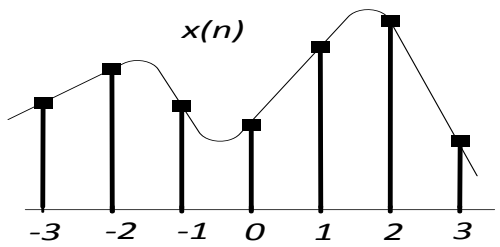
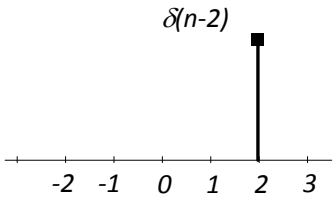
Cyfrowe przetwarzanie sygnałów – filtry cyfrowe



Rys. 1. Sygnał cyfrowy jest ciągiem impulsów z których każdy jest opisany przez swoje położenie i przez wartość



Rys. 2. Impuls na pozycji 2

Do określania pozycji impulsu wykorzystujemy funkcję Diraca w postaci

$$\delta(n-k)=\begin{cases}1 & n=k\\ 0 & n\neq k\end{cases}$$

Tak więc impuls opisany jest jako

$$x(k)=x(n)\delta(n-k)$$

Gdzie pierwszy składni oznacza wartość impulsu, a drugi jego położenie. Cały sygnał będzie więc opisany jak

$$x(n)=\sum_{k=-\infty}^{\infty} x(k)\ \delta(n-k)$$

Układy LTI , Odpowiedź impulsowa

Będziemy zajmowali się układami LTI (linear, time invariant – liniowy czasowo niezależny)

Liniowość oznacza że można stosować zasadę superpozycji, a więc

$$f(x_1 + x_2) = y_1 + y_2$$

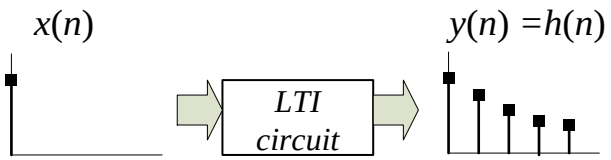
$$f(cx_1) = cy_1$$

Układ jest niezależny w czasie jeśli przesunięcie w czasie sygnału wejściowego spowoduje równoważne przesunięcie w czasie sygnału wyjściowego. A więc jeśli

$$x(n)=x_1(n-n_o)$$

To $y(n)=y_1(n-n_o)$.

Parametrem opisującym jednoznacznie układ LTI jest jego odpowiedź impulsowa h(n), a więc zgodnie z nazwą jest to odpowiedz układu na pojedynczy impuls jednostkowy.



Rys. 3. Odpowiedź impulsowa układu piąteg rzędu

Odpowiedź impulsowa opisywana jest przez odpowiednie współczynniki a więc h1, h2, h3 itd./

Splot cyfrowy

Wyobraźmy sobie że chcemy wyznaczyć funkcję F sygnału to znaczy y (n) = F[x(n)] a więc

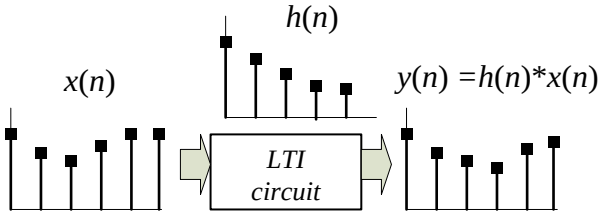
$$y(n) = F\left\{\sum_{k=-\infty}^{\infty} x(k)\delta(n-k)\right\} = \sum_{k=-\infty}^{\infty} x(k)F\{\delta(n-k)\}$$

Co możemy zapisać jako

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

Jest to równanie splotu (convolution) a więc równanie pozwalające określić sygnał wyjściowy y(n) układu opisanego przez odpowiedź impulsową h(n).

Możliwe jest tez dokonanie operacji odwrotnej rozplotu (deconvolution) czyli określić sygnał wejściowy na podstawie znajomości sygnały wyjściowego i odpowiedzi impulsowej,



Rys. 4. Operacja splotu

Równanie splotu można zapisać w dwóch równoważnych wariantach

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = x(n) * h(n)$$

lub

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) = h(n) * x(n)$$

Algorytm realizacji splotu

Algorytm realizacji splotu jest relatywnie prosty – składa się z trzech operacji: przesunięcia, mnożenia i dodawania (rys. 5 i 6).

Na wstępie odwracamy funkcję jednostkową h(n) tak żeby h1 było na początku.

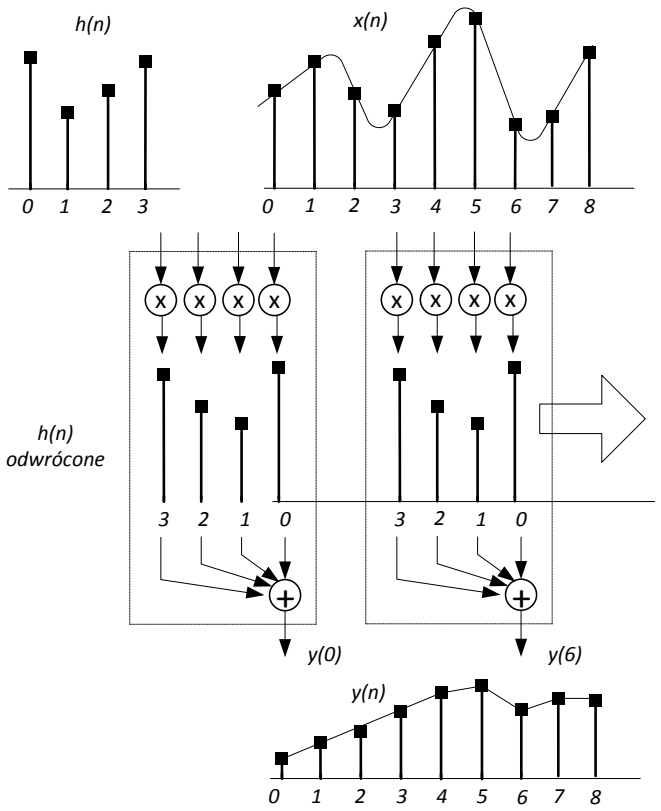
Następnie przesuwamy funkcję h(n) tak aby jej początek trafił na punkt x(n) którego odpowiedź y(n) chcemy wyznaczyć (na rysunku 5 jest to x(6).

Drugi krok to wymnażamy sąsiadów a więc x6 przez h1, h5 przez h2 itd.

Wyniki mnożenia dodajemy i zapisujemy jako odpowiedź y(6).

Następnie przesuwamy h(n) na pozycję 7 i powtarzamy czynności (lub h(n) stoi a przesuwają się danej x(n) – drugi wariant równania splotu.

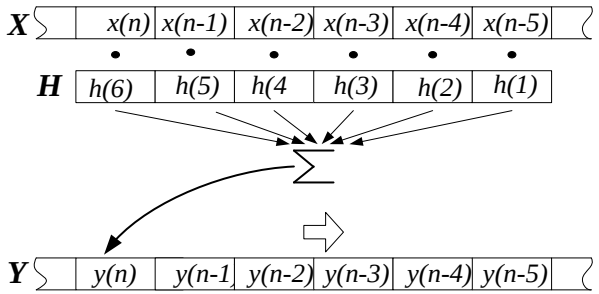
Operacja splotu jest operacją on-line to znaczy wymnożenie i dodawanie musi być zrealizowane nim nadejdzie nowy impuls x(n).



Rys.5. Algorytm realizacji splotu

Transformata Z

Rys.6. Przedstawiony na rys. 5 algorytm można inaczej przedstawić jako operacje na rejestrach – bieżącym x(n), stałtm h(n) i bieżącym y(n)



A więc

$$y(n) = \sum h(k)\, x(n-k) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) \ldots$$

Operację przesunięcia dobrze ilustruje transformata Z gdzie

$$Z\{x(n-1)\} = z^{-1} X(z)$$

Lub ogólnie

$$Z\{x(n-m)\} = z^{-m} X(z)$$

A więc nasze równanie splotu

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) = h(n) * x(n)$$

$$h(n) = \sum_{k=0}^{\infty} h(k)\delta(n-k)$$

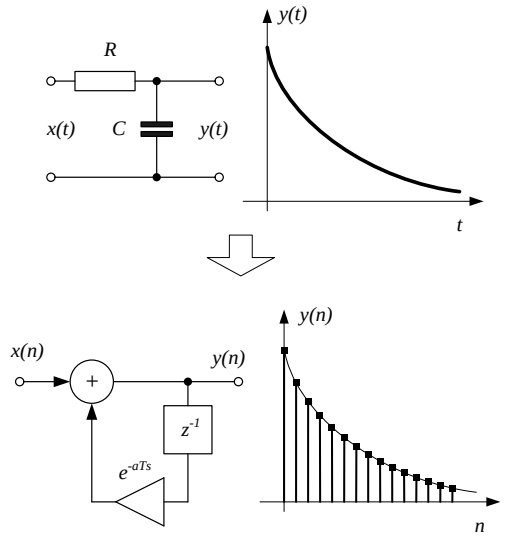
Można zapisać w prostszej postaci:

$$Y(z) = H(z)X(z)$$

$$H(z) = \sum_{k=0}^{\infty} h_k z^{-k}$$

Zasada działania filtru cyfrowego

Rys. 7. Cyfrowy odpowiednik filtru analogowego RC



Widzimy więc że odpowiednikiem filtru RC jest operacja przesunięcia z^{-1}, wymnożenia i dodania a więc jest to opisana wcześniej operacja splotu.

A więc operację filtrowania opisuje równanie splotu

$$Y(k) = H(k) \cdot X(k)$$

Gdzie $$H(z) = \frac{Y(z)}{X(z)} = \sum a(k) z^{-k}$$

Lub $$H(z) = \sum_{k=0}^{N-1} a(k) z^{-k} = a_0 + a_1 z^{-1} + a_2 z^{-2} + ..$$

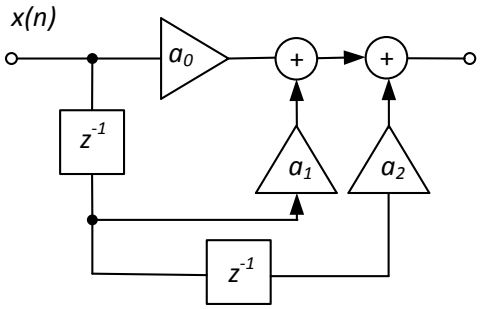
Lub

$$y(n) = \sum_{k=0}^{N-1} a(k) x(n-k) = a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) + ..$$

Gdzie a0, a1, a2 itd. Współczynniki filtru.

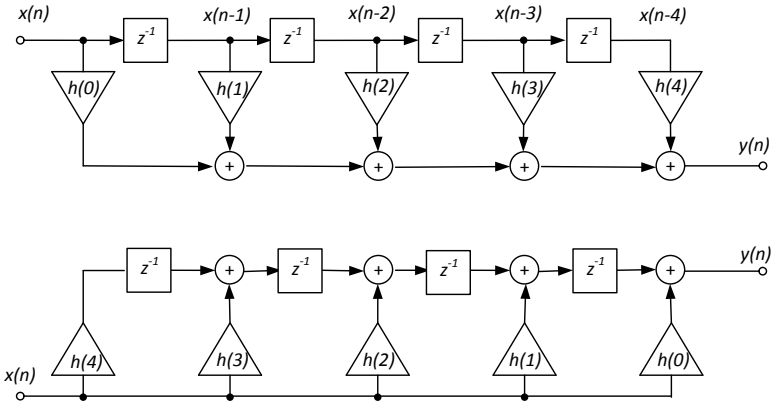
Filtry SOI (FIR)

Rys. 8. Zwielokrotniając operację przedstawioną na rys. 7 otrzymujemy filtr wyższego rzędu, tu filtr drugiego rzędu na rys. 9 filtr czwartego rzędu.



Przedstawiony na rys. 8, 9 filtr nosi nazwę: skończona odpowiedź impulsowa SOI (finite impulse response FIR)

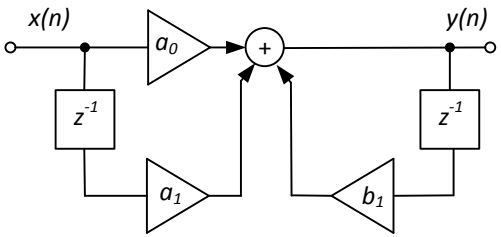
Rys. 9. Filtr SOI czwartego rzędu



Filtry NOI (IIR)

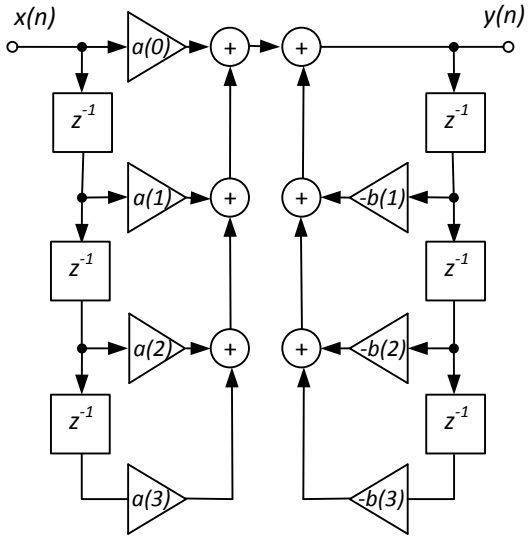
Rys.10. Obok przedstawionego wcześniej filtru SOI stosowane są też filtry NOI (nieskończona odpowiedź impulsowa luf infinite impulse response IIR).

W filtrach tych sygnał pobierany jest nie tylko z wejścia ale i z wyjścia:



Filtry te mogą być bardziej skuteczne niż filtry SOI ale kosztem możliwości wzbudzenia się (ponieważ istnieje tu sprzężenie zwrotne) stąd nazwa.

Rys. 11. Filtr NOI czwartego rzędu - mamy jakby dwa filtry SOI z których jeden pobiera sygnał z wejścia, drugi z wyjścia.



Stabilność filtru NOI

Przedstawione wcześniej równanie na odpowiedź impulsową zmienia się na

$$H(z)=\frac{Y(z)}{X(z)}=\frac{\sum_{k=0}^{N-1}a(k)z^{-k}}{1+\sum_{k=1}^{M}b(k)z^{-k}}$$

lub

$$H(z)=\frac{Y(z)}{X(z)}=\frac{a_1z^{-1}+a_2z^{-2}+a_3z^{-3}+\ldots}{1+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}+\ldots}$$

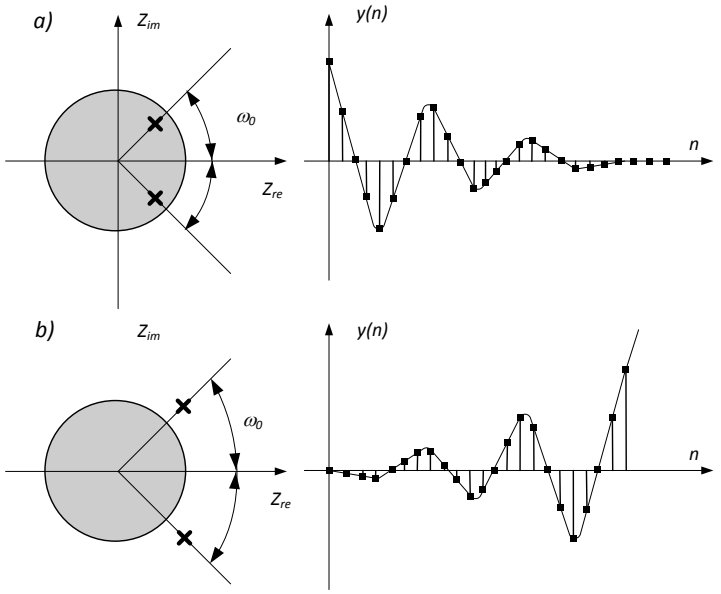
Pojawia się więc mianownik który może przyjmować wartości zerowe co oznacza wzbudzanie się układu.

Powyższe równanie można zapisać w postaci

$$H(z)=\frac{(z-z_1)(z-z_z)(z-z_3)\ldots}{(z-p_1)(z-p_2)(z-p_3)\ldots}$$

Gdzie p1, p2, p3 oznaczają bieguny funkcji kiedy przyjmuje ona zero

Rys. 12 W przypadku filtru NOI konieczna jest więc sprawdzenie warunków stabilności. Jeśli przedstawimy H(z) na płaszczyźnie zespolonej to to układ jest stabilny jeśli bieguny znajdują się wewnątrz koła jednostkowego -



Przykład projektu filtru

Rozpatrzmy przykład filtru który z sygnału 50 mV i 10 Hz usuwa zakłócenia 15 mV i 50 Hz.

W tym celu obliczamy skrypt Matlaba o postaci:

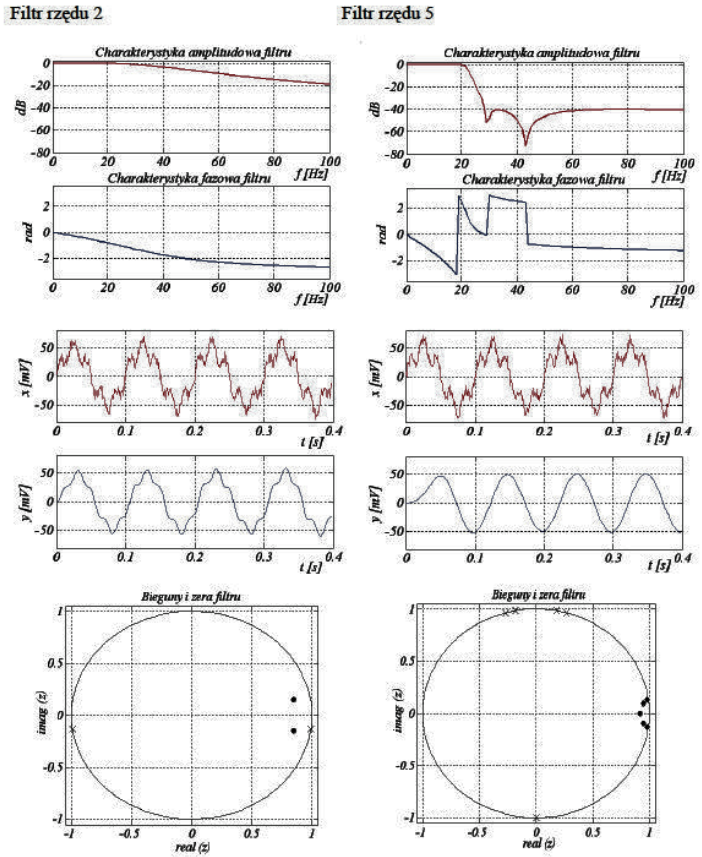
[B,A] = ellip (nf, rp, rs, fzn, low) – co oznacza – filtr eliptyczny dolnoprzepustowy „low” , rzędu nf, z zafalowaniem rs, o częstotliwości znamionowej fzn. Otrzymujemy odpowiednie współczynniki A i B:

A = [1,0000 -4,7646 9,1057 -8,7239 4,1898 -0,8069]

B = [0,0035 -0,0100 0,0066 0,0066 -0,0100 0,0035]

Charakterystyki obliczonego filtru drugiego i piątego rzędu przedstawia rys. 13

Rys. 13. Rezultat projektu filtru



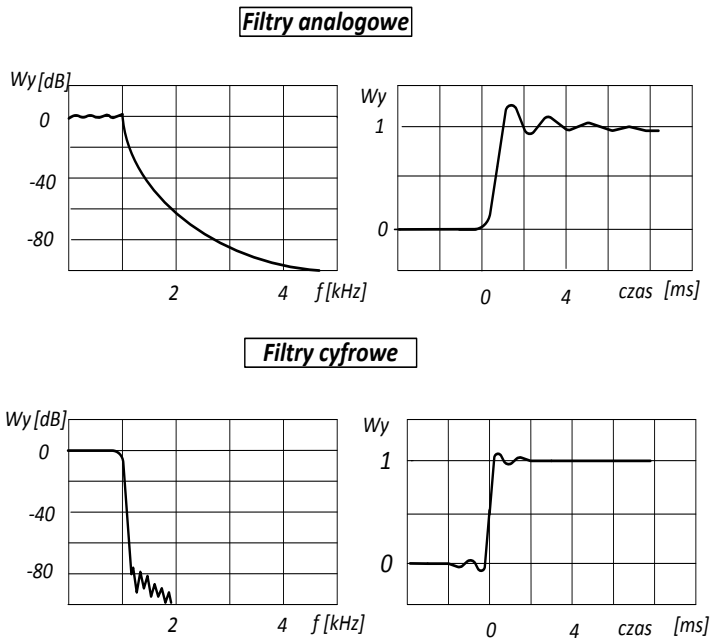
Porównanie filtru analogowego i cyfrowego

Filtr analogowy rzadko bywa wyższego rzędu niż 5. Ograniczeń tych nie ma filtr cyfrowy gdzie rząd filtru zależy tylko o szybkości procesora. W książce Smitha porównano charakterystyki częstotliwościowe i czasowe filtru analogowego i cyfrowego.

Jak wynika z rys. 14 filtr cyfrowy ma zdecydowanie bardziej stromą charakterystykę amplitudowa i krótsze czasy ustalania się wyniku.

Ale z kolei filtry cyfrowe mają ograniczenia częstotliwości pracy (częstotliwość próbkowania przetwornika A/D) jak i dynamiki (liczba bitów przetwornika A/D)

Rys. 14. Porównanie charakterystyk filtru.



Filtry adaptacyjne

Filtry cyfrowe w porównaniu z analogowymi mają jeszcze jedną istotną cechę - jest to w zasadzie tylko software.

Właściwości filtru można więc łatwo on-line zmieniać. To stworzyło możliwość projektowania filtrów adaptacyjnych które wg. Określonego algorytmu dobierają sobie współczynniki.

Jednym z takich filtrów jest np. przedstawiony na rys. 15 filtr Wienera używany powszechnie do usuwania szumu (algorytm adaptacyjny tak ustala współczynniki filtru) żeby usuwał on część sygnału skorelowanego z szumem).

Rys. 15. Przykład adaptacyjnego filtru Wienera