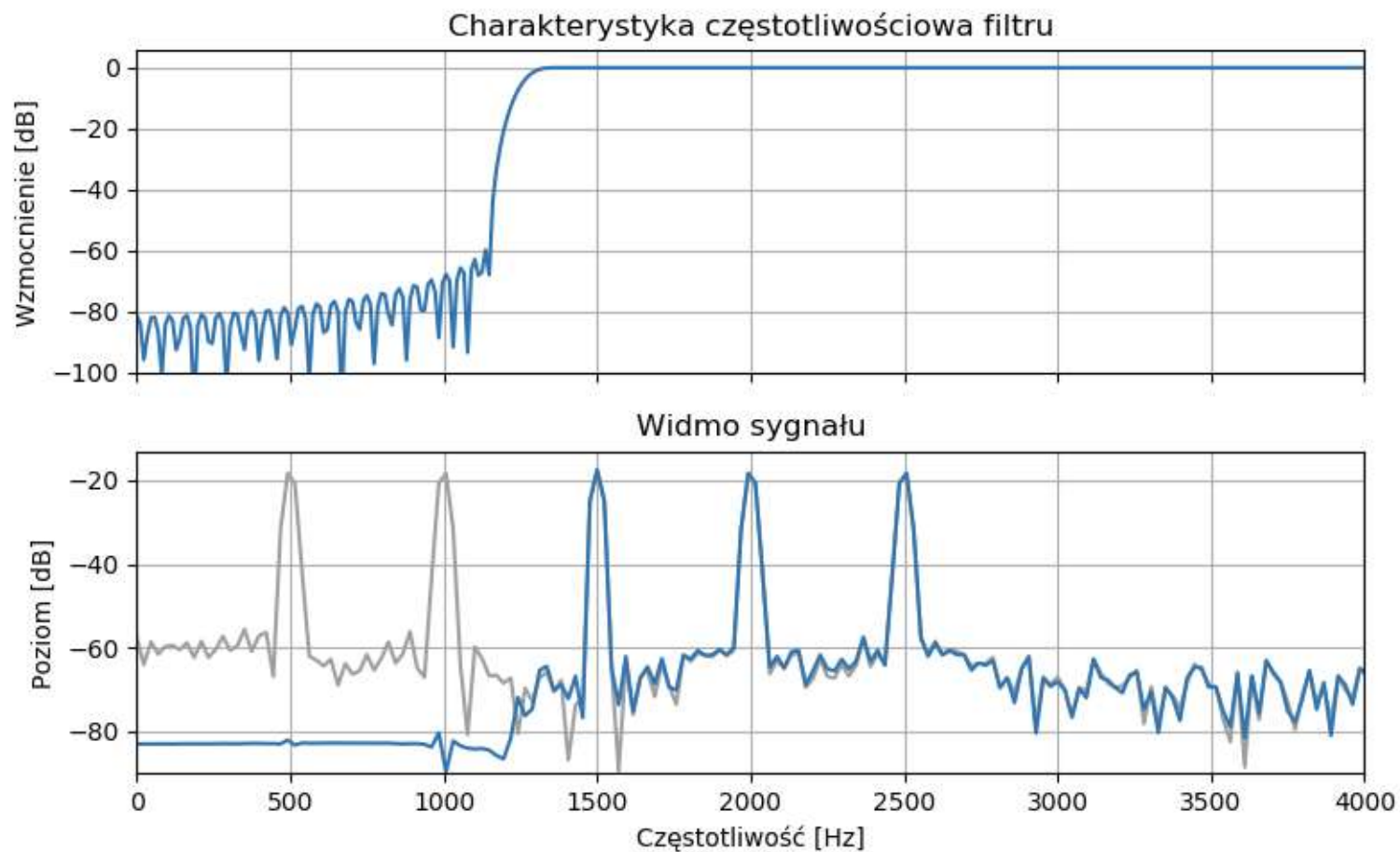
Projekt 1a – to samo, ale wykorzystamy okno Kaisera.

- Szerokość pasma przejściowego: 200 Hz, jak poprzednio.
- Minimalne tłumienie: 60 dB.

```
# musimy znormalizować szerokość pasma
nk, beta = sig.kaiserord(60, 2 * 200 / 48000)
if nk % 2 == 0:
    nk += 1 # dla GP długość musi być nieparzysta
# nk=873, beta=5.65326

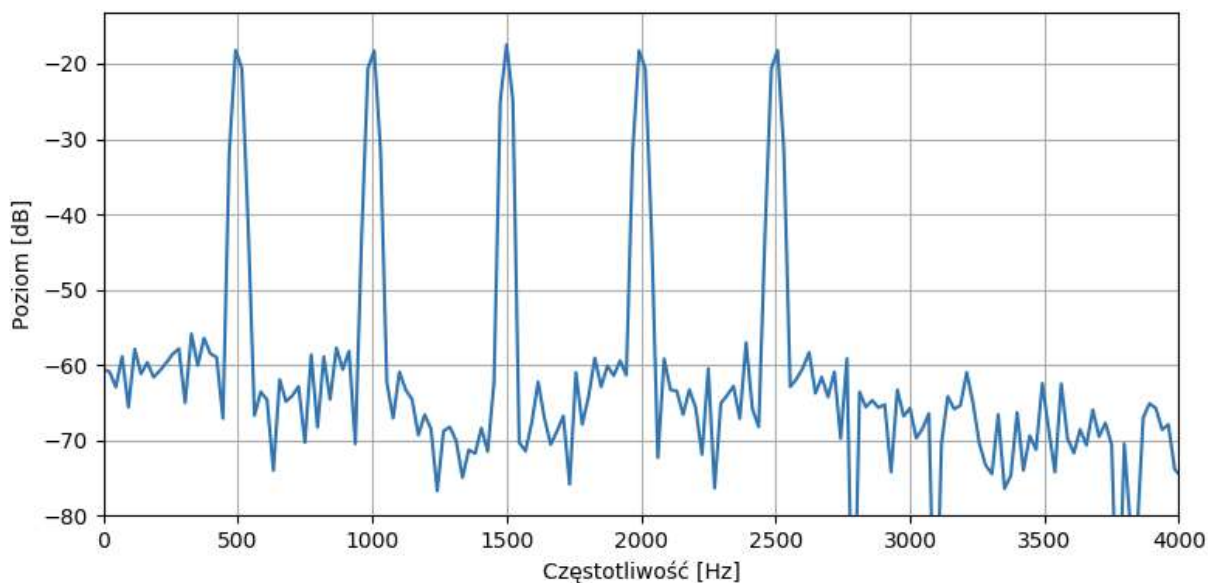
h1a = sig.firwin(nk, 1250, window=('kaiser', beta),
                 pass_zero='highpass', fs=48000)
```

Projekt 1a - wynik



Projekt 2 – zostawiamy tylko drugi i trzeci sinus.

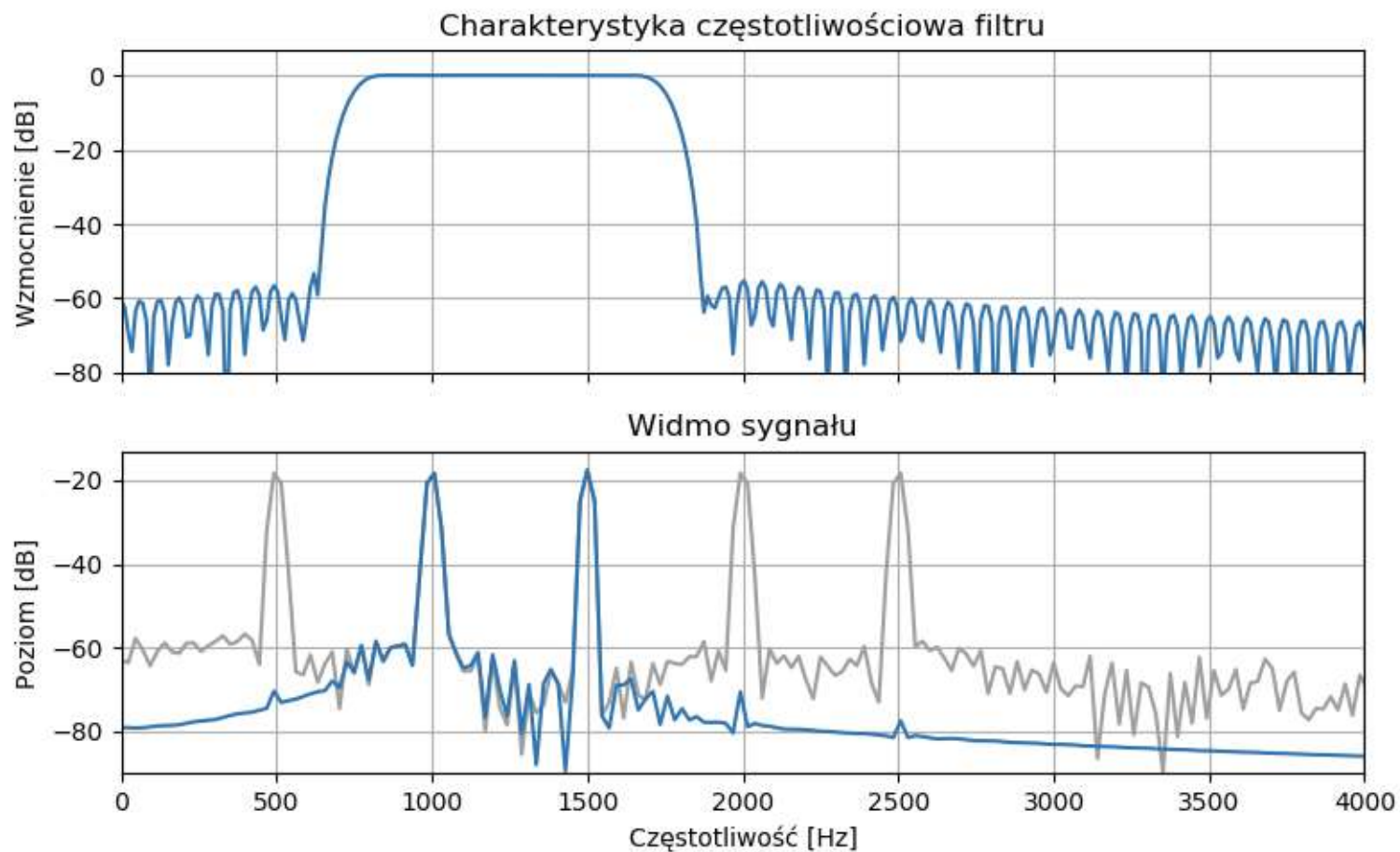
- Filtr pasmowo-przepustowy
- Częstotliwości graniczne: 750 Hz, 1750 Hz
- Szerokość pasma przejściowego: 20 Hz
- Stosujemy to samo okno: Hamminga, $N = 801$



- Tym razem zastosujemy metodę okienkowania z obliczaniem w dziedzinie częstotliwości.
- Zakładamy idealną charakterystykę widmową:
 - częstotliwości graniczne pasm **przepustowych** i **zaporowych**:
0, 730, 750, 1750, 1770, 24000
 - wzmocnienia filtru na tych częstotliwościach:
0, 0, 1, 1, 0, 0

```
h2 = sig.firwin2(801, (0, 730, 750, 1750, 1770, 24000),  
                (0, 0, 1, 1, 0, 0),  
                window='hamming', fs=48000)
```

Projekt 2 – wynik



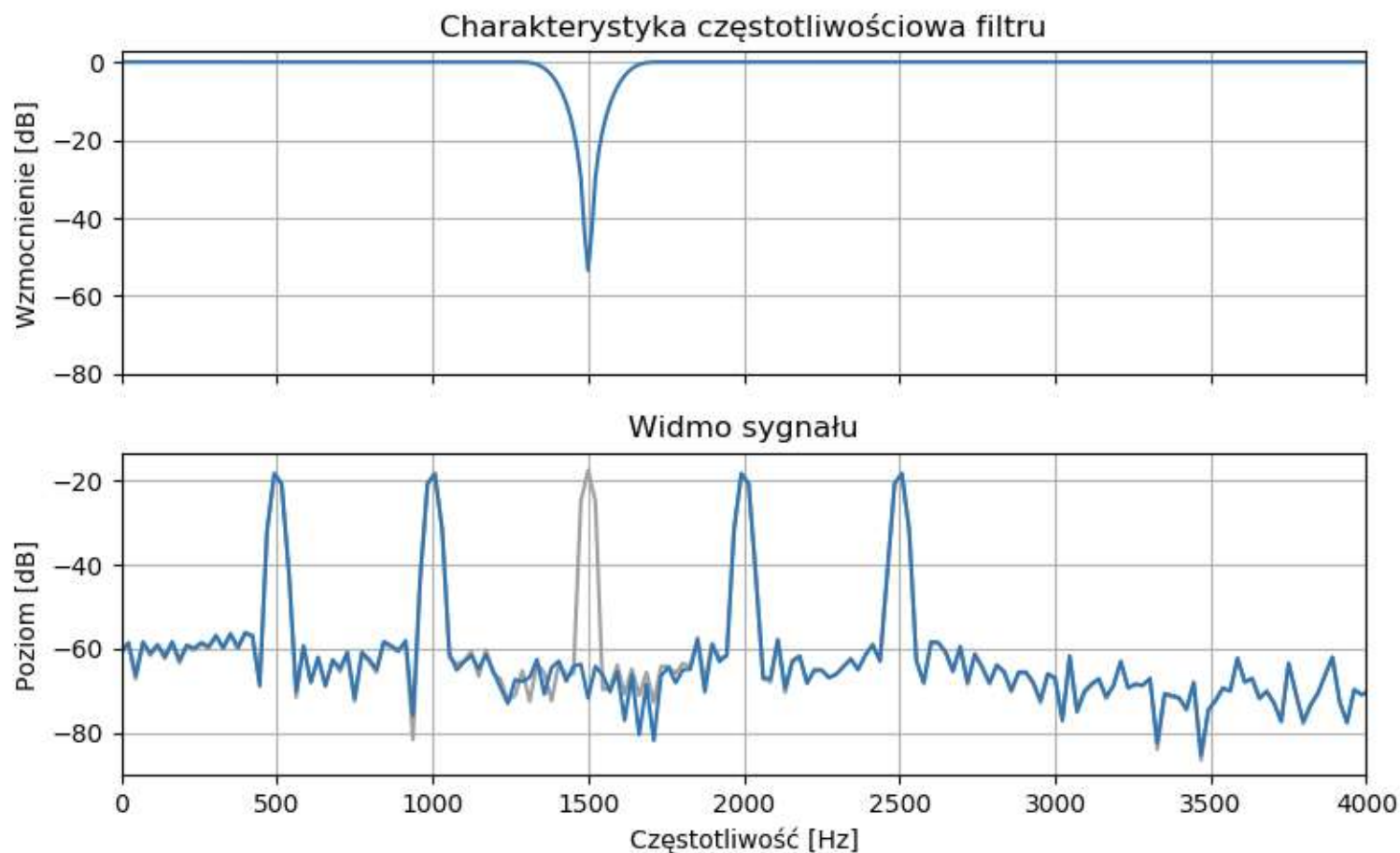
Projekt 3 – usunięcie środkowego sinusa

- Filtr pasmowo-zaporowy typu *notch*
- Cz. graniczna 1500 Hz, p. przejściowe 200 Hz.
- Zastosujemy metodę najmniejszych kwadratów.
- Sposób definiowania charakterystyki
– taki sam jak w projekcie 2.

```
h3 = sig.firls(801, (0, 1290, 1490, 1510, 1710, 24000),  
               (1, 1, 0, 0, 1, 1),  
               fs=48000)
```

Projekt 3 – wynik

Jeżeli chcemy większe tłumienie, musimy zwiększyć długość filtru i/lub poszerzyć pasmo przejściowe.

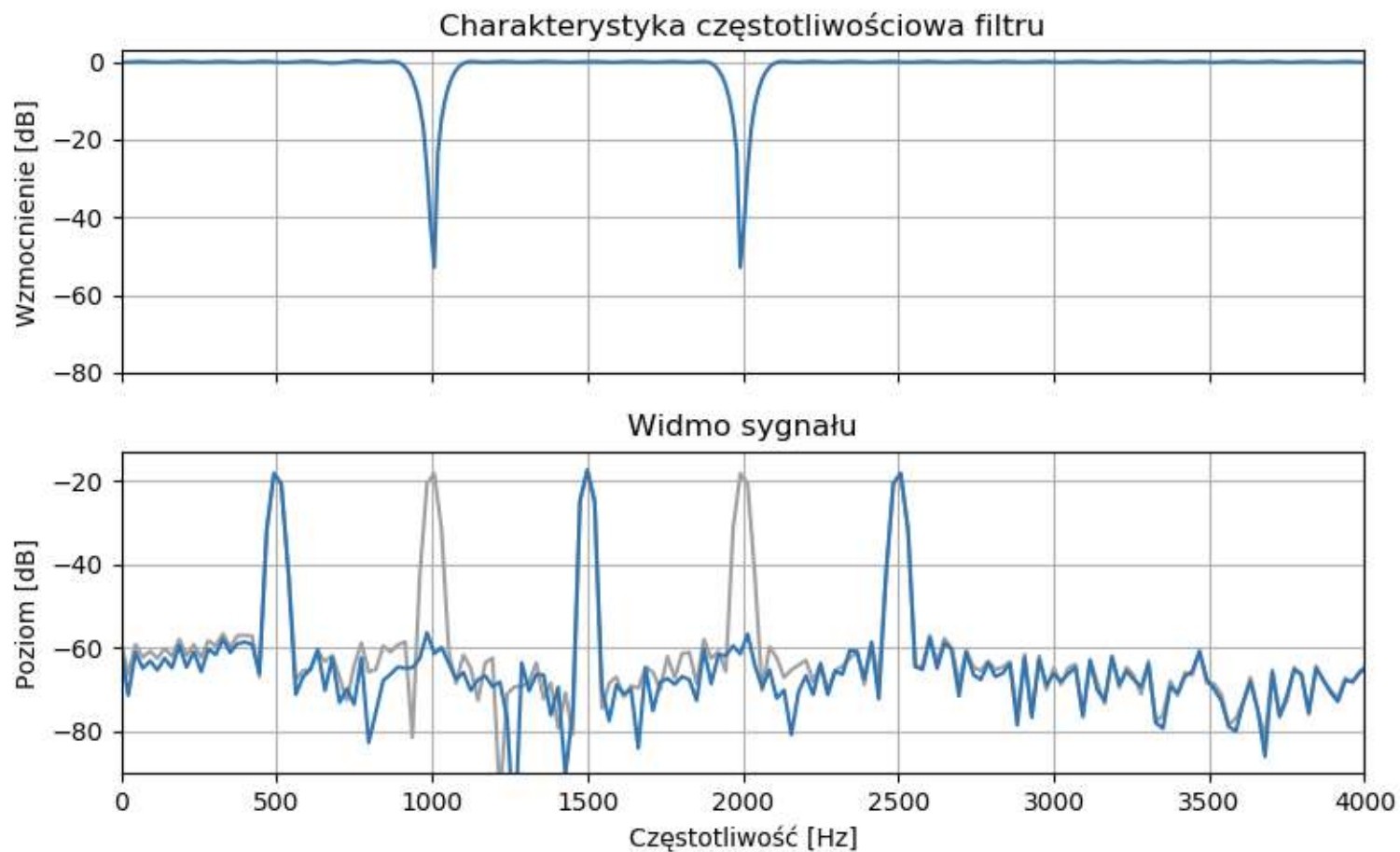


Projekt 4 – usuwamy drugi i czwarty sinus, korzystając z metody Parksa-McClellana („Remez”)

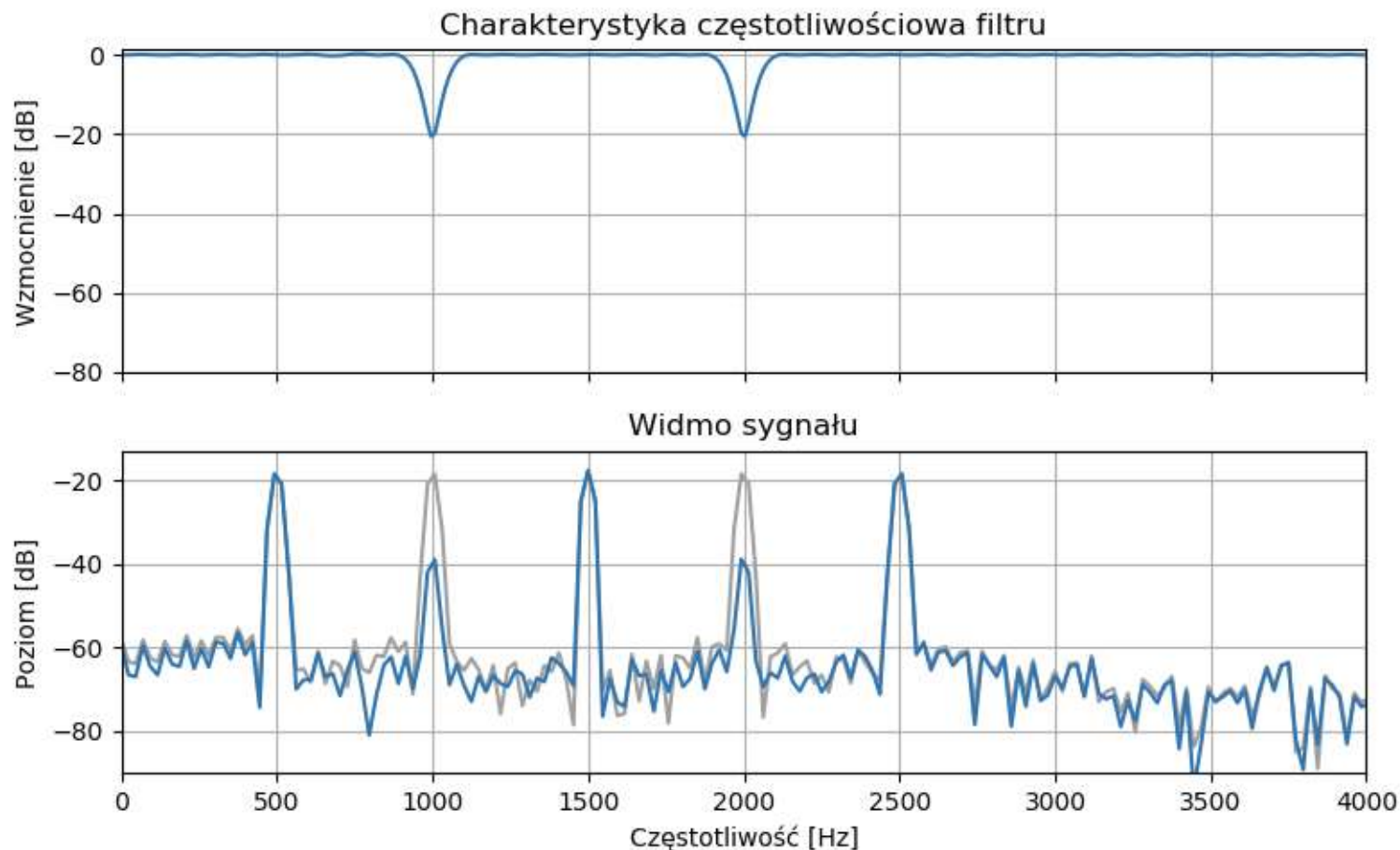
- Projektujemy filtr podając częstotliwości graniczne pasm oraz wzmocnienia – jedno na każde pasmo.
- Redukujemy pasma przejściowe do 100 Hz
(dla 200 Hz algorytm nie uzyskał zbieżności).

```
h4 = sig.remez(801,  
    (0, 890, 990, 1010, 1110, 1890, 1990, 2010, 2110, 24000),  
    (    1,          0,          1,          0,          1),  
    fs=48000)
```


Projekt 4 - wynik



Projekt 4a: nie usuwamy sinusów, ale tłumimy je o 20 dB (współczynniki: $[1, 0.1, 1, 0.1, 1]$)



Operacja filtracji FIR filtrem o transmitancji H :

$$X(f) \longrightarrow \boxed{H(f)} \longrightarrow Y(f) \quad Y(f) = H(f) \cdot X(f)$$

Filtr FIR jest układem LTI:

- jest liniowy,
- odpowiedź impulsowa jest stała w czasie.

A zatem w dziedzinie czasu:

$$y[n] = h[n] * x[n]$$

A więc: filtracja FIR jest operacją splotu liniowego współczynników filtru z próbkami sygnału.

Sytuacja praktyczna: filtr operuje na sygnale ciągłym, teoretycznie nieskończonym (np. dźwięk z mikrofonu).

Możliwe podejścia:

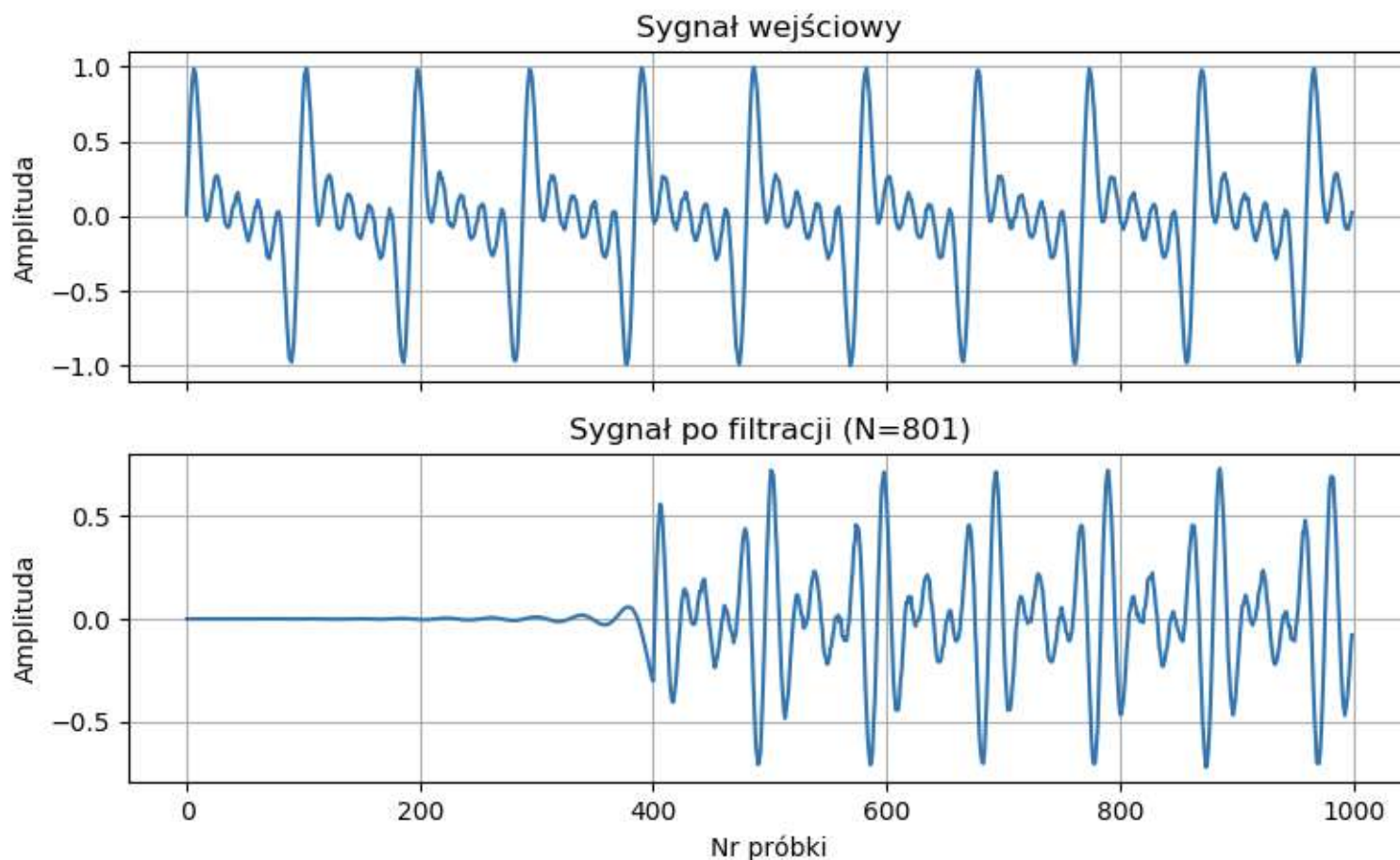
- przetwarzanie sygnału **próbka po próbce**
 - minimalizacja opóźnień
- przetwarzanie **blokowe** – zebranie bloku próbek, przetworzenie całego bloku naraz
 - zwiększa opóźnienie (np. blok 1024 próbek dla $f_s = 48$ kHz: opóźnienie 21,3 ms),
 - ale daje możliwość bardziej wydajnej filtracji.

O czym trzeba pamiętać:

- włączamy się z filtracją w pewnym momencie,
- wcześniej istniały pewne próbki sygnału (nie mamy ich),
- filtr zakłada, że wcześniej w sygnale były zera (co zazwyczaj nie jest prawdą),
- zatem pierwsze $(N-1)/2$ wyniki filtracji (dla filtru liniowofazowego) są błędne (nie ma wystarczających danych),
- tworzą one **stan nieustalony** (*transient*) filtru,
- wyniki te powinniśmy odrzucić.

Ilustracja stanu nieustalonego filtra:

filtr GP, długość 801 – stan nieustalony trwa 400 próbek



Skoro filtracja FIR jest splotem, to możemy wykonać ją w dziedzinie częstotliwości:

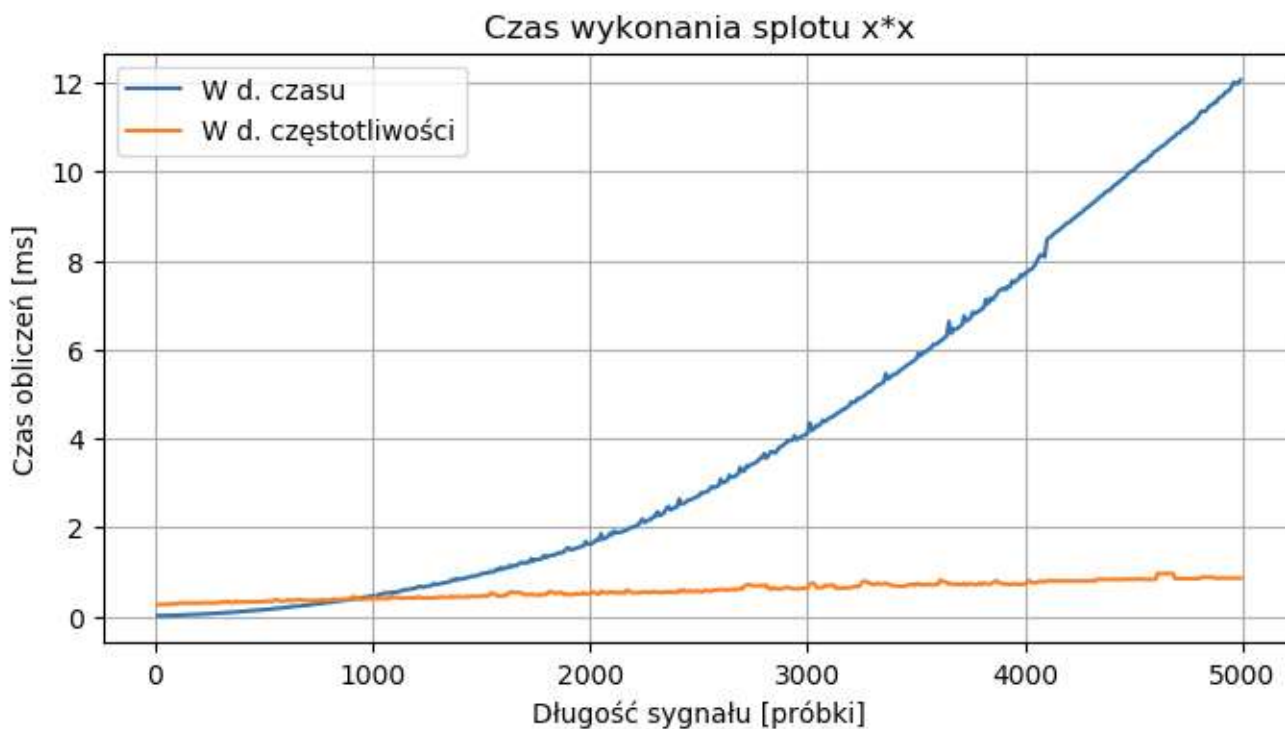
- obliczyć transmitancję filtra (FFT na współczynnikach),
- obliczyć widmo bloku próbek (FFT),
- przemnożyć je przez siebie,
- wykonać odwrotne przekształcenie Fouriera (IFFT),
- mamy wynik splotu / filtracji.

- Jeżeli po prostu przemnożymy transformaty, otrzymamy wynik splotu **kołowego** – wyniki zawiną się i dodadzą się do siebie. Nie o to chodzi.
- Filtracja FIR jest splotem **liniowym**.
- Splot liniowy sygnałów o długościach L , M daje wynik o długości $L+M-1$.
- Aby wykonać splot liniowy, musimy **uzupełnić zerami** blok próbek i blok współczynników do długości co najmniej **$L+M-1$** , przed obliczeniem transformaty.

Po co robić splot „na około”? Czy tak będzie szybciej?

Czas splotu sygnału x z samym sobą, obie metody

(obliczone za pomocą `scipy.signal.choose_conv_method`)



- Ze wzrostem długości sygnału, czas wykonania splotu rośnie w przybliżeniu:
 - wykładniczo dla splotu w dziedzinie czasu,
 - liniowo dla splotu w dziedzinie częstotliwości.
- Jedynie dla małych długości (<500) spłot w dziedzinie czasu jest wyraźnie szybszy.
- Dla większych długości (>1000): spłot w dziedzinie częstotliwości jest szybszy, różnica rośnie ze wzrostem długości sygnału.
- Spłot w dziedzinie częstotliwości nazywa się **szybkim splotem** (*fast convolution, FFT convolution*)

Uwagi o filtracji FFT w dziedzinie częstotliwości

- Transformatę współczynników filtru wystarczy obliczyć tylko jeden raz – nie zmienia się.
- Czas wykonania szybkiego splotu zależy od wydajności algorytmu FFT:
 - niektóre implementacje wymagają, aby długość transformaty była potęgą dwójki,
 - w większości współczesnych implementacji FFT zaleca się, aby długość transformaty była iloczynem niskich liczb pierwszych (2, 3, 5, 7).

Przykład praktyczny:

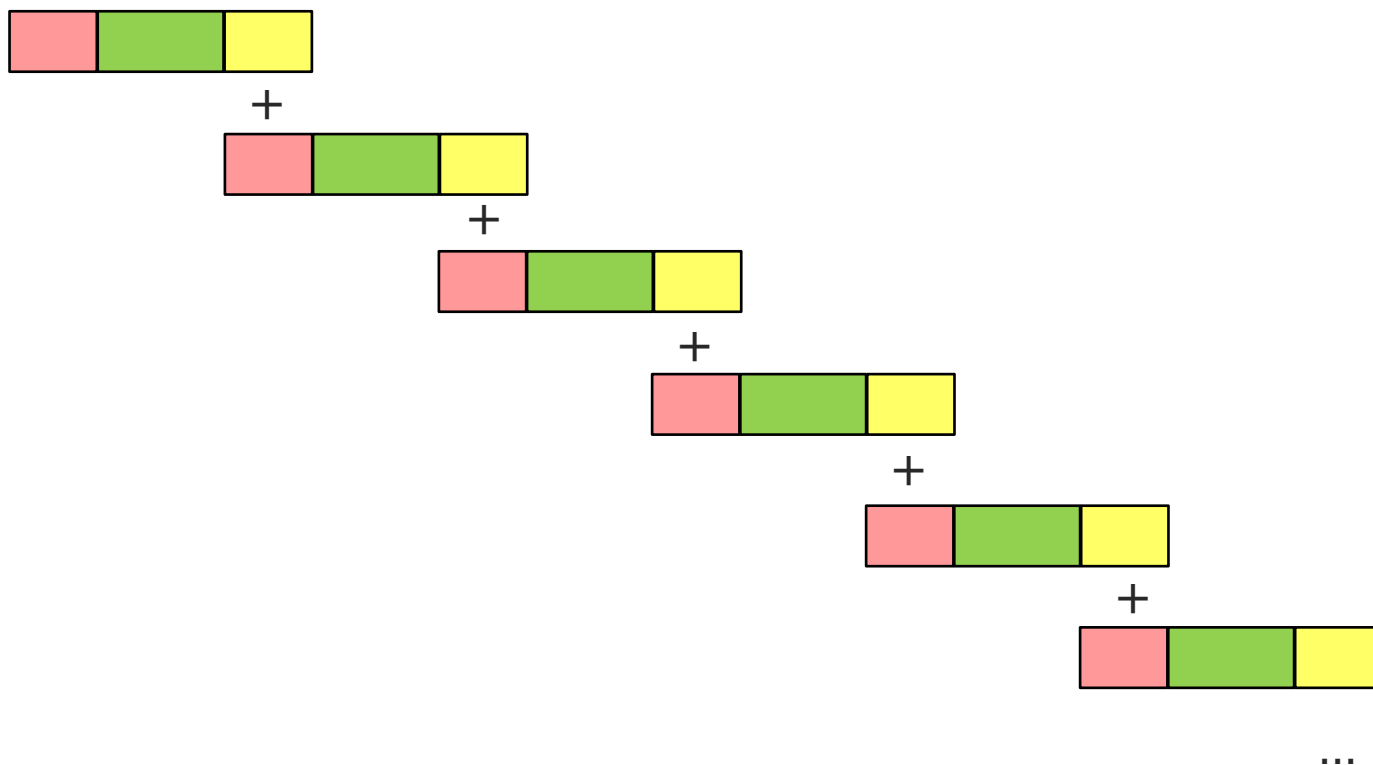
- przetwarzamy bloki próbek o długości $L=1024$
- filtr ma długość M , rozmiar splatanych bloków musi mieć długość $N = L+M-1$
- przyjmijmy $N = 2048 = 2^{11}$ (dla wygody FFT)
- wtedy $M = N-L+1 = 1025$
(długość filtru jest nieparzysta – to dobrze)
- każdy blok 1024 próbek uzupełniamy 1024 zerami, liczymy FFT, mnożymy przez transformatę współczynników, liczymy IFFT
- dostajemy 2048 próbek wyniku i ... co dalej?

- Na wejściu mamy L próbek, na wyjściu musi być tyle samo próbek (L).
- Dostajemy $L+M-1$ wyników. Co zrobić z nadmiarowymi wynikami? Odrzucić?
- Załóżmy, że przetwarzamy któryś z kolei blok.



- Pierwsze $M-1$ wyniki (czerwone) są błędne – stan nieustalony filtru.
- Kolejne $L-M+1$ wyniki (zielone) są poprawne.
- Ostatnie $M-1$ wyniki (żółte) – co z nimi zrobić?

Rozwiązanie problemu jest proste: do pierwszych $M-1$ próbek wyniku (stan nieustalony) dodajemy „nadmiarowe” $M-1$ próbki z poprzedniego bloku, pierwsze L próbki wyniku wysyłamy na wyjście.



Schemat postępowania:

1. Pobierz blok L próbek z wejścia, uzupełnij zerami do długości $N \geq L+M-1$, oblicz FFT.
2. Przemnóż przez transformatę współczynników filtru.
3. Oblicz IFFT wyniku mnożenia.
4. Do pierwszych $M-1$ wartości wyniku dodaj zawartość bufora.
5. Wyślij pierwsze L wartości wyniku na wyjście.
6. Zapisz pozostałe $M-1$ wartości do bufora.
7. GOTO 1.

- Opisana metoda splotu blokowego sygnału nosi nazwę *overlap-add* (OLA)
 - zakładkowanie z dodawaniem.
- Umożliwia ona filtrację ciągłych bloków próbek, w dziedzinie częstotliwości lub czasu.
- W porównaniu z filtracją „próbka po próbce”:
 - skracamy czas obliczeń (gdy N jest duże),
 - zwiększamy opóźnienie przetwarzania
 - musimy poczekać aż uzbiera się cały blok próbek, dopiero wtedy je przetwarzamy.

Istnieje alternatywna metoda: *overlap-save* (OLS)

- bloki próbek wejściowych pobieramy „na zakładkę”
– przesuwamy okno o L próbek, $M-1$ próbek powtarza się w kolejnym bloku,
- obliczamy splot tak jak w OLA,
- odrzucamy cały stan nieustalony ($M-1$ próbek),
- resztę (L próbek) wysyłamy na wyjście.

Złożoność obliczeniowa metod OLA i OLS jest taka sama. Wybieramy metodę prostszą do implementacji w danym systemie.

PODSUMOWANIE – zalety filtrów FIR:

- bardzo prosty algorytm (splot liniowy),
- względnie łatwe projektowanie,
- typowe filtry mają liniową fazę, a więc stałe opóźnienie dla wszystkich częstotliwości, nie wprowadzają zniekształceń fazowych,
- zawsze są stabilne: jeżeli wyłączymy sygnał wejściowy, po pewnym czasie sygnał na wyjściu również stanie się zerowy,
- łatwe do implementacji w typowych systemach DSP.

PODSUMOWANIE – wady filtrów FIR:

- musimy stosować filtry o dużej długości aby zapewnić dobre tłumienie i wąskie pasmo przejściowe,
- dużo obliczeń dla „długich” filtrów (zalecane stosowanie szybkiego splotu),
- duża zajętość pamięci (współczynniki, bufor próbek),
- opóźnianie sygnału przez filtry – zwiększa się z długością filtru, może być odczuwalne, np. przy przetwarzaniu dźwięku.