Wydział Elektryczny
ZTMAiPC

LABORATORIUM CYFROWEGO PRZETWARZANIA SYGNAŁÓW

Ćwiczenie 5

Wieloczęstotliwościowe przetwarzanie sygnałów – interpolacja i decymacja

1. Cel ćwiczenia

- Zapoznanie się z metodami i efektami cyfrowej zmiany częstotliwości próbkowania sygnałów dyskretnych.
- Przeprowadzenie dekompozycji i rekonstrukcji podpasmowej sygnału za pomocą zaprojektowanego banku filtrów pasmowoprzepustowych.

2. Podstawy teoretyczne

Przetwarzanie wieloczęstotliwościowe (*multirate processing*) jest ważnym obszarem cyfrowego przetwarzania sygnałów dyskretnych, który nie ma odpowiednika w dziedzinie analogowej. Podstawowe idee polegają na takiej konwersji sygnału, aby mógł on być przetwarzany z możliwie najmniejszą częstotliwością, dopasowanie częstotliwości do określonego standardu, pojemności kanału transmisyjnego lub mocy obliczeniowej procesora DSP. Operacja zmniejszania częstotliwości próbkowania sygnału (o czynnik całkowity) nazywana jest *decymacją*. Jeżeli potrzebne jest zwiększenie częstotliwości przetwarzania, np. w celu dokładniejszej reprezentacji sygnału, stosuje się operację odwrotną, czyli *interpolację*. Kombinacja interpolacji i decymacji w połączeniu z odpowiednią filtracją dolnoprzepustową umożliwia zmianę częstotliwości próbkowania o czynnik ułamkowy.

Oprócz podstawowych operacji interpolacji i decymacji przetwarzanie wieloczęstotliwościowe obejmuje bardziej zaawansowane tematy, takie jak synteza banków filtrów pasmowoprzepustowych do rozkładu i rekonstrukcji sygnałów czy rozkład sygnału za pomocą transformacji falkowych (*wavelets*).

2.1. Interpolacja sygnału

Proces interpolacji o czynnik U polega w zasadzie na estymacji lub rekonstrukcji U-1 wartości sygnału $x(n)$, próbkowanego z częstotliwością f_{sx}, pomiędzy istniejącymi próbkami. Pierwszym elementem interpolatora (rys. 1) jest ekspander, który pomiędzy każde sąsiednie próbki sygnału wejściowego wstawia U-1 próbek zerowych (*upsampling*). Sygnał wyjściowy z ekspandera

$$x_e(m) = \begin{cases} x(n)|_{n=m/U} = x(m/U) & \text{dla } m = 0, \pm U, \pm 2U, \ldots \\ 0 & \text{dla pozostalych } m \end{cases} \tag{5.1}$$

ma większą częstotliwość próbkowania $f_{sy}=U \cdot f_{sx}$. W dziedzinie częstotliwości zespolonej $X_e(z)=X(z^U)$.

Drugim elementem interpolatora jest filtr dolnoprzepustowy (LP) $H_I(z)$, który nadaje dodanym próbkom wartości niezerowe przez interpolację pomiędzy próbkami oryginalnymi generując na wyjściu sygnał interpolowany $y(m)$. Idealny filtr LP interpolatora ma charakterystykę widmową

$$H_I(e^{j\Omega}) = \begin{cases} U & \text{dla } 0 \le |\Omega| \le \dfrac{\pi}{U} \\ 0 & \text{dla pozostalych } \Omega \end{cases} . \tag{5.2}$$

i zerowe opóźnienie grupowe $\tau(\Omega)$. Wzmocnienie równe U zapewnia takie same amplitudy obwiedni sygnałów $x(n)$ i $y(m)$ (tylko co U-ta próbka sygnału wejściowego filtra jest niezerowa). Skala unormowanej pulsacji kątowej $\Omega=\Omega_y=\Omega_x/U$, ponieważ

$$\Omega_x = \frac{2\pi f_x}{f_{sx}}, \quad \Omega_y = \frac{2\pi f_y}{f_{sy}}. \tag{5.3}$$

Charakterystyka $H_I(e^{j\Omega})$ odpowiada nieskończonej i *nieprzyczynowej* (tzn. o wartościach niezerowych dla czasu $m<0$) charakterystyce impulsowej:

$$h_I(m) = \begin{cases} \dfrac{\sin(m\pi/U)}{m\pi/U} & \text{dla } m \neq 0 \\ 1 & \text{dla } m = 0 \end{cases}. \tag{5.4}$$

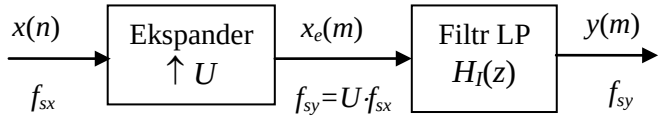
gdzie $\sin(\pi x)/(\pi x)=\mathrm{sinc}(x)$. Podane warunki mogą być w dobrym przybliżeniu spełnione tylko przez nieprzyczynowy filtr odpowiednio wysokiego rzędu. Filtr interpolatora, o paśmie przepustowym $\Omega_{pass}=\pi/U$, projektuje się zwykle określając parametr $0<\alpha\leq 1$ wypełnienia pasma Nyquista przez ograniczone widmo sygnału wejściowego. Mniejsza wartość α powoduje poszerzenie pasma przejściowego i zwiększenie tłumienia w paśmie zaporowym (dla określonego rzędu filtra), ale też powstanie w paśmie zaporowym regionów „don't care".

W dziedzinie czasu wyjście interpolatora jest splotem:

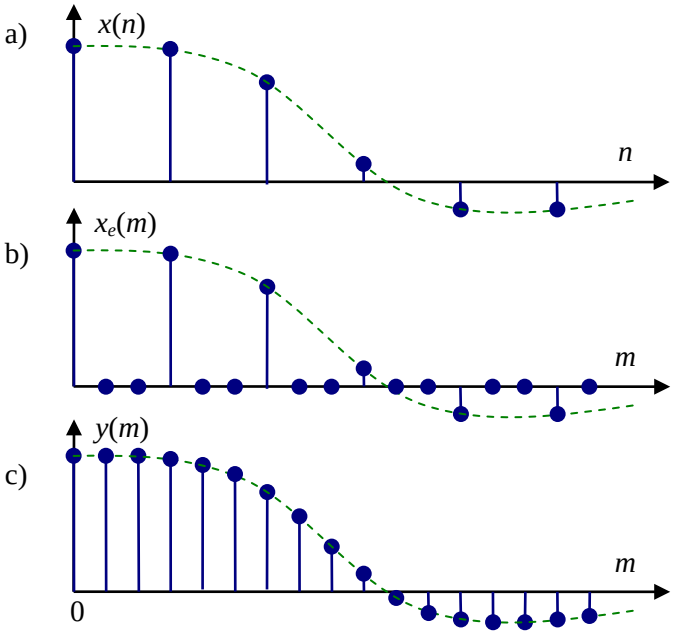
$$y(m) = h_I(m) * x_e(m) = \sum_{k=-\infty}^{\infty} h_I(m-k)x_e(k) = \sum_{k=-\infty}^{\infty} h_I(m-k)x(k/U), \tag{5.5}$$

a w dziedzinie transformaty Z:

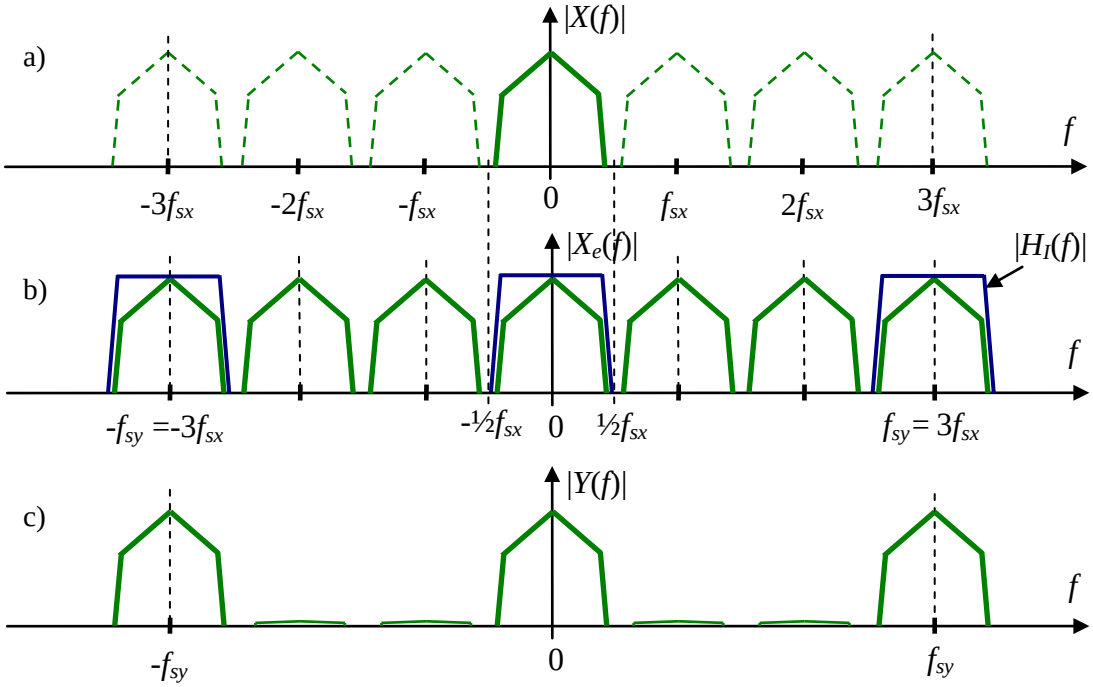
$$Y(z) = H_I(z)\cdot X_e(z) = H_I(z)\cdot X(z^U). \tag{5.6}$$



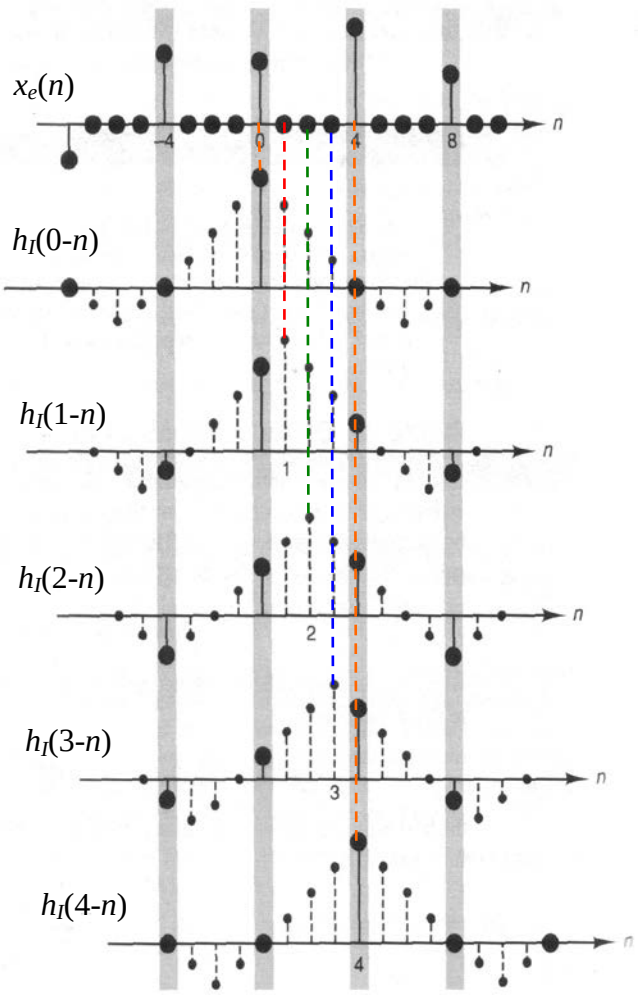
Rys. 1. Schemat interpolacji o czynnik U



Rys. 2. Przebiegi sygnałów w procesie interpolacji o czynnik U=3: a) sygnał wejściowy, b) sygnał wyjściowy z ekspandera (*upsampling*), c) sygnał wyjściowy po filtracji wygładzającej LP



Rys. 3. Widma sygnałów w procesie interpolacji o czynnik U=3: a) sygnał wejściowy, b) sygnał z ekspandera (*upsampling*) i charakterystyka $|H_I(f)|$ filtra LP, c) sygnał wyjściowy interpolatora po filtracji



Rys. 4. Interpolacja jako proces filtracji liniowej – splotu $x_e(n)$ z charakterystyką impulsową filtra $h_I(n)$. Odwrócona charakterystyka $h_I(-n)$ jest przesuwana na tle sygnału $x_e(n)$. Dla każdego przesunięcia $y(n)$ to suma iloczynów niezerowych próbek i współczynników filtra. Zwrócić uwagę, kiedy zera $x_e(n)$ nakładają się na $h_I(k-n)$ i kiedy zera $h_I(k-n)$ nakładają sę na $x_e(n)$.

W wielu przypadkach wystarczająco dobre wyniki daje zastosowanie prostej interpolacji liniowej lub ekstrapolacji zerowego rzędu ZOH (*zero order hold*; stosowana w przetwornikach C/A). W przypadku interpolacji ZOH każda próbka sygnału $x(n)$ jest po prostu powtarzana U razy:

$$y(m)=\begin{cases} x_e(0) & m=0,1,\ldots,U-1 \\ x_e(U) & m=U,U+1,\ldots,2U-1 \\ x_e(2U) & m=2U,2U+1,\ldots,3U-1 \\ \vdots \end{cases}, \tag{5.7}$$

co daje schodkowy sygnał wyjściowy i odpowiada filtrowi o przyczynowej charakterystyce impulsowej:

$$h_{ZOH}(m)=\delta(m)+\delta(m-1)+\ldots+\delta(m-(U-1)). \tag{5.8}$$

Interpolacja liniowa jest realizowana jako operacja nieprzyczynowa, której może odpowiadać charakterystyka impulsowa:

$$h_{lin}(m)=\begin{cases} 1-|m|/U & \text{dla } |m|\le U-1 \\ 0 & \text{dla pozostalych } m \end{cases} \tag{5.9}$$

(interpolacja tylko pomiędzy dwiema próbkami sygnału oryginalnego).

2.2. Decymacja sygnału

Decymacja jest operacją odwrotną do interpolacji. Decymacja o czynnik D polega na D-krotnym zmniejszeniu częstotliwości próbkowania sygnału. Pierwszym elementem decymatora jest filtr dolnoprzepustowy $H_D(z)$ (rys. 5), którego zadaniem jest ograniczenie pasma sygnału wejściowego $x(n)$ do zakresu D razy mniejszego. Charakterystyka widmowa idealnego filtra LP decymatora

$$H_D(e^{j\Omega})=\begin{cases} 1 & \text{dla } 0\le|\Omega|\le\dfrac{\pi}{D} \\ 0 & \text{dla pozostalych } \Omega \end{cases}. \tag{5.10}$$

gdzie skala $\Omega=\Omega_x=\Omega_y/D$. W praktyce filtr decymatora projektuje się zadając pulsację początku pasma *zaporowego* $\Omega_{stop}=\pi/D$.

Filtracja LP zapobiega aliasingowi po następującej dalej operacji redukcji częstotliwości próbkowania polegającej na pozostawieniu co D-tej próbki sygnału przefiltrowanego $x_f(n)$: $f_{sy}=f_{sx}/D$. Zmniejszenie szybkości próbkowania ogranicza na wyjściu reduktora D razy pasmo Nyquista, czyli zakres częstotliwości możliwych do reprezentowania w sygnale wyjściowym $y(m)$. Sygnał wyjściowy:

$$y(m)=x_f(n)|_{n=mD}=h_D(n)*x(n)|_{n=mD}=\sum_{k=-\infty}^{\infty}h_D(k)\cdot x(mD-k), \tag{5.11}$$

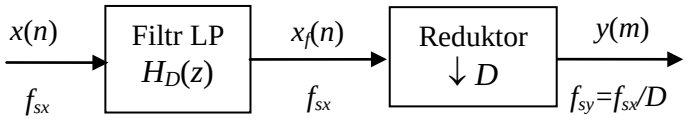
gdzie $h_D(n)$ jest charakterystyką impulsową filtra $H_D(z)$. W przypadku idealnej filtracji LP w dziedzinie transformaty Z:

$$Y(z)=\frac{1}{D}X(z^{1/D}). \tag{5.12}$$

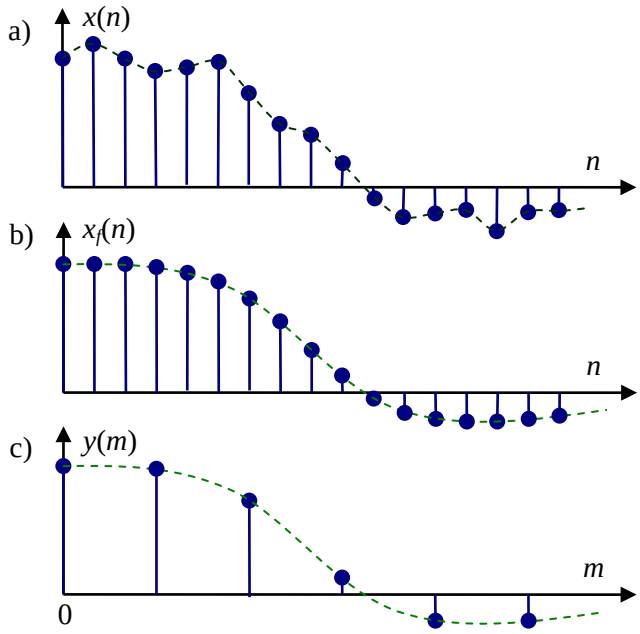
Filtracja LP sygnału w decymatorze może być realizowana w sposób nieprzyczynowy (*off-line*), podobnie jak w interpolatorze, lub przyczynowy (*on-line*) tylko na bazie próbek dostępnych do momentu aktualnego, co wprowadza przesunięcie fazowe. W przypadku filtra SOI rzędu $N=2L$ o symetrycznej odpowiedzi $h_D(n)$, charakterystyki przyczynowa i nieprzyczynowa są wzajemnie przesunięte:

$$h_{nieprzycz}(n)=h_{przycz}(n+L), \qquad H_{nieprzycz}(z)=z^L H_{przycz}(z). \tag{5.13}$$

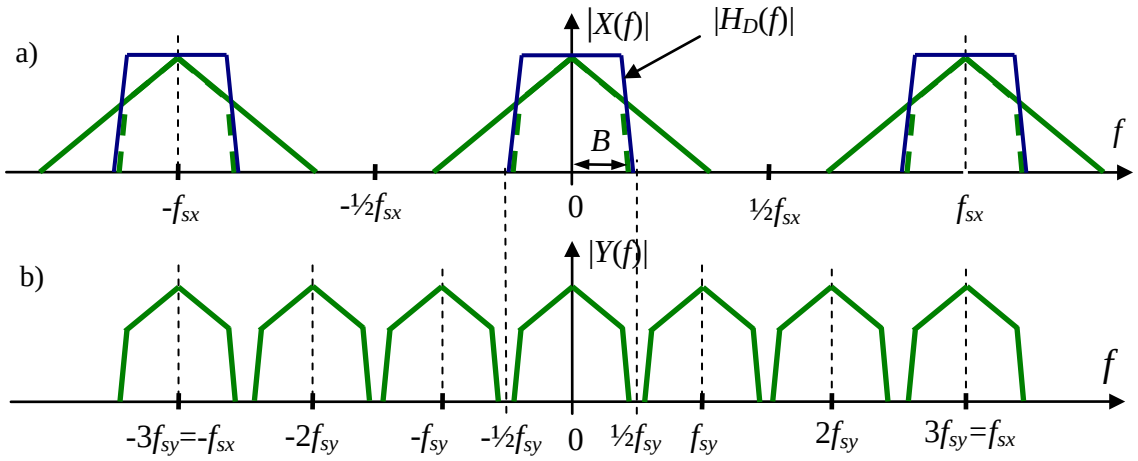
Ze względu na oszczędności obliczeniowe i łagodniejsze wymagania co do filtrów LP decymacja dla dużych czynników D realizowana jest jako kombinacja dwóch lub kilku decymacji z mniejszymi wartościami D.



Rys. 5. Schemat decymacji o czynnik *D*



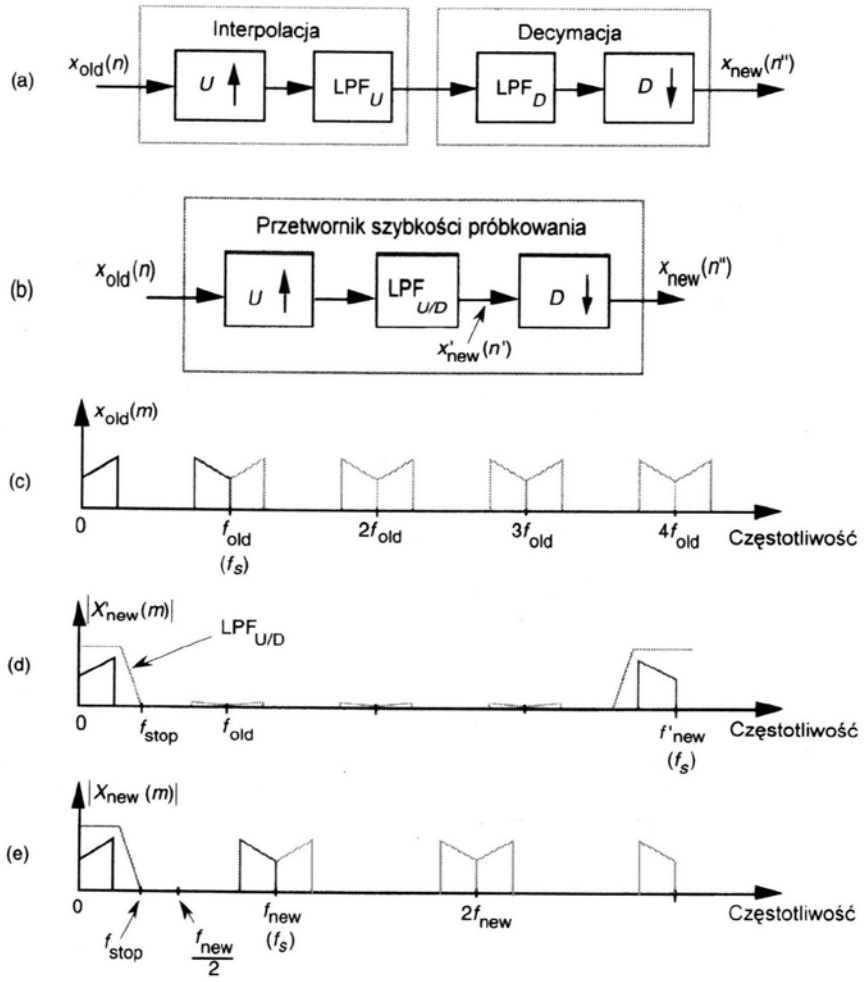
Rys. 6. Przebiegi sygnałów w procesie decymacji o czynnik *D*=3: a) sygnał wejściowy, b) sygnał po filtracji LP, c) sygnał wyjściowy po redukcji częstotliwości próbkowania (*downsampling*)



Rys. 7. Widma sygnałów w procesie decymacji o czynnik *D*=3: a) sygnał wejściowy i charakterystyka $|H_D(f)|$ filtra LP ograniczającego pasmo, b) sygnał wyjściowy decymatora po filtracji i redukcji częstotliwości próbkowania

2.3. Połączenie interpolacji i decymacji

Połączenie interpolacji i decymacji umożliwia zmianę szybkości próbkowania o czynnik wymierny *U*/*D* (rys. 8). Oba filtry LP połączone szeregowo pracują na częstotliwości wyjściowej ekspandera, co umożliwia zastąpienie ich pojedynczym filtrem o transmitancji $H_I(z)H_D(z)$.



Rys. 8. Przetworzenie częstotliwości próbkowania o czynnik U/D=4/3: a) kombinacja interpolacji i decymacji, b) zastąpienie dwóch filtrów LP jednym o transmitancji $H_I(z)H_D(z)$, c) widmo sygnału wejściowego, d) widmo sygnału wyjściowego filtra po interpolacji, e) ostateczne widmo sygnału wyjściowego po decymacji. Oznaczenia: $f_{sx}=f_{s\_old}$ $f_{sy}=f_{s\_new}$

2.4. Próbkowanie sygnału pasmowego

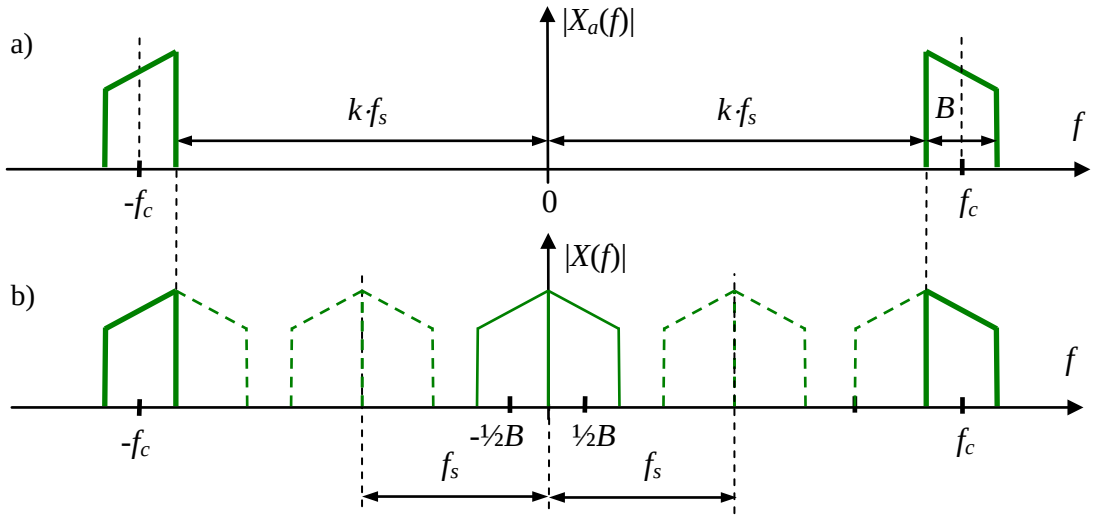
Standardowe próbkowanie dolnopasmowe sygnału o pasmowym widmie o szerokości B wokół częstotliwości nośnej (środkowej) f_c (rys. 9a) teoretycznie wymaga dużej częstotliwości próbkowania $f_s \geq f_c + B/2$. Zastosowanie redukcji szybkości próbkowania (downsamplingu, decymacji bez filtracji) umożliwia uzyskanie powielenia oryginalnego widma wokół częstotliwości zerowej wskutek aliasingu, bez straty informacji (rys. 9b). Jest to znacznie oszczędniejsza metoda reprezentacji sygnałów pasmowych stosowana w dekompozycji sygnału za pomocą banków filtrów pasmowoprzepustowych.

W celu uniknięcia nakładania się powieleń widma sygnału zredukowana częstotliwość próbkowania musi spełniać warunek

$$\frac{f_c + B/2}{k + \frac{1}{2}} \leq f_s \leq \frac{f_c - B/2}{k}, \quad f_s \geq 2B \tag{5.14}$$

gdzie k jest liczbą naturalną.

Odtworzenie sygnału oryginalnego jest możliwe po przeprowadzeniu interpolacji pasmowo-przepustowej: powrocie do większej (pierwotnej) częstotliwości próbkowania i filtracji BP wydzielającej tylko oryginalne pasmo sygnału.



Rys. 9. Próbkowanie sygnału pasmowego: a) widmo oryginalnego sygnału ciągłego, b) powielenia widma sygnału próbkowanego, B – szerokość pasma sygnału, f_c – częstotliwość nośna (centralna), f_s – częstotliwość próbkowania

2.5. Podpasmowa dekompozycja i rekonstrukcja sygnału za pomocą banku filtrów

Przetwarzanie sygnału w podpasmach częstotliwości jest stosowane w wielu praktycznych aplikacjach, m.in. algorytmach kompresji audio (MPEG-1,2 z poziomem mp3 oraz MPEG-4) i obrazów (JPEG2000). Jedną z metod dekompozycji sygnału na składowe pasmowe jest zastosowanie banku (zespołu) filtrów BP pokrywających zakres widma sygnału, a następnie decymacja składowych sygnału w podpasmach, co umożliwia np. kompresję sygnału (w zależności od wariancji składowych sygnału w podpasmach reprezentuje się je w formie zawierającej mniej lub więcej bitów).

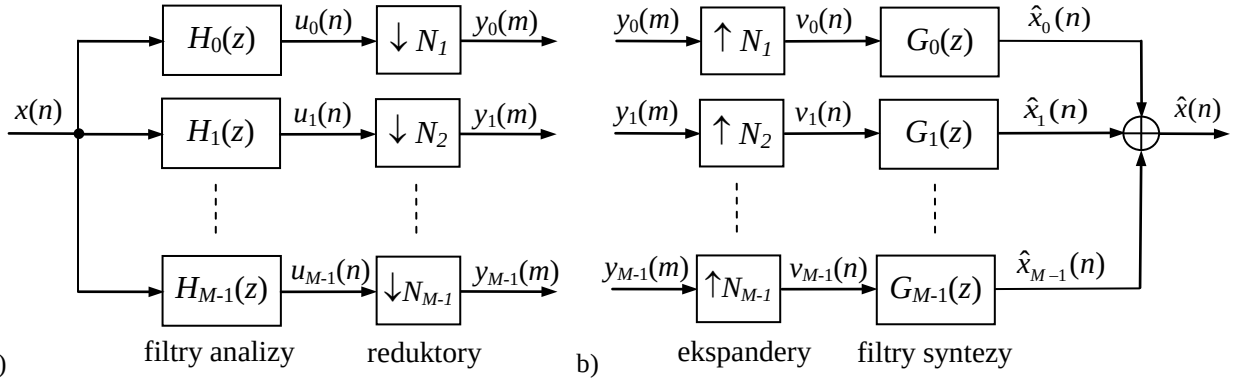
M-kanałowy bank filtrów do podpasmowej dekompozycji sygnału $x(n)$ jest przedstawiony na rys. 10a, a charakterystyki amplitudowo-częstotliwościowe jego filtrów BP otrzymane w wyniku rzeczywistej modulacji kosinusowej jednego prototypu - na rys. 11. Przy jednakowych szerokościach pasm wszystkie współczynniki reduktorów i ekspanderów są jednakowe $N_i=N$, $i=0,1,...,M-1$. Kiedy liczba pasm jest równa współczynnikowi redukcji, $N=M$, mamy do czynienia z *próbkowaniem krytycznym*, kiedy $N>M$ występuje podpróbkowanie, a kiedy $N<M$ występuje nadpróbkowanie. Odpowiednie umiejscowienie charakterystyk $H_k(e^{j\Omega})$ filtrów na osi częstotliwości uzyskuje się w tym przypadku jako efekt modulacji kosinusowej (w dziedzinie czasu) charakterystyki impulsowej $p(n)$, $0\le n\le L-1$, filtra prototypowego LP o pulsacji granicznej $\Omega_p=\pi/(2M)$, który należy zaprojektować. Charakterystyki impulsowe poszczególnych filtrów BP banku analizy:

$$h_k(n) = 2\,p(n)\cdot\cos\left[\frac{\pi}{M}\left(k+\frac{1}{2}\right)\left(n-\frac{L-1}{2}\right)+(-1)^k\frac{\pi}{4}\right],\quad k=0,1,\ldots,M-1 \tag{5.15}$$

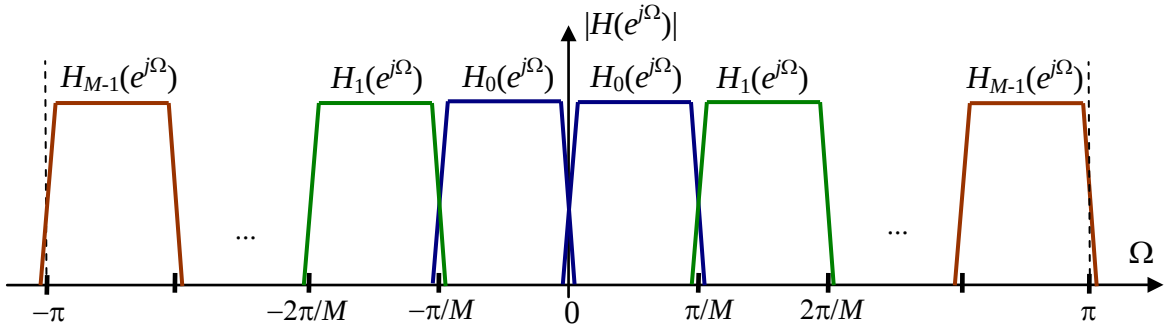
Modulacja kosinusowa charakterystyki impulsowej prototypu powoduje przesuwanie charakterystyki widmowej do odpowiedniego położenia na osi pulsacji Ω, ponieważ:

$$h(n)\cdot\cos\Omega_0 n = h(n)\frac{e^{-j\Omega_0 n}+e^{j\Omega_0 n}}{2} \quad\leftrightarrow\quad \frac{1}{2}\left[H(e^{j(\Omega-\Omega_0)})+H(e^{j(\Omega+\Omega_0)})\right] \tag{5.16}$$

Odtworzenie $\hat{x}(n)$ sygnału oryginalnego uzyskuje się po interpolacji, czyli przepuszczeniu zdecydowanych składowych $y_k(m)$ przez ekspandery i filtry BP rekonstrukcji (syntezy, rys. 10b). Jeżeli prototyp $p(n)$ jest symetryczny, to odpowiedzi impulsowe filtrów syntezy $g_k(n)$ są *odwróconymi w czasie* odpowiedziami $h_k(n)$.



Rys. 10. *M*-kanałowy bank (zespół) filtrów podpasmowej analizy (a) i syntezy (b) sygnału (*subband decomposition and reconstruction*)



Rys. 11. Charakterystyki amplitudowo-częstotliwościowe *M*-pasmowego banku filtrów otrzymane w wyniku rzeczywistej modulacji kosinusowej charakterystyki impulsowej $p(n)$ prototypu LP

Zależności pomiędzy transformatami *Z* sygnałów z rys.10 są następujące:

- po filtracji : $U_k(z) = H_k(z) \cdot X(z), \qquad k = 0,1,\ldots,M-1$ (5.17)

- po redukcji: $Y_k(z) = \frac{1}{N}\sum_{l=0}^{N-1} U_k(w_N^l z^{1/N}), \qquad w_N = e^{-j2\pi/N}$ (5.18)

$Y_k(z)$ jest sumą powielonych poprzesuwanych (zmodulowanych przez w_N^l) widm $U_k(z^{1/N})$.

- wyjścia ekspanderów: $V_k(z) = Y_k(z^N)$ (5.19)

- zrekonstruowane składowe: $\hat{X}_k(z) = G_k(z) \cdot V_k(z)$ (5.20)

Sygnał zrekonstruowany:

$$\hat{X}(z) = \sum_{k=0}^{M-1} \hat{X}_k(z) = \sum_{k=0}^{M-1} G_k(z) Y_k(z^N) = \sum_{k=0}^{M-1} \left(G_k(z) \frac{1}{N} \sum_{l=0}^{N-1} H_k(w_N^l z) X(w_N^l z) \right) =$$
$$= \sum_{l=0}^{N-1} \left(X(w_N^l z) \frac{1}{N} \sum_{k=0}^{M-1} H_k(w_N^l z) G_k(z) \right) = \sum_{l=0}^{N-1} X(w_N^l z) A_l(z) \qquad (5.21)$$

gdzie:

$$A_l(z) = \frac{1}{N} \sum_{k=0}^{M-1} H_k(w_N^l z) G_k(z) \qquad (5.22)$$

Jeżeli spełnione będą warunki:

$$\begin{aligned} A_0(z) &= \frac{1}{N}\sum_{k=0}^{M-1} H_k(z)G_k(z) = \alpha \cdot z^{-n_0} \\ A_l(z) &= 0, \quad l = 1,2,\ldots,N-1 \end{aligned}, \tag{5.23}$$

to odtworzenie $\hat{x}(n) = \alpha \cdot x(n-n_0)$ jest przeskalowanym i opóźnionym, ale niezniekształconym sygnałem oryginalnym, a składowe zmodulowane są wytłumione. Dlatego (5.23) są warunkami dokładnej rekonstrukcji (*perfect reconstruction* – PR), które powinny być spełnione, przynajmniej w przybliżeniu, przez zaprojektowany bank filtrów. Niezerowe wartości $A_l(z)$ dla $l \neq 0$, spowodowane nieidealnymi charakterystykami filtrów, powodują tzw. *przecieki widmowe* składowych pomiędzy podpasmami.

W algorytmach podpasmowego kodowania (kompresji) sygnałów, składowe podpasmowe $y_k(m)$ po redukcji są jeszcze kwantowane co do wartości. Liczba poziomów kwantowania R_k (czyli bitów reprezentacji) w podpaśmie jest tym większa, im większa jest wariancja var(y_k) składowej. Ponieważ całkowita liczba bitów na zakodowaną próbkę $R=\Sigma R_k$ jest ograniczona i zależy np. od założonego *bitrate* $I=R \cdot f_s$ (bitów/s, f_s – częstotliwość próbkowania), podpasma o najmniejszej wariancji mogą w ogóle nie być reprezentowane w zakodowanym sygnale.

3. Obliczenia komputerowe - zadania do wykonania wg instrukcji

Uwaga:
- Bloki instrukcji Matlaba do realizacji poszczególnych zadań wygodnie jest kopiować do edytora Matlaba i tam dokonywać edycji parametrów, dopisywania nowych linii itp., a następnie uruchamiać jako skrypty komendą **Run** ▷.
- Przy przechodzeniu do następnego punktu ćwiczenia zwracać uwagę, żeby nie dochodziło do ewentualnego nakładania się na siebie w tym samym oknie wykresów różnych wielkości. Normalnie rysowanie nowego wykresu kasuje poprzednią zawartość otwartego okna, o ile ta zawartość nie została przeznaczona do zachowania za pomocą instrukcji `hold on` (domyślne `hold off` kasuje poprzednią zawartość pola wykresu w otwartym oknie). Wstawianie `hold on` jest potrzebne, kiedy instrukcja zaleca nakładanie na siebie dla porównania kilku wykresów tej samej wielkości otrzymanych przez powtarzanie jakiegoś bloku instrukcji dla różnych wartości pewnych parametru(-ów). Po zakończeniu danego zadania takie okna wykresów należy zamykać.
- Do projektowania filtrów wykorzystywane są funkcje biblioteki Matlaba ***Signal Processing Toolbox***. Listing funkcji zapisanej w formie skryptu można otrzymać w oknie komend za pomocą instrukcji `type nazwa_funkcji`.

➢ W oknie **Current Folder** przejść do folderu **Moje dokumenty/MATLAB/DSP**. Jest to folder roboczy dla ćwiczeń z *Cyfrowego przetwarzania sygnałów*.

3.1. Interpolacja sygnału

A. Interpolacja nieprzyczynowa sygnału (o ograniczonym widmie) stanowiącego sumę dwóch składowych harmonicznych:

$$x(n) = A_1 \cos\Omega_1 n + A_2 \cos\Omega_2 n$$

$$A_1 = 1.0, \Omega_1 = f_1\pi, \quad A_2 = 0.4, \Omega_2 = f_2\pi$$

gdzie częstotliwości f są unormowane do częstotliwości Nyquista (tzn. f=1 odpowiada $f_{Nyq}=f_s/2$).

- Przeprowadzić interpolację nieprzyczynową o całkowity czynnik rozszerzenia (*upsamplingu*) U sygnału złożonego z dwóch składowych harmonicznych:

```
N=500; n=0:N-1;          % liczba próbek, wektor indeksów
fn = [0.1 0.8];          % częsttl. unormowane składowych, fn=f/fNyq
A = [1 0.6];             % amplitudy składowych
x = A*cos(pi*fn'*n);     % sygnał x(n)=A(1)*cos(W(1)*n) + A(2)*cos(W(2)*n)
U = 3;                   % współczynnik rozszerzenia (upsamplingu)
xup = upsample(x,U);% wyjście ekspandera - wstawienie zer
L=4; alpha=0.9;     % rząd filtra N=2*L*U, zakładane wypełnienie pasma Nyq.
[y,b] = interp(x,U,L,alpha);% interpolacja z filtrem nieprzyczyn. H(z)=b(z)
interpsignal(y,x,xup,U);    % przebiegi sygnałów w procesie interpolacji
interpspect(y,x,xup,U);     % widma DFT sygnałów w procesie interpolacji
interpfilt(b);              % odp. częstotl. i impulsowa filtra interpolatora
```

- Uwaga: W obliczeniach wykorzystywane są funkcje `interp` i `upsample` Matlaba oraz funkcje wykreślania przebiegów i widm sygnałów oraz charakterystyk filtra interpolatora.
  Sprawdzić, czy pliki funkcji `interpsignal.m`, `interpspect.m` i `interpfilt.m` oraz `dft1.m` znajdują się w folderze roboczym. Jeżeli nie, podane poniżej listingi tych funkcji należy skopiować do edytora Matlaba i zapisać w osobnych plikach o podanych nazwach w folderze roboczym.
  Okna wykresów z kilkoma subplotami należy w razie potrzeby rozciągać w pionie i/lub poziomie dla poprawienia widoczności szczegółów.

```
function interpsignal(y,x,xup,U,nw)
% Wykresy przebiegów sygnałów w procesie interpolacji
% Input: x - sygnal wej., xup - wyj. ekspandera (upsampled),
% y - wyj. interpolatora, U - wspolcz. upsamplingu, nw=[n0,nf] - zakres OX
if nargin<5, nw=[0 100]; end; % domyślny zakres osi OX przebiegów czasowych
n = U*[0:length(x)-1];        % czas dyskretny
```

```
nup = 0:length(xup)-1;
subplot(311), stem(n,x,'o'), ylabel('x')
title('Interpolacja - przebiegi sygnałów'),
xlim([nw(1) nw(2)])           % zakres osi OX
subplot(312), stem(nup,xup,'o'), ylabel('x_{upsamp}')
xlim([nw(1) nw(2)])
subplot(313), stem(nup,y,'or'), ylabel('y_{interpol}'), xlabel('n')
xlim([nw(1) nw(2)])
-----------------------------------
function [Y,fy,Xup,X,fx] = interpspect(y,x,xup,U)
% Widma amplitudowe DFT sygnałów w procesie interpolacji
% Input: x - sygnal wejsciowy, xup - wyj. ekspandera (upsampled),
% y - wyjście interpolatora, u - wspolcz. upsamplingu
Nx=length(x); Ny=length(y);
[X,Fx] = dft1(x,Nx);      % częstotl. unorm. do [-1,1), widmo w skali Fx
[Xup,Fy] = dft1(xup,Ny);  % widmo po upsamplingu w skali Fy
[Y,Fy] = dft1(y,Ny);      % widmo syg. interpolowanego (po filtracji), Fy
if nargout==0,
  figure    % częstotl. Fx, Fy unorm. do fxNyq, fyNyq
  subplot(311), plot(Fx,abs(X)),
  title('Interpolacja - widma DFT sygnałów'),
  xlabel('\Omega_x / \pi'), ylabel('|X(\Omega_x)|')
  subplot(312), plot(Fy,abs(Xup)),
  xlabel('\Omega_y / \pi'); ylabel('|X_{upsamp}(\Omega_y)|')
  subplot(313), plot(Fy,abs(Y),'r'),
  xlabel('\Omega_y / \pi'); ylabel('|Y_{interpol}(\Omega_y)|')
end;
-----------------------------------
function interpfilt(b)
% Wykreślanie ch-ki wzmocnienia |H(W)| i impulsowej h(n) fitra H(z)=b(z)
L=(length(b)-1)/2;
n=0:length(b)-1;
W=linspace(-pi,pi,400); % wektor pulsacji, dla których obliczane jest H(W)
[H,wh]=freqz(b,1,W);    % odp. częstotliwościowa filtra B(z)
figure
subplot(211), plot(wh/pi,20*log10(abs(H))), % ch-ka częstotl. amplitudowa
title('Charakterystyki filtra interpolatora'), axis([-1 1 -80 20]), grid on
xlabel('\Omega_{y} / \pi'), ylabel('|H(\Omega)|  (dB)')
subplot(212), stem([-L:L],b,'r'),          % ch-ka impulsowa
ylabel('h(n)'), xlabel('n')
-----------------------------------
function [X,W] = dft1(x,N)
% DTF-1D sygnału x dla N pulsacji unormowanych do pulsacji Nyquista fs/2
% Dane: x - sygnał o długości L, N - liczba punktów DFT
% Wyniki: X = dft1(x), W - wektor N pulsacji unorm. [-1,1) z zerem w środku
L=length(x);
if nargin==1 N=L; end
if (N<L) error('DTFT: liczba próbek L > liczby pulsacji N'); end;
W = (2/N)*[0:N-1];
center=ceil(N/2)+1;
W(center:N)=W(center:N)-2;      % przesunięcie [1,2) do [-1,0)
W = fftshift(W);                % zamiana połówek W do zakresu [-1, 1)
X = fftshift(fft(x,N));
```

- Parametr $0<\alpha \leq 1$ funkcji `interp` jest zakładanym wypełnieniem pasma Nyquista przez widmo sygnału wejściowego $x(n)$. Mniejsza wartość α poszerza pasmo przejściowe filtra i zwiększa tłumienie w paśmie zaporowym dla danego rzędu filtra, ale powoduje powstanie regionów „don't care" w paśmie zaporowym. `b=[b0, b1, ..., bN]` – współczynniki (odpowiedź impulsowa) filtra $H(z)=b(z)$.
- Osie pulsacji na wykresach widm DFT są wyskalowane w jednostkach unormowanych do odpowiedniej pulsacji Nyquista $\Omega_{Nyq}=\pi$ (rad/sample) zarówno przed jak i po zwiększeniu

częstotliwości próbkowania. Należy pamiętać, że częstotliwość próbkowania w jednostkach fizycznych jest po upsamplingu U razy większa.

- Wyskalować osie częstotliwości przed i po interpolacji w Hz przyjmując częstotliwość próbkowania przed interpolacją f_{sx}=1000 Hz (upsampling U=3).
- Wyjaśnić zmianę położenia prążków widma składowych sygnału na osi unormowanej częstotliwości po interpolacji.
- Czy wartość parametru α jest poprawna biorąc pod uwagę wypełnienie pasma Nyquista przez wygenerowany sygnał wejściowy (najwyższą częstotliwość w sygnale wejściowym)?
- Powtórzyć interpolację dla `U=3` i czterech kombinacji parametrów: `L = 4, 8; alpha = 0.4, 1.0`.
- Porównać rezultaty interpolacji oraz charakterystyki częstotliwościowe i impulsowe filtra interpolatora. Zaobserwować i wyjaśnić położenie miejsc zerowych odpowiedzi impulsowej $h(n)$.
- Dla jakiego α filtr interpolatora pokrywa się najlepiej z filtrem opisanym wzorem (5.4)? Która z zadanych wartości α jest niepoprawna dla interpolowanego sygnału (zwrócić uwagę na poziom tłumienia składowej f_2=0.9 w zależności od wartości α)?.

B. Interpolacja nieprzyczynowa *liniowa* pomiędzy sąsiednimi próbkami – charakterystyka impulsowa filtra interpolatora jest opisana wzorem (5.9). Przeprowadzić interpolację liniową sygnału jak w pkt. A (powinien być zapisany w zmiennej `x` w pamięci roboczej Matlaba) dla `U=5` i `U=8`:

```
...
U = 5;                  % współczynnik rozszerzenia (upsamplingu)
b=1-abs(-(U-1):U-1)/U;  % odp.impuls. h(n)=b(n-U+1) filtra interpolatora lin.
y=conv(b,xup);          % splot sygnału z zerami z odp. impuls. filtra
y=y(U:U+length(xup)-1); % wyjście interpolatora - wycinek splotu
interpsignal(y,x,xup,U);    % przebiegi sygnałów w procesie interpolacji
interpspect(y,x,xup,U);     % widma DFT sygnałów w procesie interpolacji
interpfilt(b);              % odp. częstotl. i impulsowa filtra interpolatora
```

- Porównać wyniki interpolacji z uzyskanymi w pkt. A.

C. Interpolacja sygnału dźwiękowego (*** należy włączyć głośniki komputera ***).

- Wczytać plik dźwiękowy w formacie `wav` o podanej nazwie do zmiennej `x` i przeprowadzić interpolację sygnału.
  Częstotliwość próbkowania `fs` i liczba bitów próbki `nbits` sygnału są podawane w oknie komend Matlaba. Sygnał po interpolacji jest odtwarzany *po naciśnięciu dowolnego klawisza* w aktywnym oknie komend.

```
[x,fs,nbits] = wavread('fire'); % wczytanie sygnału do zmiennej x
fs              % częstotliwość próbkowania (Hz)
nbits           % liczba bitów / próbkę
sound(x,fs)     % odtworzenie sygnału na wyjściu audio
nw=[2000 2200]; % zakres indeksów próbek do rys. przebiegów czasowych
U=2;            % współczynnik rozszerzenia (upsampling)
xup = upsample(x,U);% wyjście ekspandera - wstawienie zer
L=4; alpha=0.9;     % rząd filtra N=2*L*U, zakładane wypełnienie pasma Nyq.
[y,b] = interp(x,U,L,alpha);% interpolacja z filtrem nieprzyczyn. H(z)=b(z)
interpsignal(y,x,xup,U,nw); % przebiegi sygnałów w procesie interpolacji
interpspect(y,x,xup,U);     % widma DFT sygnałów w procesie interpolacji
interpfilt(b);              % odp. częstotl. i impulsowa filtra interpolatora
disp('Press any key...'), pause
sound(y,U*fs)   % odtworzenie sygnału po interpolacji
```

- Zwrócić uwagę na wykresy widm DFT sygnałów i porównać je Rys. 3 mając na uwadze inny sposób skalowania osi częstotliwości.
- Czy słyszalne są różnice dźwięku przed i po interpolacji?

3.2. Decymacja sygnału

A. Decymacja bez przesunięcia fazowego (SOI ze zwierciadlanym uzupełnianiem próbek wejściowych w celu uniknięcia przebiegów przejściowych lub nieprzyczynowa NOI).

- Wygenerować sygnał wejściowy zawierający dwie składowe harmoniczne i przeprowadzić decymację sygnału. Do realizacji decymacji wykorzystywana jest funkcja `decimate` Matlaba, która domyślnie stosuje przed redukcją filtrację LP *zero-phase* (filtracja *nieprzyczynowa z zerowym przesunięciem fazowym* za pomocą funkcji `filtfilt`) filtrem NOI Czebyszewa I rzędu N=8 o częstotliwości odcięcia f_0=0.8*(f_s/2)/D, gdzie D jest czynnikiem redukcji (*downsamplingu*).

```
N=500; n=0:N-1;          % liczba próbek, wektor indeksów
fn = [0.05 0.4];         % częstotl. unormowane składowych, fn=f/fNyq
A = [1 0.6];             % amplitudy składowych
x = A*cos(pi*fn'*n);     % sygnał x(n)=A(1)*cos(W(1)*n) + A(2)*cos(W(2)*n)
D = 2;                   % współczynnik redukcji (downsamplingu)
xdown = downsample(x,D);% redukcja sygnału - downsampling bez filtracji LP
y = decimate(x,D);       % decymacja nieprzyczynowa zero-phase
Wp=0.9/D, N=8; rp=0.05; % parametry filtra Czebyszewa I
[b,a]=cheby1(N,rp,Wp);  % H(z)=b(z)/a(z)
W=linspace(-pi,pi,400); % wektor pulsacji, dla których obliczane jest H(W)
[H,W]=freqz(b,a,W);      % odp. częstotliwościowa filtra H(W)
decimsignal(y,x,xdown,D);   % przebiegi sygnałów w procesie decymacji
decimspect(y,x,xdown,D);    % widma DFT sygnałów w procesie decymacji
figure
plot(W/pi,20*log10(abs(H))), % ch-ka częstotl. amplitud. filtra decymatora
title('Charakterystyka filtra decymatora'), grid on
xlabel('\Omega_{x} / \pi'), ylabel('|H(\Omega)|  (dB)'), axis([-1 1 -80 10])
```

- Uwaga: W obliczeniach wykorzystywane są funkcje `decimate` i `downsample` Matlaba oraz funkcje wykreślania przebiegów i widm sygnałów oraz charakterystyk filtra decymatora.
  Sprawdzić, czy pliki funkcji `decimsignal.m`, `decimspect.m` i `dftdupl.m` znajdują się w folderze roboczym. Jeżeli nie, podane poniżej listingi tych funkcji należy skopiować do edytora Matlaba i zapisać w osobnych plikach o podanych nazwach w folderze roboczym.

```
function [y,xdown,ndown] = decimsignal(y,x,xdown,D,nw)
% Wykresy przebiegów sygnałów w procesie decymacji
% Input: y - sygnał po decymacji, x - sygnal wej., xdown - wyj. reduktora
% D - wspolcz. decymacji, nw=[n0,nf] - zakres OX
if nargin<5, nw=[0 100]; end; % domyślny zakres osi OX przebiegów czasowych
n = 0:length(x)-1;          % czas dyskretny przed decymacją
ndown = downsample(n,D);    % downsampling indeksów czasu
figure
subplot(311), stem(n,x,'o'), ylabel('x')
title('Decymacja - przebiegi sygnałów'),
xlim([nw(1) nw(2)])         % zakres osi OX
 subplot(312), stem(ndown,xdown,'o'), ylabel('x_{downsamp}')
 xlim([nw(1) nw(2)])
subplot(313)
stem(ndown,y,'or'), ylabel('y_{decim}'), xlabel('n')
xlim([nw(1) nw(2)])
-------------------------------
function [Y,fy,Xdown,X,fx] = decimspect(y,x,xdown,D)
% Widma amplitudowe DFT sygnałów w procesie decymacji
% Input: x - sygnal wejscioFy, xdown - syg. downsampled bez filtracji,
% y - sygnal po decymacji z filtracja LP, D - wspolcz. decymacji
Nx=length(x); Ny=length(y);
[X,Fx] = dft1(x,Nx);
[Xdown,Fx] = dft1(xdown,Nx);
% [Xdownd,Fxd] = dftdupl(Xdown,Fx,D); % powielenie widma Xdown w skali Fx
[Xdown,Fy] = dft1(xdown,Ny);        % widmo Xdown w skali Fy
[Y,Fy] = dft1(y,Ny);
```

```
if nargout==0,
  figure    % częstotl. Fx, Fy unorm. do fxNyq, fyNyq
  subplot(311), plot(Fx,abs(X)),
  title('Decymacja - widma DFT sygnałów'),
  xlabel('\Omega_x / \pi'), ylabel('|X(\Omega_x)|')
  subplot(312), plot(Fy,abs(Xdown)),
  xlabel('\Omega_y / \pi'); ylabel('|X_{downs}(\Omega_y)|')
  subplot(313), plot(Fy,abs(Y),'r')
  xlabel('\Omega_y / \pi'); ylabel('|Y_{decim}(\Omega_y)|')
end;
--------------------------------
function [Xd,Fd] = dftdupl(X,F,k)
% Powielanie widma X(W) k-razy co 2
% Input: X - widmo oryginalne, F - wektor pulsacji z zakresu [-1,1)
% Output: Xd - widmo powielone, Wd - pulsacja powielona z okresem 2
N=length(X);
Xd=zeros(k*N,1);
Fd=zeros(k*N,1);
for i=1:k,
  Xd(1+(i-1)*N:i*N)=fftshift(X);
  Fd(1+(i-1)*N:i*N)=F-(k-2*i+1);
end;
Xd=fftshift(Xd);
```

- Wyskalować osie częstotliwości przed i po decymacji w Hz przyjmując częstotliwość próbkowania przed decymacją f_{sx}=1000 Hz (downsampling D=2).
- Wyjaśnić zmianę położenia prążków widma składowych sygnału na osi unormowanej częstotliwości po interpolacji. Jakie są unormowane częstotliwości składowych sygnału po decymacji? Czy mieszczą się w paśmie Nyquista po decymacji?
- Na wykresach przebiegów czasowych zwrócić uwagę na brak przesunięcia fazowego sygnałów wejściowego i wyjściowego decymatora.

- Powtórzyć obliczenia dla decymacji o czynnik `D=3 i D=4`.

- Wyjaśnić różnice przebiegów i widm sygnału po redukcji (*downsampled*) w porównaniu z sygnałem decymowanym z filtracją LP, biorąc pod uwagę D-krotne zawężenie pasma Nyquista w wyniku redukcji. Czy obie składowe sygnału mieszczą się w paśmie Nyquista po decymacji? Wyjaśnić położenie prążka widma reprezentującego składową sygnału zredukowanego o większej częstotliwości jako efekt aliasingu.

- Wywołanie funkcji `decimate` z opcją `'fir'` realizuje decymację nieprzyczynową z filtrem SOI wyższego rzędu niż filtr NOI Czebyszewa (realizacja lub pominięcie tego podpunktu zależne od decyzji prowadzącego zajęcia):

```
...
y = decimate(x,D,'fir');      % decymacja z filtrem SOI
N=30; alpha=0.8; Wp=alpha/D;  % parametry domyślne filtra SOI LP
b = fir1(N,Wp); a=1;          % H(z)=b(z), a(z)=1
...
```

B. Akustyczna prezentacja efektów decymacji i redukcji sygnału.
*** Należy włączyć głośniki komputera *** i przeprowadzić obliczenia dla sygnału wygenerowanego jak w pkt. A (ale z większą liczbą próbek) w zakresie częstotliwości akustycznych dla `D=2 i D=3.` Sygnał po decymacji oraz sygnał po redukcji (tylko downsampling) są odtwarzane na wyjściu audio po kolei *po naciśnięciu dowolnego klawisza* w aktywnym oknie komend.

```
N=12000; n=0:N-1;       % liczba próbek, wektor indeksów
fn = [0.05 0.4];        % częstotl. unormowane składowych, fn=f/fNyq
A = [1 0.6];            % amplitudy składowych
x = A*cos(pi*fn'*n);    % sygnał x(n)=A(1)*cos(W(1)*n) + A(2)*cos(W(2)*n)
D = 2;                  % współczynnik redukcji (downsamplingu)
```

```
xdown = downsample(x,D);% redukcja sygnału - downsampling bez filtracji LP
y = decimate(x,D);      % decymacja nieprzyczynowa z filtrem NOI
fs = 11025;             % częstotl. próbkowania odtwarzanego sygnału w Hz
sound(x,fs)             % odtworzenie sygnału oryginalnego
disp('Press any key...'), pause
sound(y,fs/D)           % odtworzenie sygnału po samej decymacji
disp('Press any key...'), pause
sound(xdown,fs/D)       % odtworzenie sygnału po samej redukcji
```

- Porównać dźwięki po decymacji i redukcji ze sobą oraz z dźwiękiem oryginalnym. Dlaczego dla `D=3` sygnał po redukcji brzmi inaczej niż sygnał po decymacji?

C. Decymacja przyczynowa z przesunięciem fazowym.

- Przeprowadzić obliczenia decymacji z filtrem przyczynowym będącym aproksymacją SOI idealnego filtra LP o nieskończonej odpowiedzi impulsowej opisanej funkcją *sinc* (Zobacz informacje **Help** do funkcji `sinc`). Charakterystyka impulsowa filtra rzędu $N=2L$:

$$h_{FIR}(n) = \frac{\alpha}{D}\mathrm{sinc}\left(\frac{\alpha(n-L)}{D}\right), \quad 0 \le n \le 2L$$

```
N=500; n=0:N-1;         % liczba próbek, wektor indeksów
fn = [0.05 0.4];        % częstotl. unormowane składowych, fn=f/fNyq
A = [1 0.6];            % amplitudy składowych
x = A*cos(pi*fn'*n);    % sygnał x(n)=A(1)*cos(W(1)*n) + A(2)*cos(W(2)*n)
D = 2;                  % współczynnik redukcji (downsamplingu)
xdown = downsample(x,D);% redukcja sygnału - downsampling bez filtracji LP
alpha=0.9; L=8;         % wypełnienie pasma Nyquista, rząd filtra N=2*L
b = alpha/D * sinc(alpha*[-L:L]/D); % odp.impulsowa filtra LP SOI h(n)=b(n)
xf = filter(b,1,x);     % filtracja LP przyczynowa, a(z)=1
y = downsample(xf,D);   % wyj. decymatora: redukcja po ograniczeniu widma
W=linspace(-pi,pi,400); % wektor pulsacji, dla których obliczane jest H(W)
[H,W]=freqz(b,1,W);     % odp. częstotliwościowa filtra H(W)
decimsignal(y,x,xdown,D);   % przebiegi sygnałów w procesie decymacji
decimspect(y,x,xdown,D);    % widma DFT sygnałów w procesie decymacji
figure
subplot(211), plot(W/pi,20*log10(abs(H))),% ch-ka amplitud. filtra decymatora
title('Charakterystyka filtra decymatora'), grid on
xlabel('\Omega / \pi'), ylabel('|H(\Omega)|  (dB)'), axis([-1 1 -80 10])
subplot(212), stem([0:2*L],b,'r'),        % ch-ka impulsowa
ylabel('h(n)'), xlabel('n')
```

- Przeprowadzić obliczenia i porównać wyniki dla czterech kombinacji parametrów: `D=2` i `D=3` oraz `L=8` i `L=15`.

- Zwrócić uwagę na stan przejściowy na początku sygnału wyjściowego decymatora i jego przesunięcie fazowe w stanie ustalonym w stosunku do sygnału wejściowego oraz ich zależność od L i wyjaśnić ten efekt.

D. Decymacja sygnału dźwiękowego (*** należy włączyć głośniki komputera ***).

- Wczytać plik dźwiękowy w formacie `wav` o podanej nazwie do zmiennej `x` i przeprowadzić decymację nieprzyczynową *zero-phase* sygnału. Częstotliwość próbkowania `fs` sygnału wejściowego są podawane w oknie komend Matlaba. Sygnał po decymacji oraz sygnał po redukcji (tylko downsampling) są odtwarzane po kolei *po naciśnięciu dowolnego klawisza* w aktywnym oknie komend.

```
[x,fs] = wavread('fire'); % wczytanie sygnału do zmiennej x
fs              % częstotliwość próbkowania (Hz)
sound(x,fs)     % odtworzenie sygnału na wyjściu audio
nw=[2000 2400]; % zakres indeksów próbek do rys. przebiegów czasowych
D = 2;          % współczynnik redukcji (downsampling)
xdown = downsample(x,D);% redukcja - downsampling bez wstępnej filtracji LP
y = decimate(x,D);      % decymacja nieprzyczynowa zero-phase
```

```
decimsignal(y,x,xdown,D,nw);% przebiegi sygnałów w procesie decymacji
decimspect(y,x,xdown,D);    % widma DFT sygnałów w procesie decymacji
disp('Press any key...'), pause
sound(y,fs/D)   % odtworzenie sygnału po decymacji
disp('Press any key...'), pause
sound(xdown,fs/D) % odtworzenie sygnału po samej redukcji
```

- Powtórzyć eksperyment i odsłuchać wyniki dla decymacji (redukcji) o czynnik D=5 oraz D=8.

➢ Skomentować słyszalne różnice dźwięku przed i po decymacji? Porównać zawartość widmową sygnału wejściowego i wyjściowego. Zwrócić uwagę na widoczne ograniczenie zakresu widma sygnału po decymacji.

➢ Porównać dźwięk po prawidłowej decymacji (z wstępną filtracją LP) z dźwiękiem po samej redukcji.

3.3. Dekompozycja sygnału za pomocą banku filtrów pasmowoprzepustowych BP

A. Projektowanie filtra prototypowego LP SOI do banku filtrów.

- Wprowadzić dane projektowe i zaprojektować prototyp z wykorzystaniem funkcji optymalnego projektowania prototypu prototype.

```
M = 4;    % liczba kanałów banku filtrów - decyduje o szer. pasma prototypu
rs = 96;  % min. tłumienie w paśmie zaporowym w dB (stopband rippling)
L = 201;  % dlugosc odpowiedzi impulsowej p(n) prototypu (= rząd filtra +1)
p = prototype(M,L,rs);  % wyznaczenie odp. impulsowej p(n) prototypu
prototype(M,L,rs);      % wykresy ch-tyk prototypu
```

- Uwaga: W obliczeniach przeprowadzanych w tym punkcie wykorzystywane są funkcje z pracy Zielińskiego [6]. Sprawdzić, czy pliki funkcji prototype.m, bank1.m, bank2.m znajdują się w folderze roboczym. Jeżeli ich tam nie ma, podane poniżej listingi tych funkcji należy skopiować do edytora Matlaba i zapisać w osobnych plikach o podanych nazwach w folderze roboczym.

```
function h = prototype(M,L,rs,alfa,Lp,ifast)
% Projektowanie filtrów prototypowych dla zespołu filtrów
% z modulacją kosinusową DCT-IV
% Input: M - liczba kanałów, L - długość filtra (rząd N + 1)
%        rs - tlumienie  w pasmie zaporowym (dB)
%        alfa - współcz. pasma przejsciowego df=alfa*fc
%        Lp - liczba punktów ch-ki częstotliwościowej
%        ifast - 1=szybka metoda wyznaczenia czy filtr jest M-pasmowy,
% Zielinski, Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań, str.536

if nargin<6, ifast=1; end; % Domyslne parametry wejściowe
if nargin<5, Lp=128; end;
if nargin<4, alfa=0.9; end;
if nargin<3, rs=96; end;
impeg = 0;      % 0 = długość parzysta , 1 = długość nieparzysta
tol = 1e-8;     % tolerancja obliczeń filtra
% Inicjalizacja parametrów wewnętrznych
    MM = 2*M;       % liczba kanałów zespolonych
    fc = 1/MM;      % częstotliwość graniczna (odcięcia)
    df = alfa * fc; % pasmo przejściowe
% Oblicz długość filtra
    if (rs>21) D = (rs-7.95)/14.36; else D = 0.922; end
    Nb = D/df+1;
% Współczynnik beta okna Kaisera
    if (rs<=21) beta = 0.; end
    if ((21<rs) & (rs<50)) beta = 0.5842*(rs-21)^0.4+0.07886*(rs-21); end
    if (rs>=50) beta = 0.1102*(rs-8.7); end
% Charakterystyka filtra
    h = fir1(L-1, fc, kaiser(L, beta));
% Błąd charakterystyki częstotliwościowej
    err_new = 0;
```

```
    n = 0 : L-1;
    for f = 0 : (1/MM)/(Lp-1) : 1/MM;
       H1 = exp(-j*2*pi*f*n) * h';
       H2 = exp(-j*2*pi*(f-1/MM)*n) * h';
       err_new = max(err_new, abs(H1*conj(H1)+H2*conj(H2)-1));
    end
    err_old = 0.0;
% Inicjalizacja optymalizacji
    fc_opt = fc;
    fc_step = fc/100;
    direct = -1;
% Pętla główna
  while ( abs(err_new - err_old) > tol )
     err_old = err_new; fc_opt = fc_opt + direct * fc_step;
     h = fir1(L-1, fc_opt, kaiser(L, beta));
     if(ifast==0)    % wersja wolna --------------
        err_new = 0;
        n = 0 : L-1;
        for f = 0 : (1/MM)/(Lp-1) : 1/MM;
            h1 = sum(h .* exp( -j*2*pi*f*n));
            h2 = sum(h .* exp( -j*2*pi*(f-1/MM)*n));
            h1 = h1 * conj(h1);
            h2 = h2 * conj(h2);
            err_new = max(err_new, abs(h1+h2-1));
        end
     end
     if(ifast==1)   % wersja szybka -------------
        err_new = 0; rh = xcorr(h); nf = 1:floor((L-1)/(2*M));
        err_new = max(err_new, sum(abs(rh(L + MM * nf))));
     end
     if (err_new > err_old)
       fc_step = fc_step/2; direct = -direct;
     end
  end
 if (impeg==1) h = [0  h]; L = L+1; end  % Dopasow. do standardu MPEG audio
 h = h'; % save p.dat h /ascii
 Nf=2048;
 W = -pi:2*pi/Nf:pi;
 H = freqz(h,1,W);
if nargout==0,
    disp('Parametry filtra prototypowego i okna Kaisera')
    fc, Nb, beta
    disp('Parametry optymalizacji')
    fc_opt, err_new   % podglad wybranych parametrów optymalizacji
    figure
    stem(n,h,'o'); grid; ylabel('h(n)'), xlabel('n');
    title('Ch-ka impulsowa h(n)=p(n) filtra prototypowego LP SOI'),
    figure
    subplot(211), plot(W/pi,abs(H)), grid; ylabel('|H(\Omega)|'),
    title('Ch-ka częstotliwosciowa prototypu p(n)'), ylim([0 1.2]),
    subplot(212), plot(W/pi,20*log10(abs(H))), grid; axis([-1 1 -180 20]),
    ylabel('|H({\Omega})| (dB)'), xlabel('\Omega / \pi');
end;
----------------------------
function [h,g] = bank1(M,p)
% Projektowanie banku filtrów analizy i syntezy z modulacją kosinusową
% Input:    M - liczba kanałów,  p - ch-ka impulsowa p(n) prototypu LP,
%           fs - częstotl. próbkowania (Hz)
% Output:   h - macierz MxL odp. impulsowych f. analizy,
%           g - macierz MxL odp. impulsowych f. syntezy,
% Zielinski, Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań, str.537
```

```
L = length(p);  % długość filtra (rząd + 1)
skala = sqrt(2.0/(M*(p'*p)));
p = skala*p;    % skalowanie energii odp. impulsowej prototypu
Nf=2048; W = -pi:2*pi/Nf:pi;
P = freqz(p,1,W);
% Generowanie odpowiedzi impulsowych i częstottliwościowych filtrów analizy i
syntezy
h = zeros(M,L); g = h;
H = zeros(M,Nf+1); G = H;
n = 0:L-1; W = -pi:2*pi/Nf:pi;
% Projektowanie M filtrów BP banku za pomocą modulscji kosinusowej
for k = 0:M-1,
    h(k+1,n+1) = p'.*cos( pi./M *(k+0.5)*(n+M/2) ); % filtry analizy
    g(k+1,n+1) = p'.*cos( pi./M *(k+0.5)*(n-M/2) ); % filtry syntezy
    H(k+1,:) = freqz(h(k+1,:),1,W);
    G(k+1,:) = freqz(g(k+1,:),1,W);
end
if nargout==0,  % -------- Wykresy ------------
  disp('Parametry odp. impulsowej p(n)');
  suma = sum(p), maxp = max(p), energy  = p' * p, dcgain = abs(P(1)),
% ------- Filtry analizy (dekompozycji) --------
  figure
  subplot(411), stem(n,h(1,:),'.'); ylim([-0.3 0.3]),
  title('Ch-ki impulsowe filtrów analizy'); ylabel('h_0(n)');
  subplot(412), stem(n,h(2,:),'.'); ylabel('h_1(n)'), ylim([-0.3 0.3])
  subplot(413), stem(n,h(3,:),'.'); ylabel('h_2(n)'), ylim([-0.3 0.3])
  subplot(414), stem(n,h(M,:),'.'); ylim([-0.3 0.3])
  ylabel('h_{M-1}(n)'); xlabel('n')
    figure
    subplot(211), plot(W/pi,abs(H)); grid, ylabel('|H(\Omega)|');
    title('Ch-ki czestotliwosciowe filtrów analizy'); ylim([0 1.2]),
    subplot(212), plot(W/pi,20*log10(abs(H))); grid, axis([-1 1 -150 20]),
    ylabel('|H(\Omega)| (dB)'); xlabel('\Omega / \pi')
 % ------- Filtry syntezy (rekonstrukcji) -------
  figure
  subplot(411), stem(n,g(1,:),'.m'); ylabel('g_0(n)'), ylim([-0.3 0.3])
  title('Ch-ki impulsowe filtrów syntezy');
  subplot(412), stem(n,g(2,:),'.m'); ylabel('g_1(n)'), ylim([-0.3 0.3])
  subplot(413), stem(n,g(3,:),'.m'); ylabel('g_2(n)'), ylim([-0.3 0.3])
  subplot(414), stem(n,g(M,:),'.m'), ylim([-0.3 0.3])
  ylabel('g_{M-1}(n)'); xlabel('n')
 end;
----------------------------
function [hg,HG,Fd] = bank2(h,g)
% Sumaryczne ch-ki banku filtrów analizy i syntezy
% Input:    h - (MxL) macierz odp. impulsowych filtrów analizy
%           g - (MxL) macierz odp. impulsowych filtrów analizy
% Output:   hg - sumaryczna odp. impulsowa, HG - sumaryczna ch-ka widmowa
%           Fd - przeciek widmowy
% Zielinski, Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań, str.537
% Zniekształcenia nieliniowe i fazowe
M=size(h,1); L=size(h,2);
Nfft=2048;
W = -pi:2*pi/Nfft:pi;
hgk = zeros(M,2*L-1);
nc = 0:2*L-2;
for k=0:M-1,
  hgk(k+1,:) = conv(h(k+1,:),g(k+1,:));
end;
hg = sum(hgk);
hg1=hg; hg1(L)=0;
```

```
HG = freqz(hg,1,W);
% Zniekształcenia aliasingowe
Ffft=W(1:end-1)/pi;
Fd = zeros(1, Nfft);
n = 0:L-1;
for m = 1:M-1,       % po wszystkich 'przeciekach' widmowych
    modZ = exp(j*2*pi*m*n/M);
    A = zeros(1, Nfft);
     for k=0:1:M-1,  % po wszystkich filtrach
         hk = h(k+1,:); gk = g(k+1,:);
         Gk = fft( gk, Nfft );
         Hk = fft( hk.*modZ, Nfft );
         Ak = Gk.*Hk;
         A = A + Ak;
     end
    Fd = Fd + (A.*conj(A));
end
Fd = fftshift(sqrt(Fd));
Fd = 20*log10(Fd);
if nargout==0,  % --------- Wykresy ----------
  figure
  subplot(211), plot(nc,hg); grid; ylabel('splot h(n)*g(n)'),
  title('Sumaryczna odp. impulsowa filtrów analizy i syntezy');
  subplot(212), plot(nc,hg1); grid; xlabel('n'),
  title('Sumaryczna odp. impulsowa bez impulsu środkowego');
 figure
 subplot(311), plot(W/pi, abs(HG)); grid; ylim([0.997 1.003]),
 ylabel('|H(\Omega)|')
 title('Sumaryczna ch-ka amplitudowa i fazowa filtrów analizy i syntezy');
 subplot(312), plot(W/pi, unwrap(angle(HG))); grid;
 ylabel('\phi(\Omega) (rad)')
 subplot(313),plot(Ffft, Fd,'r'); axis([Ffft(1) Ffft(end) -160 -80]), grid;
 title('Suma przecieków widmowych'), ylabel('A_l (dB)')
 xlabel('\Omega / \pi');
end;
```

- Zaobserwować zależność charakterystyki amplitudowej prototypu od tych liczby kanałów banku i poziomu tłumienia w paśmie zaporowym przeprowadzając obliczenia dla `M=10; rs=60;`
- Powtórzyć obliczenia dla początkowych wartości parametrów (w celu przywrócenia `p`).

B. Projektowanie banku filtrów BP analizy (*dekompozycji*) i syntezy (*rekonstrukcji*) sygnału na podstawie prototypu LP zaprojektowanego w pkt. A.

- Wyznaczenie macierzy `h` (wymiar `MxL`) odpowiedzi impulsowych filtrów analizy i `g` - filtrów syntezy (*k*-ty wiersze macierzy jest charakterystyką filtra $h_k(n)$ ($g_k(n)$), k=0,1,...,M-1) metodą modulacji kosinusowej.

```
[h,g] = bank1(M,p);  % projektowanie banku M filtrów BP analizy i syntezy
bank1(M,p);          % wykresy ch-tyk amplitud. i impulsowych filtrów
```

- Przeanalizować skrypt funkcji `bank1.m`. Zwrócić uwagę i wyjaśnić skalowanie energii odpowiedzi impulsowej $p(n)$ prototypu. Znaleźć (na wykresach) symetrię odpowiedzi impulsowych filtrów analizy $h_k(n)$ i syntezy $g_k(n)$. Jaka symetria wiąże te odpowiedzi? Czy są to filtry o liniowej fazie?

C. Przeprowadzić analizę (rozkład na `M` składowych pasmowych) sygnału dźwiękowego za pomocą zaprojektowanych filtrów analizy (*** należy włączyć głośniki komputera ***).

- Przetwarzanym sygnałem będzie szum biały obejmujący cały zakres częstotliwości. Wczytać i odtworzyć sygnał:

```
[x,fs,nbits]=wavread('szum');
sound(x,fs,nbits);      % odtworzenie sygnału na wyjściu audio
disp('Parametry sygnału z pliku .wav')
```

```
L=length(x), fs, nbits  % parametry sygnału w oknie komend
```

- Przeprowadzić dekompozycję sygnału za pomocą zaprojektowanych filtrów BP o charakterystykach impulsowych zapisanych w macierzy h.
- Składowe podpasmowe sygnału $y_k(n)$, k=0,1,...,M-1, są zapamiętywane w wierszach macierzy y.
  Uwaga: Składowe podpasmowe sygnału są odtwarzane na wyjściu audio *po naciśnięciu dowolnego klawisza w aktywnym oknie komend* Matlaba.

```
nw=[2000 2500];     % lupa czasowa (zakres próbek) do rys. wykresów
[y,vary,varx] = decompose(h,x,fs,nw);  % dekompozycja sygnału na składowe
decompose(h,x,fs,nw);% wykresy dwóch pierwszych i ostat. składowej k=0,1..M-1
```

- Uwaga: Sprawdzić, czy plik funkcji decompose.m stosowanej w obliczeniach znajduje się w folderze roboczym. Jeżeli nie, podany poniżej listing funkcji należy skopiować do edytora Matlaba i zapisać w osobnych plikach o podanych nazwach w folderze roboczym.
  Przeanalizować działanie funkcji.

```
function [y,vary,varx] = decompose(h,x,fs,nw)
% Rozkład sygnału x na M podpasm (wiersze y) za pomocą banku filtrów
% analizy, których odp. impulsowe są podane w wierszach macierzy h (MxL).
% Liczenie wariancji vary pasm i wyświetlanie ramki nw=[ni nf].
if nargin<4, ni=500; nf=1000; else  ni=nw(1); nf=nw(2);  end;
nframe=nf-ni+1; % liczba próbek w ramce
x=x';
varx=var(x(ni:nf));
N=length(x);
[M,L]=size(h);
t=[ni:nf]/fs;
y=zeros(M,N);
vary=zeros(M,1);
Y=zeros(M,nframe);
F=-1+2*[0:nframe-1]/nframe;
for n=1:M
  y(n,:)=filter(h(n,:),1,x);                % filtracja całego sygnału
  vary(n)=var(y(n,ni:nf));                  % wariancja ramki
  Y(n,:)=abs(fftshift(fft(y(n,ni:nf))));    % widmo amplitudowe ramki
end;
X=abs(fftshift(fft(x(ni:nf))));
XdB=20*log10(X); YdB=20*log10(Y);
if nargout==0,  % ------ odtworzenie dźwięków i wykresy ---------
  disp('Naciskaj dowolny klawisz, żeby odtworzyć składowe podpasmowe'),
  pause
  sound(x,fs), disp('sygnal x'), pause
  for n=1:M,
    sound(y(n,:),fs), fprintf('podpasmo y(%d)    wariancja var = %7.3e\n',...
        n-1,vary(n)), pause
  end;
  figure
  subplot(411), plot(t,x(ni:nf));grid, axis([t(1) t(end) -1 1]), ylabel('x'),
  title('Przebiegi czasowe sygnału analizowanego i jego składowych');
    subplot(412), plot(t,y(1,ni:nf),'m'); grid, ylabel('u_0');
    xlim([t(1) t(end)])
    subplot(413), plot(t,y(2,ni:nf),'m'); grid, ylabel('u_1');
    xlim([t(1) t(end)])
    subplot(414), plot(t,y(M,ni:nf),'m'); grid, ylabel('u_{M-1}');
    xlim([t(1) t(end)]), xlabel('t (s)')
  figure
  subplot(411), plot(F,XdB); grid, axis([-1 1 -50 50]), ylabel('|X|'),
  title('Widma składowych sygnału analizowanego (dB)');
    subplot(412), plot(F,YdB(1,:),'m'); grid, axis([-1 1 -50 50]),
    ylabel('|U_0|');
```

```
    subplot(413), plot(F,YdB(2,:),'m'); grid, axis([-1 1 -50 50]),
    ylabel('|U_1|');
    subplot(414), plot(F,YdB(M,:),'m'); grid, axis([-1 1 -50 50]),
    ylabel('|U_{M-1}|'); xlabel('\Omega / \pi')
end;
```

- W oknie komend podawane są natężenia (wariancje, zapisywane w wektorze `vary`) poszczególnych składowych wykorzystywane dalej przy kwantowaniu do ustalenia rozdzielczości bitowej podpasm. Zaobserwować na wykresach poziomy i zanotować wariancje składowych w podpasmach. Czy są one zbliżone (tak powinno teoretycznie być w przypadku szumu białego)?

D. Kwantowanie poziomów składowych w podpasmach.

- Określić liczbę bitów R na próbkę sygnału po skwantowaniu (*bitrate* $I=R\cdot f_s$ bitów/s) i wyznaczyć alokację bitów na podpasma (zależną od wariancji `vary`). Liczby bitów R_k przypadające na k-te pasmo są podawane w oknie komend:

```
R=8;                % liczba bitów do rozdziału
Rk=bitalloc(R,vary); % rozdział R bitów na M=length(vary) pasm, Rk w k-tym
```

- Uwaga: Sprawdzić, czy plik funkcji `bitalloc.m` stosowanej w obliczeniach znajduje się w folderze roboczym. Jeżeli nie, podany poniżej listing funkcji należy skopiować do edytora Matlaba i zapisać w osobnych plikach o podanych nazwach w folderze roboczym.
  Przeanalizować działanie funkcji.

```
function Rk = bitalloc(R,vary)
% Uproszczona optymalna alokacja R bitów podpasm o wariancjach w vary
% na Rk-bitowe reprezentacje podpasm
M=length(vary);         % liczba podpasm
Rk=zeros(M,1);          % alokacja początkowa po 0 bitów
sigRk=vary;             % wariancje początkowe
while R>0,              % rozdzielanie bitów do wyczerpania
  [maxsig,k]=max(sigRk);% wyzn. pasma k o maks. wariancji
  Rk(k)=Rk(k)+1;        % przydzielenie bitu do k-tego pasma
  sigRk(k)=sigRk(k)/2;  % 2-krotna redukcja wariancji pasma
  R=R-1;                % zmniejszenie pozostałej do rozdziel. liczby bitów
end;
fprintf('Wyznaczona alokacja R bitów na podpasma:\n'),
for k=1:M,
    fprintf('podpasmo k = %d    liczba bitów Rk = %d\n',k-1,Rk(k)),
end;
disp(' '),
```

E. Przeprowadzić redukcję (*downsampling*) częstotliwości próbkowania składowych pasmowych o czynnik M (powoduje to powielenie widm pasmowych), a następnie kwantowanie zredukowanych składowych pasmowych. Liczba poziomów składowej po skwantowaniu jest przez liczbę bitów przypadającą na dane pasmo:

```
yd=downsample(y',M); % downsampling składowych w podpasmach o czynnik M
yd=yd';              % yd - macierz składowych po downsamplingu
yq=decompquant(yd,fs,M,Rk,nw);    % obliczenia, yq - składowe skwantowane
decompquant(yd,fs,M,Rk,nw);       % wykresy efektów analizy i kwantowania
```

- Uwaga: Sprawdzić, czy plik funkcji `decompquant.m` stosowanej w obliczeniach znajduje się w folderze roboczym. Jeżeli nie, podany poniżej listing funkcji należy skopiować do edytora Matlaba i zapisać w osobnych plikach o podanych nazwach w folderze roboczym.
  Przeanalizować działanie funkcji zwracając uwagę na realizację operacji kwantowania poziomów składowych podpasmowych.

```
function yq = decompquant(yd,fs,r,Rk,nw)
% Kwantowanie do Rk bitów skład. pasm yd (macierz MxNd)z M filtrów analizy
% fs - częstotl. próbkowania (przed downsamplingiem), r - czynnik redukcji
% nw=[ni nf] - zakres rysowania w próbkach (przed downsamplingiem)
if nargin<5, ni=500; nf=1000; else ni=nw(1); nf=nw(2); end;
```

```
ni=floor(ni/r); nf=floor(nf/r);     % zakres próbek po downsamplingu
nframe=nf-ni+1;     % liczba próbek w ramce
M=size(yd,1); Nd=size(yd,2);
yq=zeros(size(yd));
for n=1:M,          % kwantowanie z obliczonym przydziałem bitów
   divid=2.5*sqrt(var(yd(n,:)));    % przedział kwantowania dla pasma n
   y=yd(n,:)/divid;
   if Rk(n)~=0,
     yq(n,:)=round(y*Rk(n))/Rk(n)*divid; % wartości skwantowane
   else
     yq(n,:)=0;
   end;
end;
nfft=400;
Yd=zeros(M,nfft); Yq=Yd;
for n=1:M,
  [Y,F] = dft1(yd(n,ni:ni+nfft-1),nfft);
  Yd(n,:)= abs(Y');             % widmo przed kwantowaniem
  [Y,F] = dft1(yq(n,ni:ni+nfft-1),nfft);
  Yq(n,:)= abs(Y');             % widmo po kwantowaniu
end;
YqdB=20*log10(Yq);  YdB=20*log10(Yd);
nd=ni:nf;
if nargout==0,  % ------ wykresy ---------
  figure        % wykresy schodkowe stairs po skwantowaniu
  subplot(311), plot(nd,yd(1,ni:nf)), hold on, grid,
  stairs(nd,yq(1,ni:nf),'r'); xlim([nd(1) nd(end)])
  ylabel('y_0'), title('Zredukowane i skwantowane składowe podpasmowe');
    subplot(312), plot(nd,yd(2,ni:nf)), hold on, grid,
    stairs(nd,yq(2,ni:nf),'r'); xlim([nd(1) nd(end)]), ylabel('y_1');
  subplot(313), plot(nd,yd(M,ni:nf)), hold on, grid,
  stairs(nd,yq(M,ni:nf),'r'); xlim([nd(1) nd(end)]),
  ylabel('y_{M-1}'); xlabel('n_{down}')
    figure
    subplot(311), plot(F,YdB(1,:), F,YqdB(1,:),'r:'); grid,
    axis([min(F) max(F) -50 50]), ylabel('|Y_0|'),
    title('Widma zredukowanych składowych w paśmie podstawowym (dB)');
      subplot(312), plot(F,YdB(2,:), F,YqdB(2,:),'r:'); grid,
      axis([min(F) max(F) -50 50]), ylabel('|Y_1|');
    subplot(313), plot(F,YdB(M,:), F,YqdB(M,:),'r:'); grid,
    axis([min(F) max(F) -50 50]), ylabel('|Y_{M-1}|');
    xlabel('\Omega_{down} / \pi'), legend('przed kwantowaniem','po
kwantowaniu')
end;
```

- Dlaczego wystarczy przeprowadzenie samej redukcji (`downsample`), a nie pełnej decymacji składowych podpasmowych?
- Sprawdzić, czy liczba poziomów na wykresach składowych skwantowanych zgadza się z liczbą bitów przydzieloną na każde pasmo.
- Porównać widma składowych zarejestrowane przed i po redukcji częstotliwości próbkowania.

3.4. Rekonstrukcja sygnału za pomocą banku filtrów

Uwaga: W tym punkcie wykorzystywane są zmienne dekompozycji sygnału wyliczone w pkt.3.3. Należy sprawdzić, czy istnieją one w pamięci **Workspace** Matlaba.

A. Rekonstrukcja sygnału ze składowych za pomocą zaprojektowanych w pkt.3.3 filtrów syntezy `g:`

```
xr=reconstruct(g,yq,fs,varx,nw);    % obliczenia rekonstrukcji sygnału
reconstruct(g,yq,fs,varx,nw);       % wykresy przebiegów i widm
```

- Uwaga: Sprawdzić, czy plik funkcji `reconstruct.m` stosowanej w obliczeniach znajduje się w folderze roboczym. Jeżeli nie, podany poniżej listing funkcji należy skopiować do edytora Matlaba i zapisać w osobnych plikach o podanych nazwach w folderze roboczym.
  Przeanalizować działanie funkcji. Wariancja `varx` sygnału oryginalnego jest wykorzystywana do uzyskania sygnału zrekonstruowanego $\hat{x}(n)$ (wektor `xr`) o takim samym poziomie. Funkcja dokonuje zwiększenia częstotliwości próbkowania składowych do wartości pierwotnej (*upsampling* razy `M`), filtracji BP za pomocą filtrów `g` i zsumowania wydzielonych składowych zrekonstruowanych.

```
function xr = reconstruct(g,y,fs,varx,nw)
% Upsampling składowych sygnału y i przepuszczenie przez M filtrów BP
% syntezy o odp. impulsowych w macierzy g (MxL)
if nargin<5, ni=500; nf=1000; else ni=nw(1); nf=nw(2); end;
nframe=nf-ni+1;     % liczba próbek w ramce
M=size(y,1);
yu=upsample(y',M);  % upsampling składowych do częstotl. pierwotnej
yu=yu';
N=size(yu,2);
yr=zeros(M,N);
Yr=zeros(M,nf-ni+1);
for n=1:M,
   yr(n,:)=filter(g(n,:),1,yu(n,:));        % filtracja całego sygnału
   Yr(n,:)=abs(fftshift(fft(yr(n,ni:nf)))); % widmo amplitudowe ramki
end;
xr=sum(yr);         % rekonstrukcja sygnału: sumowanie składowych podpasm
varxr=var(xr(ni:nf));
reskal=sqrt(varx/varxr);
yr=yr*reskal;       % skalowanie do wariancji składowych wejściowych
Yr=Yr*reskal;
xr=xr'*reskal;      % skalowanie do pierwotnej wariancji
% xr(xr>1.0)=1.0; xr(xr<-1.0)=-1.0;
Xr=abs(fftshift(fft(xr(ni:nf))));   % widmo amplitud. sygnału zrekonstr.
YdB=20*log10(Yr);  XdB=20*log10(Xr);
F=-1+2*[0:nframe-1]/nframe; % częstotl. unormowane do [-1,1)
t=[ni:nf]/fs;       % ramka czasowa w (s)
if nargout==0, % --------- wykresy -----------
  figure
  subplot(411), plot(t,xr(ni:nf)); grid, axis([t(1) t(end) -1 1]),
  ylabel('x_r = \Sigma x_{ri}'),
  title('Przebiegi sygnału rekonstruowanego i jego składowych');
    subplot(412), plot(t,yr(1,ni:nf),'r'); grid, xlim([t(1) t(end)]),
    ylabel('x_{r0}');
  subplot(413), plot(t,yr(2,ni:nf),'r'); grid, xlim([t(1) t(end)]),
  ylabel('x_{r1}');
    subplot(414), plot(t,yr(M,ni:nf),'r'); grid, xlim([t(1) t(end)]),
    ylabel('x_{r(M-1)}'); xlabel('t')
  figure
  subplot(411), plot(F,XdB); grid, axis([-1 1 -50 50]),
  ylabel('|X_r|'), title('Widma składowych sygnału rekonstruowanego (dB)');
    subplot(412), plot(F,YdB(1,:),'r'); grid, axis([-1 1 -50 50]),
    ylabel('|X_{r0}|');
  subplot(413), plot(F,YdB(2,:),'r'); grid, axis([-1 1 -50 50]),
  ylabel('|X_{r1}|');
    subplot(414), plot(F,YdB(M,:),'r'); grid, axis([-1 1 -50 50]),
    ylabel('|X_{r(M-1)}|'); xlabel('\Omega / \pi')
end;
```

- Porównać ("na oko") przebiegi i widma sygnału odtworzonego $\hat{x}(n)$ i jego składowych $\hat{x}_k(n)$ z otrzymanymi w pkt. 3.3 wykresami sygnału wejściowego $x(n)$ i składowych analizy $y_k(n)$ oraz ich widm.

- Odtworzyć sygnał zrekonstruowany na wyjściu audio (*** sygnał na głośniki ***):

```
sound(xr,fs);
```

B. Porównanie przebiegów sygnału oryginalnego `x` i odtworzonego po dekompozycji i kwantowaniu `xr`.

```
nr=[0 length(x)-1];     % zakres czasowy wykresów
n=nr(1):nr(2);
figure
subplot(211), plot(n, x(n+1)), grid, xlabel('n'), ylabel('x')
title('Sygnał oryginalny x i zrekonstruowany po dekompozycji x_r')
subplot(212), plot(n, xr(n+1)), grid, xlabel('n'), ylabel('x_r')
```

➢ Zwrócić uwagę i wyjaśnić przesunięcie sygnału odtworzonego względem wejściowego (patrz: charakterystyka fazowa w następnym podpunkcie).

- Zaobserwować *sumaryczne* charakterystyki impulsowe oraz widmowe całego toru filtrów analizy i syntezy łącznie oraz tzw. przeciek widmowy związany z przenikaniem części składowej z jednego podpasma do sąsiednich, a wynikający ze skończonego tłumienia filtrów BP w pasmach zaporowych:

```
bank2(h,g);
```

➢ Wyjaśnić obecność impulsu środkowego na wykresie splotu $h(n)*g(n)$ odpowiedzi impulsowych banków filtrów analizy i syntezy. Jaki jest idealny przebieg tej odpowiedzi?

➢ Wyjaśnić przebieg wypadkowej charakterystyki częstotliwościowej $H(\Omega)G(\Omega)$ banku filtrów analizy i syntezy. Jaki jest przebieg idealnej charakterystyki wypadkowej?

3.5. Powtórzenie dekompozycji i rekonstrukcji dla innego sygnału i parametrów

- Powtórzyć obliczenia **od pkt.3.3.C** dla innego sygnału (nie będącego szumem), np.

```
[x,fs,nbits]=wavread('fire');
...
```

➢ Porównać natężenie składowych do dekompozycji w podpasmach, alokację bitów na podpasma oraz przebiegi widm.

➢ Czy słychać różnicę pomiędzy sygnałem oryginalnym i zrekonstruowanym po analizie i skwantowaniu?

4. Zadania do realizacji samodzielnej

Zadanie 4.1. Powtórzyć obliczenia jak w pkt.3.3 i 3.4 dla innych wartości `M` (przeprowadzić analizę dla większej liczby podpasm, np. `M=32` jak w kodowaniu MP3), `L` (np. `L=200`- parzyste, `L=401`), `i rs`. Zaobserwować wpływ zmian na wyniki.

5. Opracowanie sprawozdania

W sprawozdaniu należy zawrzeć zarejestrowane wyniki eksperymentów przeprowadzonych według instrukcji oraz zadanych przez prowadzącego zadań do realizacji samodzielnej z odpowiednimi opisami oraz wyjaśnieniami opartymi na wiedzy z wykładu i literatury, ze szczególnym uwzględnieniem problemów wskazanych symbolem ➢.

Literatura

1. Milic L.: *Multirate Filtering for Digital Signal Processing. Matlab Applications*, Information Science Ref., 2008
2. Ingle V.K., Proakis J.G.: *Essentials of Digital Signal Processing Using Matlab*, 3 ed., Cengage Learning, 2012.
3. Mitra S.K.: *Digital Signal Processing. A Computer Based Approach*, 4 ed., McGraw-Hill, 2011.
4. Leis J.W.: *Digital Signal Processing Using Matlab for Students and Researchers*, John Wiley & Sons, 2011.
5. Orfanidis S.J.: *Introduction to Signal Processing*, Prentice Hall, 2010.
6. Lyons R.G.: *Wprowadzenie do cyfrowego przetwarzania sygnałów*, WKŁ, 1999.
7. Zieliński T.: *Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań*, WKiŁ, 2005.
8. Smith S.W.: *Cyfrowe przetwarzanie sygnałów. Praktyczny poradnik dla inżynierów i naukowców*, Wyd. BTC, 2007.

Opracował: Dr inż. Janusz Baran Częstochowa, 1999-2013