Laboratorium Techniki Obliczeniowej i Symulacyjnej

Ćwiczenie 9. Implementacja algorytmów cyfrowego przetwarzania sygnałów.

Opracował: dr inż. Sebastian Dudzik

1. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z możliwościami implementacji algorytmów cyfrowego przetwarzania sygnałów w programie MATLAB, w szczególności do obrazowania widma sygnału próbkowanego i projektowania filtrów cyfrowych

2. Wprowadzenie

2.1. Analiza widmowa dyskretnych sygnałów deterministycznych

Pod pojęciem sygnał dyskretny x będziemy rozumieli ciąg liczbowy $x(n)$, którego elementami są próbki $x(n)$ sygnału ciągłego (analogowego) pobierane w chwilach $t_n = nT_s$, gdzie T_s jest okresem próbkowania. Indeks n oznacza dyskretny czas t_n unormowany względem okresu próbkowania: $n = t_n/T_s$.

Transformatą Fouriera (widmem Fourierowskim $DTFT$ – Discrete Time Fourier Transform) sygnału dyskretnego w czasie $x(n)$ i o ograniczonej energii, nazywamy funkcję:

$$DTFT[x(n)] = X(e^{-j\Omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\Omega n}, \tag{1}$$

gdzie:

$$\Omega = \omega T_s = 2\pi f/f_s, \tag{2}$$

jest pulsacją unormowaną względem częstotliwości próbkowania $f_s = 1/T_s$.

W praktyce obserwuje się (rejestruje) jedynie pewien fragment $x_0(n)$ sygnału $x(n)$ i dlatego obliczenie widma (1) wymagającego nieskończonego ciągu próbek nie jest możliwe (poza tym tylko dla ograniczonej klasy sygnałów suma (1) jest zbieżna). Fragment x_0 można interpretować jako nałożenie na $x(n)$ okna czasowego (prostokątnego impulsu o jednostkowej amplitudzie) $w(n)$ o długości N. Stosując cyfrowe metody obliczeń transformatę (1) można obliczyć tylko dla skończonego zbioru dyskretnych wartości Ω_m. Ponieważ $X_0(e^{j\Omega})$ jest funkcją okresową wystarczy obliczać widmo tylko dla przedziału $[0, 2\pi)$ lub $[-\pi, \pi)$. Przyjmując, że obliczamy N wartości widma (tyle ile jest próbek sygnału) dla równomiernie rozłożonych pulsacji unormowanych:

$$\Omega_m = \frac{2\pi m}{N},\ N = 0, 1, \ldots, N-1, \tag{3}$$

można przedstawić N-punktowe, dyskretne przekształcenie Fouriera (DFT), w postaci następującej zależności:

$$DFT[x_0(n)] = X_0(m) = \sum_{n=0}^{N-1} x_0(n) e^{-j\frac{2\pi nm}{N}},\ N = 0, 1, \ldots, N-1. \tag{4}$$

Obliczanie $DFT[x_0(n)]$ zamiast widma ciągłego $DTFT[x(n)]$ wnosi zniekształcenia wynikające z:

- brania do obliczeń jedynie skończonej liczby próbek (nakładania okna czasowego),
- obliczania dyskretnego zbioru wartości transformaty (próbkowania widma ciągłego).

Parę transformat DFT i IDFT implementuje się numerycznie stosując szybkie algorytmy FFT (Fast Fourier Transform). DFT jest głównym narzędziem analizy widmowej sygnałów dyskretnych.

2.2. Projektowanie filtrów cyfrowych o nieskończonej odpowiedzi impulsowej

W większości przypadków, filtracja polega na usunięciu z sygnału wchodzącego do układu szumów i innych zakłóceń i wyodrębnieniu sygnału użytecznego przy zachowaniu dopuszczalnego poziomu zniekształceń. Innym rodzajem filtracji jest celowe przekształcenie danego sygnału w inny. Przykładem mogą być np. filtry różniczkujące, filtry Hilberta lub banki filtrów pasmowych służące do dekompozycji sygnału (np. w celu jego kompresji). W obu przypadkach dochodzi się do problemu syntezy filtra cyfrowego na podstawie narzuconych wymagań projektowych. Wymagania projektowe do zadania projektowania filtra określa się najczęściej w dziedzinie częstotliwości. W przypadku kiedy znane jest pasmo sygnału użytecznego, a widmo zakłóceń leży poza tym pasmem, filtr powinien mieć stałą (np. rowną 1) charakterystykę amplitudową i liniową charakterystykę fazową w tym paśmie (paśmie przepustowym). Poza pasmem użytecznym charakterystyka amplitudowa powinna mieć wartości bliskie zeru.

Filtry o nieskończonej odpowiedzi impulsowej (NOI lub IIR – infinite impulse response) mają transmitancję wymierną postaci:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_N z^{-N}}{a_0 + a_1 z^{-1} + \ldots + a_N z^{-N}}, \tag{5}$$

gdzie: z — zmienna zespolona.

Metody projektowania filtrów cyfrowych stanowią oddzielną dziedzinę wiedzy, której zakres wykracza w znaczącym stopniu poza zakres wiadomości prezentowanych w niniejszej instrukcji. Z tego powodu w ćwiczeniu zostaną przedstawione jedynie dwie metody projektowe, nakładające wymagania na charakterystykę amplitudową filtra:

- metoda dyskretyzacji prototypów analogowych,
- metody optymalizacyjne.

Metoda dyskretyzacji prototypów analogowych umożliwia projektowanie filtrów o standardowych charakterystykach LP (dolnoprzepustowych), HP (górnoprzepustowych), BS (pasmowo zaporowych) oraz BP (pasmowo przepustowych) i polega na zaprojektowaniu najpierw filtra analogowego o transmitancji ciągłej $H_a(s)$ spełniającego postawione wymagania, a następnie przekształceniu $H_a(s)$ do docelowej postaci dyskretnej $H(z)$. Zaletą tej metody są stosunkowo proste, bezpośrednie formuły projektowe, a wadą brak możliwości projektowania filtrów o dowolnym przebiegu charakterystyki amplitudowej.

Jako prototypy analogowe wykorzystuje się kilka typów filtrów różniących się właściwościami:

- filtr Butterwortha (o maksymalnie płaskiej charakterystyce amplitudowej w paśmie przepustowym),
- filtr Czebyszewa I rodzaju (o charakterystyce równomiernie pofalowanej w paśmie przepustowym i monotonicznej w paśmie zaporowym),
- filtr Czebyszewa II rodzaju (o charakterystyce równomiernie pofalowanej w paśmie zaporowym i monotonicznej w paśmie przepustowym),
- filtr eliptyczny (Cauera, o charakterystyce równomiernie pofalowanej w obu pasmach i najwęższym (najbardziej stromym — dla określonego rzędu N) paśmie przejściowym,
- filtr Bessela (wersja analogowa ma prawie stałą wartość opóźnienia grupowego w paśmie przepustowym i monotoniczną charakterystykę amplitudową).

Przy projektowaniu filtra trzeba osiągnąć kompromis pomiędzy stromością pasma przejściowego a liniowością charakterystyki fazowej, która w wielu zastosowaniach, m.in. w technice audio, może mieć istotne znaczenie. Rozwiązania krańcowe to filtr eliptyczny oraz filtr Bessela, natomiast filtr Butterwortha ma właściwości pośrednie.

Zaprojektowany prototyp analogowy przekształca się do postaci dyskretnej korzystając z określonej metody dyskretyzacji. Najczęściej stosuje się:

- metodę odwzorowania biliniowego płaszczyzny s na płaszczyznę z, określoną podstawieniem:
$$H(z) = H_a(s)|_{s=\frac{2}{T_s}\frac{1-z^{-1}}{1+z^{-1}}}, \tag{6}$$
gdzie $H_a(s)$ jest transmitancją Laplace'a analogowego prototypu filtra.
- metodę niezmienniczej odpowiedzi impulsowej. Jest to przekształcenie opierające się na warunku takich samych wartości odpowiedzi impulsowej prototypu analogowego i filtra dyskretnego w momentach próbkowania:
$$h_d(n) = h_a(t)|_{t=nT_s} \tag{7}$$
Procedura polega na rozłożeniu $H_a(s)$ na ułamki proste i przekształceniu biegunów ciągłych na dyskretne według zależności $z = e^{sT_s}$.

Metody optymalizacyjne umożliwiają projektowanie filtrów o dowolnym (kawałkami liniowym) kształcie charakterystyki amplitudowej, której wymagany przebieg jest określony na zbiorze pulsacji dyskretnych $\Omega_i : i = 1, 2, \ldots, L$ z przedziału $[0, \pi]$ (współrzędne Ω_i określają zwykle punkty załamania charakterystyki, czyli granice pasm przepustowych i zaporowych). Jako wskaźnik jakości optymalizacji definiuje się błąd aproksymacji. Przy projektowaniu filtrów NOI jest to najczęściej uogólniony błąd średniokwadratowy:

$$E_{sk}(\Omega_i) = \sum_{i=1}^{L} W(\Omega_i)(|H_d(e^{-j\Omega_i})| - |H(e^{-j\Omega_i})|)^2, \tag{8}$$

gdzie $W(\Omega)$ to wagi przypisane poszczególnym punktom.

Dla ustalonego rzędu filtra współczynniki wielomianów $B(z)$ i $A(z)$ transmitancji dobiera się w taki sposób, aby błąd (8) był minimalny. Ostatnim etapem jest numeryczna implementacja zaprojektowanej transmitancji filtra NOI. W celu uzyskania mniejszej wrażliwości na błędy numeryczne stosuje się np. kaskadowe (szeregowe) lub równoległe połączenie tzw. sekcji bikwadratowych, które stanowią filtry NOI drugiego rzędu.

3. Program ćwiczenia

1. Uruchomienie programu MATLAB.
   W ćwiczeniu wykorzystano program MATLAB w wersji 5.3 (R11.1). Uruchomienie programu następuje poprzez skrót na pulpicie (Matlab5.3) lub bezpośrednio z katalogu *C:\MatlabR11\ bin*.

2. Uruchomienie programu Wordpad.exe.
   Program można uruchomić poprzez wywołanie: *Start\Programy\Akcesoria\ Wordpad* lub poprzez skrót na pulpicie.

3. Przejście do katalogu roboczego dla grupy laboratoryjnej.
   Domyślnym katalogiem startowym (roboczym) programu MATLAB jest *C:\Matlab R11\ work*. Zadanie polega na przejściu do podkatalogu katalogu *work*. Podkatalog (utworzony na pierwszych zajęciach laboratoryjnych) nazwany jest wybranymi 2 nazwiskami studentów, wchodzących w skład grupy laboratoryjnej.

   (a) Wprowadzić:
   `>>pwd`
   W programie MATLAB każde wprowadzone polecenie zatwierdza się klawiszem `<ENTER>`. Zwrócić uwagę na ścieżkę dostępu do katalogu bieżącego.

   (b) Wprowadzić:
   `>>cd` *`nazwa_podkatalogu`*
   Parametr *nazwa_pod-katalogu* powinien składać się z nazwisk 2 wybranych studentów grupy laboratoryjnej (np. `>>cd KowalskiNowak`).

4. Analiza widm DTFT sygnałów dyskretnych o skończonej długości

(a) Napisać w języku MATLAB funkcję wykładniczą postaci $x(t) = e^{-at}$, próbkowaną co T_s sekund. W edytorze programu MATLAB wprowadzić:

```
function c9_f_exp(a, Ts)
%Sygnały wykładnicze
%                  -at                      n           -a*Ts
% - ciągły xa(t)=e   ; - dyskretny xd(n)=b , gdzie b=e
% Parametry:
%   a   - biegun transformaty sygnału analogowego
%   Ts  - okres próbkowania

L=300; % liczba punktów na wykresie
dt=4/a/(L-1);
t=0:dt:4/a;  % czas ciągły (zakres do 4*stała czasowa)

b=exp(-a*Ts);
N=round(4/a/Ts);
n=0:N;   % czas dyskretny

plot(t, exp(-a*t))
hold on
stem(n*Ts, b.^n,'m:')
title('Sygnał wykładniczy ciągły i dyskretny')
xlabel('Czas t')
```

Zapisać funkcję w pliku c9_f_exp.

(b) Uruchomić funkcję dla $a = 1$ i $T_s = 0.5$. Skopiować wykres do programu Wordpad. Skopiować zawartość okna poleceń programu MATLAB oraz zawartość okna edytora do programu Wordpad. Wyczyścić zawartość okna poleceń programu MATLAB.

(c) Utworzyć w języku MATLAB funkcję prezentującą widmo $DTFT$ dyskretnego sygnału wykładniczego postaci $x(t) = b^n$, gdzie: $b = e^{-aT_s}$ dla zadanego okresu próbkowania. W edytorze programu MATLAB wprowadzić:

(d)
```
function c9_f_exp_widmo(a, Ts)
% Porównanie widma sygnału dyskretnego (N->inf)
% z widmem sygnału analogowego w zakresie od [0, ffinal] w [Hz].
% Amplitudy widm unormowane do 1 dla f=0.
% Parametry:
%   a - biegun transformaty sygnału analogowego
%   Ts - okres próbkowania

L=512; % liczba punktów na wykresie
ffinal=2; % końcowa wartość częstotliwości [Hz]
```

```
wfinal=2*pi*ffinal;  % przeliczenie do rad/s

w=-wfinal:wfinal/(L-1):wfinal;% pulsacja w rad/s
p=exp(-a*Ts);  % biegun transformaty dyskretnej

[Ha, Wa] = freqs(a, [1 a], w); % Xa(jw)
[Hd, Wd] = freqz(1-p, [1 -p], Ts*w); % Xd(exp(jW))
Wd=Wd/Ts/(2*pi);% częstotliwość w rad/s

subplot(2,1,1)
plot(Wa/(2*pi), abs(Ha), Wd, abs(Hd), ':')
hold on;
xlabel('Czestotliwosc [Hz]');
ylabel('Amplituda');
title('Amplituda unormowana do 1')

subplot(2,1,2)
plot(Wa/(2*pi), angle(Ha)*180/pi, Wd,...
angle(Hd)*180/pi, ':')
hold on;
ylabel('Faza [stopnie]')
```

Zapisać funkcję w pliku c9_f_exp_widmo.

(e) Uruchomić funkcję dla $a = 1$ i $T_s = 0.2, 0.5, 1$. Skopiować wykresy do programu Wordpad. Skopiować zawartość okna poleceń programu MATLAB oraz zawartość okna edytora do programu Wordpad. Wyczyścić zawartość okna poleceń programu MATLAB. Zaobserwować zmiany okresowości widma sygnału dyskretnego i związek tego okresu z T_s, przeliczyć skalę do częstotliwości unormowanej $f_n = f/f_s$.W jakim zakresie częstotliwości wystarczy obserwować widmo w skali unormowanej?

(f) Utworzyć w języku MATLAB funkcję prezentującą widma amplitudowe $DTFT$ dla częstotliwości unormowanej $f_n = f_s/f$ z zakresu (-0.5,0.5], dla różnych długości sygnału stałego postaci:

$$x(n) = \begin{cases} 1 & \text{dla} \quad 0 \leq n \leq N-1 \\ 0 & \text{dla} \quad n \geq N \end{cases} \tag{9}$$

Transformata $DTFT$ sygnału, powinna być obliczana jako L–punktowe DFT, gdzie $L >> N$, z uzupełnianiem zerami. W edytorze programu MATLAB wprowadzić:

```
function [Xs,fn] = c9_unit(N)
% Porównanie widma sygnału dyskretnego jednostkowego w zależności
% od czasu trwania sygnału (liczby N niezerowych próbek)
% Sygnał dyskretny jednostkowy xd(n)=1 dla 0<=n<=N-1
```

```
% Wywołanie: [Xs,fn] = c9_unit(N);
% Parametry: N - liczba próbek

L=512; % liczba pktów na wykresie

K=0:(N-1);
xs=zeros(1,L); xs(1:N)=ones(1,N);

[Xs,Wn]=dtft(xs); % widmo sygnału obciętego (L-N próbek=0), -pi<Wn<pi

fn=Wn/(2*pi); % częstotl. unormowana

plot(fn, abs(Xs), ':')
xlabel('Czestotliwosc unormowana fn=f/fs');
ylabel('Amplituda widma');
hold on
```

Zapisać funkcję w pliku c9_unit.

(g) Uruchomić funkcję dla $N = 5, 10$ Skopiować wykresy do programu Wordpad. Skopiować zawartość okna poleceń programu MATLAB oraz zawartość okna edytora do programu Wordpad. Wyczyścić zawartość okna poleceń programu MATLAB.

(h) Utworzyć w języku MATLAB funkcję prezentującą widma amplitudowe $DTFT$ dla częstotliwości unormowanej $f_n = f_s/f$ z zakresu (-0.5,0.5], dla różnych długości sygnału wykładniczego postaci:

$$x(n) = \begin{cases} b^n & \text{dla} \quad 0 \leq n \leq N-1 \\ 0 & \text{dla} \quad n \geq N \end{cases} \tag{10}$$

Transformata $DTFT$ sygnału, powinna być obliczana jako L–punktowe DFT, gdzie $L >> N$, z uzupełnianiem zerami. W edytorze programu MATLAB wprowadzić:

```
function [Xs,fn] = c9_exp(N)
% Porównanie widma dyskretnego sygnału nieskończonego i obciętego
%                            n
% Sygnał dyskretny xd(n)=b  dla 0 <= n <= N-1
% Wywołanie: [Xs,fn] = c9_exp(N);
% Parametry: N - liczba próbek

L=512;
b=0.9;

W=2*pi*(0:(L-1))/L;
f=W/(2*pi);
```

```
[Hd, Wd] = freqz(1, [1 -b], W);
Hd=fftshift(Hd);
Wd=swap(Wd);

K=0:(N-1);
xs=zeros(1,L); xs(1:N)=b.^K;

[Xs,Wn]=dtft(xs); % widmo sygnału obciętego (L-N próbek =0), -pi<Wn<pi

plot(Wd/(2*pi) ,abs(Hd), Wn/(2*pi), abs(Xs), ':')
xlabel('Czestotliwosc unormowana fn=f/fs');
ylabel('Amplituda widma');
hold on
```

Zapisać funkcję w pliku c9_exp.

(i) Uruchomić funkcję dla $N = 5, 10$ Skopiować wykresy do programu Wordpad. Skopiować zawartość okna poleceń programu MATLAB oraz zawartość okna edytora do programu Wordpad. Wyczyścić zawartość okna poleceń programu MATLAB.

(j) Utworzyć w języku MATLAB funkcję prezentującą widma amplitudowe $DTFT$ dla częstotliwości unormowanej $f_n = f_s/f$ z zakresu (-0.5,0.5], dla różnych długości sygnału kosinusoidalnego postaci:

$$x(n) = \begin{cases} \cos(n\Omega_0) & \text{dla} \quad 0 \leq n \leq N-1 \\ 0 & \text{dla} \quad n \geq N \end{cases} \tag{11}$$

Transformata $DTFT$ sygnału, powinna być obliczana jako L–punktowe DFT, gdzie $L >> N$, z uzupełnianiem zerami. W edytorze programu MATLAB wprowadzić:

```
function [Xs,f] = c9_cos(N, f0)
% Wywo-anie: [Xs,fn] = c9_cos(N, f0)
% Parametry:
%   N - liczba próbek
%   f0 - częstotliwość unormowana
L=512;
K=0:(N-1);
xs=zeros(1,L); xs(1:N)=cos(K*2*pi*f0);
[Xs,Wn]=dtft(xs);
plot(Wn/(2*pi), abs(Xs), ':')
xlabel('Czestotliwosc unormowana fn=f/fs');
ylabel('Amplituda widma');
title('Sygna- cosinusoidalny x(n)=cos(2*pi*f0*n)')
```

```
hold on
```

Zapisać funkcję w pliku c9_cos.

(k) Uruchomić funkcję dla $N = 5, 10, 20$. Skopiować wykresy do programu Wordpad. Skopiować zawartość okna poleceń programu MATLAB oraz zawartość okna edytora do programu Wordpad. Wyczyścić zawartość okna poleceń programu MATLAB.

5. DFT sygnału okresowego

(a) Przeprowadzić analizę DFT nieskończonego sygnału okresowego:

$$x(n) = \cos(n\Omega_0), \tag{12}$$

o pulsacji $\Omega_n = 2\pi/K$. Przyjąć $K = 32$ i wyznaczyć N-punktową DFT sygnału (12) na podstawie wycinka N=64 próbek (dokładnie 2 okresy $x(n)$). W edytorze programu MATLAB wprowadzić:

```
function y = c9_gcos(f0, N)
% Generowanie N próbek cosinusoidy o częstotliwościi unormowanej f0.
% y(n) = cos(2*pi*f0*n) dla 0 <= n <= N-1
% Wywołanie: y = c1_gcos(f0, N)
% Parametry:
%   f0 - częstotliwość unormowana (f0=1 -> częst. próbkowania fs)
%   N - liczba próbek
y = cos((0:N-1)*2*pi*f0);
if nargout==0
  plot(0:N-1, y, ':', 0:N-1, y,'o');
  xlabel('n');
  ylabel('cos(n*2*pi*f0)');
end
```

Zapisać funkcję w pliku c9_gcos. W edytorze programu MATLAB wprowadzić:

```
function [X,f] = c9_dft(x,N,c,Ts)
% N-punktowa DFT[x] z uzupełnianiem ( N-length(x) ) zerami
% Wywołanie: [X] = c9_dft(x,N,c,Ts)
% Parametry:
%   x - próbki sygnału
%   N - liczba punktów transformaty
%   c - (optional) c=1 -> rysuj DTFT[x], c=0 -> tylko DFT[x]
%   Ts - (optional) domyślnie Ts=1

if nargin==3 Ts=1;
elseif nargin==2 c=0; Ts=1;
end
```

```
L=512;
X=fft(x,N);
Xc=fft(x,L);
Wn=2*pi*(0:(N-1))/N; % pulsacja unormowana [0, 2*pi)
Wc=2*pi*(0:(L-1))/L;
f=Wn/(2*pi)/Ts;
fc=Wc/(2*pi)/Ts;

if c
 plot(fc, abs(Xc),':')
 hold on
end
stem(f, abs(X),'r:')
xlabel('f');
ylabel('|DFT(x)|');
hold on
```

Zapisać funkcję w pliku c9_dft. W linii poleceń programu MATLAB wprowadzić:

```
N=64;
x= c9_gcos(1/32,N);
X=c9_dft(x,N);
```

(b) Wyznaczyć 1 okres odtworzenia sygnału (12) stosując odwrotne, dyskretne przekształcenie Fouriera ($IDFT$). W edytorze programu MATLAB wprowadzić:

```
function xx = c9_idft(X, N, Ts)
% N-punktowa odwrotna DFT[x]
% Wywołanie: xx = c9_idft(X, N, Ts)
% Parametry:
%   X - współczynniki DFT
%   N - liczba punktów transformaty
%   Ts
if nargin==2 Ts=1; end
xx=ifft(X,N);
tt=(0:(N-1))*Ts; % przeliczenie skali czasu
stem(tt, xx,':')
xlabel('Czas n*Ts');
ylabel('xx');
title('Odwrotna DFT');
hold on
```

Zapisać funkcję w pliku c9_idft. W linii poleceń programu MATLAB wprowadzić:

```
x1=c9_idft(X,N);
```

6. Projektowanie filtrów NOI (IIR) metodą prototypów analogowych

   (a) Zaprojektować filtry dolnoprzepustowe (LP) 3-rzędu o następujących parametrach: pulsacja nominalna (w odniesieniu do pulsacji Nyquista $\Omega_{Nyq} = \pi$) $\Omega_n = 0.1\Omega_{Nyq}$, maksymalne tłumienie w paśmie przepustowym $r_p = 1$dB, minimalne tłumienie w paśmie zaporowym $r_s = 20$dB. Dyskretyzację przeprowadzić metodą odwzorowania biliniowego dla filtrów typu: Butterwortha, Czebyszewa I rodzaju, Czebyszewa II rodzaju, eliptycznego (Cauera). W edytorze programu MATLAB wprowadzić funkcję:

```
function [B,A]=c9_butt(N,Wn,typ,lin)
% Charakterystyki częstotliwościowe filtru Butterwortha
%   H(z)=B(z)/A(z) (wielomiany w ujemnych potęgach z)
% Wywołanie: [B,A]=c9_butt(N,Wn)
% Parametry:
%   N - rząd filtra (stopień wielomianów transmitancji)
%   Wn - pulsacja nominalna odniesiona do pulsacji Nyquista (fNyq=1)
%   typ - typ filtra 'low' lub 'high'

disp('Filtr Butterwortha')
if nargin==3,
   [B,A]=butter(N,Wn,typ) % współczynniki wielomianów B(z) i A(z),
else
   [B,A]=butter(N,Wn)
end

W=logspace(-1,1,400); % wektor pulsacji, dla których obliczane jest H(W)
[H,wh]=freqz(B,A,W); % wyznaczenie odpowiedzi częstotliwościowej filtra

hold on;
subplot(211)
semilogx(wh/pi,abs(H))
axis([0,2,0,1]);
ylabel('|H(z)|');

hold on;
subplot(212)
semilogx(wh/pi,unwrap(angle(H))*180/pi)
axis([0,2,-360,0]);
xlabel('f/fNyq');
ylabel('Faza')
```

   Zapisać funkcję w pliku c9_butt. W edytorze programu MATLAB wprowadzić funkcję:

```
function [B,A]=c9_cheb1(N,rp,Wn)
% Charakterystyki filtru Czebyszewa I rodzaju
%   H(z)=B(z)/A(z) (wielomiany w ujemnych potęgach z)
% Wywołanie: [B,A]=c9_cheb1(N,rp,Wn)
% Parametry:
%   N - rząd filtra (stopień wielomianów transmitancji)
%   rp - max tłumienie (rippling) w paśmie przepustowym w dB
%   Wn - pulsacja nominalna odniesiona do pulsacji Nyquista (fNyq=1)

disp('Filtr Czebyszewa I rodzaju')
[B,A]=cheby1(N,rp,Wn) % współczynniki wielomianów B(z) i A(z)

W=logspace(-1,1,400); % wektor pulsacji, dla których obliczane jest H(W)
[H,wh]=freqz(B,A,W); % wyzanczenie odpowiedzi częstotliwościowej filtra

hold on;
subplot(211)
semilogx(wh/pi,abs(H),'--r')
axis([0,2,0,1]);
ylabel('|H(z)|');

hold on;
subplot(212)
semilogx(wh/pi,unwrap(angle(H))*180/pi,'--r')
axis([0,2,-360,0]);
xlabel('f/fNyq');
ylabel('Faza')
```

Zapisać funkcję w pliku c9_cheb1. W edytorze programu MATLAB wprowadzić funkcję:

```
function [B,A]=c9_cheb2(N,rs,Wn)
% Charakterystyki częstotliwościowe filtru Czebyszewa II rodzaju
%   H(z)=B(z)/A(z) (wielomiany w ujemnych potęgach z)
% Wywołanie: [B,A]=c9_cheb2(N,rs,Wn)
% Parametry:
%   N - rząd filtra (stopień wielomianów transmitancji)
%   rs - min tłumienie (rippling) w paśmie zaporowym w dB
%   Wn - pulsacja nominalna odniesiona do pulsacji Nyquista (fNyq=1)

disp('Filtr Czebyszewa II rodzaju')
[B,A]=cheby2(N,rs,Wn) % współczynniki wielomianów B(z) i A(z)

W=logspace(-1,1,400); % wektor pulsacji, dla których obliczane jest H(W)
[H,wh]=freqz(B,A,W); % wyzanczenie odpowiedzi częstotliwościowej filtra
```

```
hold on;
subplot(211)
semilogx(wh/pi,abs(H),':m')
axis([0,2,0,1]);
ylabel('|H(z)|');

hold on;
subplot(212)
faza=unwrap(angle(H))*180/pi;
semilogx(wh/pi,faza,':m')
axis([0,2,-360,max(faza)]);
xlabel('f/fNyq');
ylabel('Faza')
```

Zapisać funkcję w pliku c9_cheb2. W edytorze programu MATLAB wprowadzić funkcję:

```
function [B,A]=c9_ellip(N,rp,rs,Wn)
% Charakterystyki częstotliwościowe filtru eliptycznego
%   H(z)=B(z)/A(z) (wielomiany w ujemnych potęgach z)
% Wywołanie: [B,A]=c9_ellip(N,rp,rs,Wn)
% Parametry:
%   N - rząd filtra (stopień wielomianów transmitancji)
%   rp - max tłumienie (rippling) w paśmie przepustowym w dB
%   rs - min tłumienie (rippling) w paśmie zaporowym w dB
%   Wn - pulsacja nominalna odniesiona do pulsacji Nyquista (fNyq=1)

disp('Filtr eliptyczny')
[B,A]=ellip(N,rp,rs,Wn) % współczynniki wielomianów B(z) i A(z)

W=logspace(-1,1,400); % wektor pulsacji, dla których obliczane jest H(W)
[H,wh]=freqz(B,A,W); % wyzanczenie odpowiedzi częstotliwościowej filtra

hold on;
subplot(211)
semilogx(wh/pi,abs(H),'c')
axis([0,2,0,1]);
ylabel('|H(z)|');

hold on;
subplot(212)
faza=unwrap(angle(H))*180/pi;
semilogx(wh/pi,faza,'c')
axis([0,2,-360,max(faza)]);
```

```
xlabel('f/fNyq');
ylabel('Faza')
```

Zapisać funkcję w pliku c9_cheb2. W linii poleceń programu MATLAB wprowadzić:

```
N=3; Wn=0.1; rp=1; rs=20;
c9_butt(N,Wn); % filtr Butterwortha
c9_cheb1(N,rp,Wn); % filtr Czebyszewa I rodzaju
c9_cheb2(N,rs,Wn); % filtr Czebyszewa II rodzaju
c9_ellip(N,rp,rs,Wn); % filtr eliptyczny (Cauera)
```

(b) Nałożyć na siebie na jednym wykresie charakterystyki dla filtrów kilku rodzajów i porównać ich przebiegi. W edytorze programu MATLAB wprowadzić skrypt:

```
% Porównanie logarytmicznych charakterystyk amplitudowych filtrów różnego typu,
% ale tego samego rzędu. Parametry: N, rp, rs, Wn muszą być w przestrzeni
% roboczej

W=logspace(-1,1,400); % wektor pulsacji, dla których obliczane jest H(W)

[B,A]=butter(N,Wn);
[Hb,wh]=freqz(B,A,W);

[B,A]=cheby1(N,rp,Wn);
[Hc1,wh]=freqz(B,A,W);

[B,A]=cheby2(N,rs,Wn);
[Hc2,wh]=freqz(B,A,W);

[B,A]=ellip(N,rp,rs,Wn);
[He,wh]=freqz(B,A,W);

f=wh/pi;

figure
semilogx(f,20*log10(abs(Hb)),f,20*log10(abs(Hc1)),...
'--r',f,20*log10(abs(Hc2)),':m',f,20*log10(abs(He)),'c')
axis([0,2,-40,0]);
ylabel('20*log|H(z)|');
xlabel('f/fNyq');
```

Zapisać skrypt w pliku c9_fdb. W linii poleceń programu MATLAB wprowadzić:

```
c9_fdb
```

4. Opracowanie sprawozdania

W sprawozdaniu należy umieścić polecenia oraz wyniki ich działania skopiowane w trakcie ćwiczenia z okna środowiska MATLAB. Do każdej linii kodu oraz do każdego wyniku, należy dodać komentarz objaśniający.

Przykład.
. . . `2+round(6/9+3*2)/2-3` — obliczenie wartości wyrażenia. Funkcja `round(6/9+3*2)` zaokrągla wynik działania `6/9+3*2` do najbliższej liczby całkowitej. . .