Analiza sygnałów biomedycznych - laboratorium

dr inż. Krzysztof Duda

Kraków 2009

Wstęp

Niniejszy skrypt został napisany dla specjalności *Pomiary technologiczne i biomedyczne* studiów II stopnia na kierunku *Elektrotechnika*. Laboratorium z przedmiotu *Analiza sygnałów biomedycznych* trwa jeden semestr. Zajęcia odbywają się w laboratorium komputerowym i polegają na samodzielnym programowaniu algorytmów analizy i przetwarzania sygnałów w środowisku Matlab. W trakcie zajęć, pisane przez studentów programy są na bieżąco sprawdzane i konsultowane przez prowadzącego.

Głównym celem laboratorium jest praktyczna ilustracja zagadnień teoretycznych omawianych na wykładzie z tego przedmiotu. Studenci zdobywają także umiejętność programowania typowych algorytmów analizy i przetwarzania sygnałów.

Rozwiązywanie zawartych w niniejszym skrypcie zadań wymaga znajomości podstaw teoretycznych omówionych w skrypcie do wykładu *Analiza sygnałów biomedycznych*.

W przypadku niektórych zadań umieszczono rozwiązania w postaci programów bądź funkcji Matlaba. Podane programy i funkcje należy traktować jako standardy programowania i w miarę możliwości modyfikować je do rozwiązywania zbliżonych problemów.

Kraków 2009

Tematy ćwiczeń laboratoryjnych

1. Generowanie sygnałów dyskretnych

Generowanie sygnałów cyfrowych opisanych analitycznie za pomocą funkcji np. $x[n]=A\sin(\omega n+\varphi)$ polega na wyliczeniu wartości tych funkcji dla zadanych argumentów.

Zadanie 1.1

Wygenerować sygnał $x[n]=A\sin(\omega n+\varphi)$ o długości $N=32$ próbki. Narysować ten sygnał, opisać osie OX i OY na wykresie. Zaobserwować przebiegi sygnału dla zmian ω z zakresu od 0 do 2π.

Rozwiązanie

Program 1.1 generuje ciąg sinusoidalny. Pierwsze trzy linie kodu służą inicjalizacji środowiska. W liniach 5-8 zebrane są występujące w programie parametry. Następnie w programie występuje blok obliczeń i blok prezentacji wyników. W miarę możliwości należy zawsze stosować powyższą strukturę programu tj.: parametry, obliczenia i wizualizacja.

Program 1.1

```
1 close all       %zamknięcie okien graficznych
2 clear all       %usunięcie zmiennych z pomięci
3 clc             %wyczyszczenie okna konsoli
4 %% parametry
5 Nx=32;          %długość sygnału x (liczba próbek)
6 w=pi/4;         %częstotliwość sygnału [rad]
7 A=2;            %amplituda sygnału
8 fi=pi/11;       %faza sygnału
9 %% obliczenia
10 n=0:Nx-1;            %wektor argumentów funkcji sin
11 x=A*sin(w*n+fi);     %wektor sygnału
12 %% wizualizacja
13 figure
14    plot(n,x,'.-')
15    xlabel('n')
16    ylabel('x[n]')
17    title(['\omega=' num2str(w,'%2.2f') ' [rad]'])
18    axis tight
```

Przebiegi sygnału $x[n]$ dla zmian ω z zakresu od 0 do 2π można zaobserwować uruchamiając program 1.1 z różnymi wartościami ω. Można też zastosować prostą animację polegającą na tym, że w pętli, na wykresie rysowane są kolejne przebiegi, tak jak zaimplementowano to w programie 1.2.

Program 1.2

```
1 close all, clear all, clc
2 %% parametry
3 Nx=32;
4 w=0:pi/600:2*pi; %wektor pulsacji
5 A=2;
6 fi=0;
7 %% obliczenia
8 n=0:Nx-1;
9 for k=1:length(w)
10    %obliczenia
11    x=A*sin(w(k)*n+fi); %wektor sygnału
12    %wizualizacja
13    plot(n,x,'.-')
14    xlabel('n')
```

```
15     ylabel('x[n]')
16     title(['\omega=' num2str(w(k),'%2.2f') ' [rad]'])
17     axis([0 Nx-1 -A A])
18     drawnow
19 end
```

Zadanie 1.2

Wygenerować sygnał $x[n]=A\sin(2\pi ft+\varphi)$ o długości N=32 próbki. Częstotliwość sygnału wynosi f=10 Hz, częstotliwość próbkowania wynosi F_s=100 Hz. Narysować ten sygnał, opisać oś OX w sekundach. Zaobserwować przebiegi sygnału dla zmian f z zakresu od 0 do $2F_s$.

Jaka jest maksymalna częstotliwość sygnału f dla ustalonej częstotliwości próbkowania F_s?

Zadanie 1.3

Wygenerować zespolony ciąg eksponencjalny $x[n]=Ae^{j\omega n}$. Narysować na jednym wykresie część rzeczywistą, część urojoną i moduł tego ciągu. Opisać wykresy za pomocą funkcji `legend`.

Jednostka urojona jest oznaczana w Matlabie jako `j`, a funkcja eksponencjalna `exp`.

Zadanie 1.4

Addytywne zakłócenie szumem przebiegu sinusoidalnego. Do sygnału z zadania 1.1 dodać liczby pseudolosowe. Do generowania liczb pseudolosowych należy wykorzystać funkcje Matlaba `rand` lub `randn`.

Zadanie 1.5

Napisać funkcję liczącą histogram. Przyjąć nazwę funkcji `hist_lab`. Argumentami funkcji są sygnał x, dla którego liczymy histogram oraz liczba przedziałów Np, w których zliczamy wartości. Funkcja zwraca liczbę wartości L i środki przedziałów w.

Przetestować działanie napisanej funkcji `hist_lab` dla sygnałów pseudolosowych o rozkładzie normalnym i równomiernym. Porównać wyniki z funkcją Matlaba `hist`.

Rozwiązanie

Program 1.3

```
1 function [L,w]=hist_lab(x,Np);
2 %Funkcja do liczenia histogramu
3 %x - sygnał
4 %Np - liczba przedziałów
5 %L - liczba wartości w przedziale
6 %w - wartości
7
8 %% Przeskalowanie wartości x do przedziału (0,1) i (0,Np-1)
9 x_min=min(x(:)); x=x(:)-x_min;
10 x_max=max(x); x=x/max(x);
11 x=round((Np-1)*x);
12 %% Liczenie histogramu
13 L=zeros(1,Np);
14 for k=1:length(x)
15    L(x(k)+1)=L(x(k)+1)+1;
16 end
17 w=(0:Np-1)/Np;
18 w=w*x_max;
19 w=w+x_min;
```

Zadanie 1.6

Wygenerować sygnał sinusoidalny zmodulowany amplitudowo sygnałem sinusoidalnym: $x[n] = (1 + k\, m[n]) \cos(\omega_0\, n)$, $0 < k \leq 1$, gdzie k- jest głębokością modulacji, a $m[n]$ przebiegiem sinusoidalnym $m[n]=\cos(\omega_m)$, $\omega_m << \omega_0$.

Zaobserwować przebiegi sygnału dla różnych wartości parametrów.

Zadanie 1.7

Wygenerować sygnał sinusoidalny zmodulowany w częstotliwości sygnałem sinusoidalnym: $x[n] = \cos(\omega_0 n + k \cos(\omega_m n)/\omega_m)$. Częstotliwość chwilowa (pochodna argumentu funkcji sinus) wynosi $m[n] = \omega_0 - k \sin(\omega_m n)$, czyli oscyluje wokół ω_0. W celu zachowania dodatniej częstotliwości sygnału nośnego musi być spełnione $\omega_0 \geq k \sin(\omega_m n)$.

Zaobserwować przebiegi sygnału dla różnych wartości parametrów.

2. Splot dyskretny i widmo sygnału dyskretnego

Zadanie 2.1
Napisać funkcję do obliczania splotu liniowego dwóch wektorów o skończonych długościach wg wzoru:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]\,. \tag{2.1}$$

Przyjąć nazwę funkcji `splot`. Porównać wyniki działania napisanej funkcji z funkcją Matlaba `conv`.

Rozwiązanie
Program 2.1 przedstawia implementację splotu liniowego bezpośrednio na podstawie wzoru definicyjnego. Dla wektorów o długościach N_x i N_h wynik splotu ma długość N_x+N_h-1. Pętla zewnętrzna po n wyznacza kolejne elementy wektora $y[n]$. W pętli wewnętrznej po k są sumowane iloczyny dla ustalonego n. Warunek `if` ogranicza indeksy do zakresu wektorów, pozostałe iloczyny są równe zero.

Program 2.1
```
1 function y=splot(x,h);
2
3 Nx=length(x);
4 Nh=length(h);
5 Ny=Nx+Nh-1;
6 y =zeros(1,Ny);
7 for n=1:Ny
8     for k=1:Nx;
9           if n-k>=0 && n-k<Nh
10                y(n)=y(n)+x(k)*h(n-k+1);
11          end
12    end
13 end
```

Program 2.2 wykorzystuje fakt, że w trakcie liczenia splotu odwrócona w czasie odpowiedź impulsowa filtra jest przesuwana wzdłuż sygnału. Przed obliczeniami sygnały x i h są uzupełniane zerami. Wewnętrzna pętla po k jest zrealizowana w linii 13 w postaci iloczynu skalarnego, następnie w linii 14 współczynniki filtra są przesuwane o jedną pozycję.

Program 2.2
```
1 function y=splot(x,h);
2 %x,h - wektory poziome
3
4 Nx=length(x);
5 Nh=length(h);
6 Ny=Nx+Nh-1;
7 h = fliplr(h(:).'); %odwrócenie w czasie współczynników filtra
8 xz=zeros(1,Ny); xz(Nh:end)=x;
9 hz=zeros(1,Ny); hz(1:Nh) =h;
10 y =zeros(1,Ny);
11 hz=hz(:); %zamiana na wektor pionowy
12 for n=1:Ny;
13    y(n)=xz*hz; %iloczyn wektorowy
14    hz = [0; hz(1:Ny-1)];
```

```
15 end
```

Zadanie 2.2
Napisać funkcję do obliczania widma sygnału dyskretnego wg wzoru:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} . \tag{2.2}$$

Przyjąć następującą deklarację funkcji:

```
1 function [Xw, w]=fourier_ciagly(x,dw,wz);
2
3 %ciągła transformacja Fouriera sygnałów dyskretnych
4 %dw - krok częstotliwości [rad]
5 %wz=[w1 w2]- zakres częstotliwości [rad]
6 %x - sygnał
7 %Xw - widmo sygnału x
8 %w - pulsacje, dla których wyznaczono widmo Xw
9 %
10 %przykładowe wywołanie
11 %[Xw, w]=fourier_ciagly(x,0.01,[-pi pi]);
```

Zadanie 2.3
Wygenerować sinusoidalny sygnał testowy (patrz zadanie 1.1) i obliczyć widmo tego sygnału za pomocą funkcji `fourier_ciagly` w przedziale częstotliwości ω od $-\pi$ do π.
Narysować przebieg czasowy sygnału testowego i jego charakterystykę amplitudową.
Obliczyć widmo i narysować charakterystyki amplitudowe dla: sygnału stałego, sygnału zespolonego eksponencjalnego, sygnału pseudolosowego (funkcje `rand`, `randn`) i fali prostokątnej (funkcja `square`).

Zadanie 2.4
Napisać funkcję realizującą filtrację cyfrową wg wzoru:

$$y[n] = \sum_{m=0}^{M} \frac{b[m]}{a[0]} x[n-m] - \sum_{k=1}^{N} \frac{a[k]}{a[0]} y[n-k] . \tag{2.3}$$

Przyjąć następującą deklarację funkcji:

```
1 function y=filter_lab(b,a,x);
2 %b - współczynniki licznika transmitancji
3 %a - współczynniki mianownika transmitancji
```

Porównać wyniki działania funkcji `filter_lab` z funkcją Matlaba `filter`.

3. Projektowanie filtrów analogowych

Zadanie 3.1
Analogowy filtr dolnoprzepustowy powinien spełniać następujące wymagania: krawędź pasma przenoszenia f_{pass}=1 kHz, dopuszczalna nieliniowość wzmocnienia w paśmie przepustowym r_p=1 dB, krawędź pasma zaporowego f_{stop}=1.5 kHz, minimalne tłumienie w paśmie zaporowym r_s=30 dB. Wyznaczyć transmitancję $H(s)$ filtrów: Butterwortha, Czebyszewa typu I, Czebyszewa typu II i eliptycznego spełniających powyższe wymagania.
Podać rzędy tych filtrów. Narysować zera i bieguny transmitancji tych filtrów.
Na jednym wykresie narysować (porównać) charakterystyki amplitudowe tych filtrów.
Na jednym wykresie narysować (porównać) charakterystyki fazowe tych filtrów.

Dla filtra Butterwortha obliczyć rząd filtra:

$$N = \frac{1}{2}\log_{10}\left(\frac{10^{r_p/10}-1}{10^{r_s/10}-1}\right)/\log_{10}\left(\frac{\Omega_{pass}}{\Omega_{stop}}\right), \tag{3.1}$$

pulsację 3 dB:

$$\Omega_c = \frac{\Omega_{pass}}{(10^{r_p/10}-1)^{\frac{1}{2N}}} \tag{3.2}$$

i bieguny transmitancji:

$$s_k = j\sqrt[2N]{-1}\,\Omega_c = \Omega_c e^{j(\pi/2N)(2k+N-1)}, \quad k = 0,1,...2N-1. \tag{3.3}$$

Przejść z postaci iloczynowej transmitancji do postaci wielomianowej funkcją `zp2tf` charakterystyki częstotliwościowe obliczyć funkcją `freqs`.

Dla filtrów Czebyszewa typu I, Czebyszewa typu II i eliptycznego zastosować funkcje Matlaba `cheby1`, `cheby2` i `ellip` do wyznaczenia ich transmitancji (funkcje te należy wywołać z parametrem s).

Zadanie 3.2
Napisać program do transformacji częstotliwości unormowanego prototypu filtra analogowego z charakterystyki dolnoprzepustowej na pasmowoprzepustową, a następnie przetransformować analogowy, dolnoprzepustowy prototypu filtra Butterwortha 3-go rzędu na filtr pasmowoprzepustowy o częstotliwościach 3dB równych 5 Hz i 7Hz. Narysować charakterystyki amplitudowe obu filtrów.

Zadanie 3.3
Napisać funkcję do wyznaczania charakterystyk częstotliwościowych transmitancji analogowej $H(s)$. W obliczeniach zastosować podstawienie:

$$s=j\Omega. \tag{3.4}$$

Wartości licznika $B(j\Omega)$ i mianownika $A(j\Omega)$ obliczyć funkcją Matlaba `polyval`.
Przyjąć następującą składnię funkcji:

```
1 function H=freqs_lab(B,A,w);
2 %Charakterystyka częstotliwościowa transmitancji H(s)=B(s)/A(s)
3 %w - pulsacje [rad/s], dla których wyliczona jest charakterystyka
```

4. Projektowanie rekursywnych filtrów cyfrowych IIR

Zadanie 4.1
Cyfrowy filtr dolnoprzepustowy powinien spełniać następujące wymagania: krawędź pasma przenoszenia f_{pass}=0.8 kHz, dopuszczalna nieliniowość wzmocnienia w paśmie przepustowym r_p=1 dB, krawędź pasma zaporowego f_{stop}=1.2 kHz, minimalne tłumienie w paśmie zaporowym r_s=30 dB. Częstotliwość próbkowania F_s=4 kHz. Wyznaczyć transmitancję $H(z)$ filtrów: Butterwortha, Czebyszewa typu I, Czebyszewa typu II i eliptycznego spełniających powyższe wymagania.
Podać rzędy tych filtrów. Narysować zera i bieguny transmitancji tych filtrów.
Na jednym wykresie narysować (porównać) charakterystyki amplitudowe tych filtrów.
Na jednym wykresie narysować (porównać) charakterystyki fazowe tych filtrów.

Transmitancję filtrów wyznaczyć funkcjami Matlaba: `butter`, `cheby1`, `cheby2` i `ellip`.

Zadanie 4.2
Napisać program implementujący transformację biliniową a następnie zastosować ten program do dyskretyzacji analogowej transmitancji $H(s)$ dolnoprzepustowego prototypu filtra Butterwortha 3-go rzędu. Porównać wyniki z funkcją `bilinear`.

Zadanie 4.3
Napisać funkcję do wyznaczania charakterystyk częstotliwościowych transmitancji dyskretnej $H(z)$. W obliczeniach zastosować podstawienie:

$$z=e^{j\omega}. \tag{4.1}$$

Wartości licznika $B(e^{j\omega})$ i mianownika $A(e^{j\omega})$ obliczyć funkcją Matlaba `polyval`.
Przyjąć następującą składnię funkcji:

```
1 function H=freqz_lab(B,A,w);
2 %Charakterystyka częstotliwościowa transmitancji H(z)=B(z)/A(z)
3 %w - pulsacje [rad], dla których wyliczona jest charakterystyka
```

Zadanie 4.4
Obliczyć odpowiedzi impulsowe i skokowe filtrów z zadania 4.1. Narysować je funkcją Matlaba `stem`. Do obliczenia odpowiedzi impulsowej i skokowej wykorzystać impuls jednostkowy i impuls skokowy oraz funkcję Matlaba `filter`.

5. Projektowanie nierekursywnych filtrów cyfrowych FIR

Zadanie 5.1

1. Zaprojektować metodą okien filtr dolnoprzepustowy typu FIR, który dla częstotliwości próbkowania F_s=4 kHz ma krawędź pasma przenoszenia f_{pass}=1 kHz.
2. Zaobserwować zmiany charakterystyki amplitudowej filtra dla różnych długości wybranego okna.
3. Zaobserwować zmiany charakterystyk amplitudowych filtra dla różnych rodzajów okien o stałej długości (np. dla okna prostokątnego i Hamminga). Narysować na jednym wykresie charakterystyki amplitudowe filtra o ustalonej długości dla wybranych okien.

Odpowiedź impulsowa idealnego filtra dolnoprzepustowego typu FIR dana jest wzorem:

$$h_{FDP}[n] = \frac{\sin(\omega_c n)}{\pi n} = \begin{cases} \sin(\omega_c n)/(\pi n), & n \neq 0 \\ \omega_c / \pi, & n = 0 \end{cases}, \tag{5.1}$$

gdzie ω_c- krawędź pasma przenoszenia w [rad].

Rozwiązanie (punkty 1 i 2)

Program 5.1 wykorzystuje prostą animację umożliwiającą obserwację wpływu długości odpowiedzi impulsowej filtra na szerokość listka głównego i położenie listków bocznych dla okna prostokątnego i okna Hamminga.

Odpowiedź impulsowa filtra jest liczona w linii 8 dla n>0 i n=0, fragment $h[n]$ dla n<0 jest symetryczny (parzysty względem osi OY) do $h[n]$ dla n>0.

Usunięcie znaku `%` w linii 9 powoduje zastosowanie okna Hamminga zamiast okna prostokątnego.

Usunięcie znaku `%` w liniach 22 i 28 powoduje zmianę wyskalowania osi na charakterystykach amplitudowych.

Program 5.1

```
1 close all, clear all, clc
2 M=1:100;
3 Fs=4e3;          %Hz
4 fpass=1e3;       %Hz
5 wc=pi*fpass/(Fs/2);   %[rad]
6 f=-Fs/2:1:Fs/2;
7 for k=1:length(M)
8     n=1:M(k); h= sin(wc*n)./(pi*n); h=[fliplr(h) wc/pi h];
9     %h = h(:).*hamming(length(h));
10    Hw = freqz(h,1,f,Fs);
11    figure(1)
12          subplot(3,1,1)
13                stem(h)
14                xlabel('n')
15                ylabel('h[n]')
16                axis tight, box on
17          subplot(3,1,2)
18                plot(f,abs(Hw))
19                xlabel('f [Hz]')
20                ylabel('|H(e^j^\omega)|')
21                axis([min(f) max(f) 0 1.2]), box on
22                %axis([0 1.2*fpass 0 1.2]), box on
23          subplot(3,1,3)
24                plot(f,20*log10(abs(Hw)))
```

```
25                xlabel('f [Hz]')
26                ylabel('|H(e^j^\omega)| [dB]')
27                axis([min(f) max(f) -70 5]), box on
28                %axis([0 1.2*fpass -70 5]), box on
29    drawnow
30 end
```

Zadanie 5.2
Zaprojektować metodą okien filtr pasmowozaporowy typu FIR, który dla częstotliwości próbkowania F_s=4 kHz ma krawędzie pasma przenoszenia f_{pass1}=0.8 kHz i f_{pass2}=1.2 kHz.

Zadanie 5.3
Zaprojektować metodą okien filtr dolnoprzepustowy typu FIR, który ma krawędź pasma przenoszenia $\omega_p=\pi/4$ rad. Wykorzystując twierdzenie o modulacji przesunąć charakterystykę częstotliwościową tego filtra o $\pi/2$ i π.

Zadanie 5.4
Cyfrowy filtr dolnoprzepustowy typu FIR powinien spełniać następujące wymagania: krawędź pasma przenoszenia f_{pass}=0.8 kHz, dopuszczalna nieliniowość wzmocnienia w paśmie przepustowym r_p=1 dB, krawędź pasma zaporowego f_{stop}=1.2 kHz, minimalne tłumienie w paśmie zaporowym r_s=60 dB. Częstotliwość próbkowania F_s=4 kHz. Zaprojektować ten filtr:
1) metodą okien, poprzez dobór odpowiedniego okna Kaisera, funkcje Matlaba `kaiserord` i `fir1`.
2) metodą Parksa-McClellana, funkcja Matlaba `firpm`.
Porównać na wykresach odpowiedzi impulsowe i charakterystyki amplitudowe tych filtrów.

6. Metody obliczania DFT

Zadanie 6.1
Napisać program liczenia prostej i odwrotnej DFT wg wzorów (6.1) i (6.2):.
DFT - analiza:

$$X[k]=\sum_{n=0}^{N-1}x[n]e^{-j(2\pi/N)kn},\ 0\le n\le N-1,\quad 0\le k\le N-1 \tag{6.1}$$

IDFT - synteza:

$$x[n]=\frac{1}{N}\sum_{k=0}^{N-1}X[k]e^{j(2\pi/N)kn},\ 0\le n\le N-1,\quad 0\le k\le N-1. \tag{6.2}$$

Obliczyć DFT sygnału sinusoidalnego, wynik porównać z funkcją Matlaba `fft`. Narysować błąd rekonstrukcji

$$\varepsilon = x[n]-\mathrm{IDFT}\{\mathrm{DFT}\{x[n]\}\}. \tag{6.3}$$

Zadanie 6.2
Wygenerować macierz przekształcenia DFT:

$$W_N^{kn}=e^{-j\frac{2\pi}{N}kn}. \tag{6.4}$$

Obliczyć DFT sygnału sinusoidalnego **x** w formie macierzowej:

$$\mathbf{X}=\mathbf{W}\mathbf{x}. \tag{6.5}$$

Zaobserwować, że macierz odwrotna funkcji bazowych jest równa przeskalowanej macierzy sprzężonej:

$$\mathbf{W}^{-1}=\frac{1}{N}\mathbf{W}^{*}. \tag{6.6}$$

Dokonać syntezy sygnału w postaci macierzowej:

$$\mathbf{x}=\frac{1}{N}\mathbf{W}^{*}\mathbf{X}. \tag{6.7}$$

Zadanie 6.3
Zaimplementować algorytm FFT z podziałem w czasie. Przyjąć następującą deklarację funkcji:

```
1 function Xw=fft_lab(xt);
2 %FFT z podziałem czasowym
```

Porównać wyniki obliczeń funkcji `fft_lab` i funkcji Matlaba `fft`.

Rozwiązanie

Program 6.1

```
1 function Xw=fft_lab(xt);
2 %FFT z podziałem czasowym
```

```
3
4 N=length(xt);
5 %% uzupełnienie zerami do długości 2^v
6 v=ceil(log2(N)); %liczba bitów
7 xt=[xt zeros(1,2^v-N)];
8 N=length(xt);
9 %% odwrócenie kolejności bitów, indeksowanie od 0
10 xo= zeros(size(xt));       %inicjalizacja zmiennej
11 b = 2.^(v-1:-1:0);         %wagi binarne
12 for k=0:N-1;
13    ind=k;
14    ko=zeros(1,v);
15    for k1=0:v-1
16          if ind-b(k1+1)>=0
17                ko(k1+1)=1; ind=ind-b(k1+1);
18          end
19    end
20    ind_o=sum(fliplr(ko).*b);
21    xo(ind_o+1)=xt(k+1);
22 end
23 %% algorytm FFT
24 X=zeros(2,N);  %pamięć dla danych i wyników
25 X(1,:)=xo;     %dane
26 WN=exp(-j*2*pi/N);
27 for k=0:v-1    %pętla po etapach
28    M=2^k;      %liczba motylków w bloku
29    for k1=0:N/2/M-1;       %pętla po blokach
30          for k2=0:M-1;     %pętla wewnątrz bloku
31                %ustalenie indeksów
32                p=k1*N/2^(v-k-1)+k2;
33                q=p+M;
34                r=2^(v-k-1)*k2;
35                %obliczenia motylkowe
36                X(2,p+1)=X(1,p+1)+WN^r*X(1,q+1);
37                X(2,q+1)=X(1,p+1)-WN^r*X(1,q+1);
38          end
39    end
40    X(1,:)=X(2,:); %obliczenia z podstawianiem
41 end
42 Xw=X(1,:);
```

Zadanie 6.4
Zaimplementować algorytm IFFT z podziałem w czasie. Przyjąć następującą deklarację funkcji:

```
1 function Xw=ifft_lab(xt);
2 %IFFT z podziałem czasowym
```

Obliczyć błąd rekonstrukcji dla funkcji `fft_lab` i `ifft_lab`.

Zadanie 6.5
Zaimplementować algorytmy FFT i IFFT z podziałem w częstotliwości. Przyjąć następujące deklaracje funkcji:

```
1 function Xw=fft_dif(xt);
2 %FFT z podziałem w częstotliwości
```

```
1 function Xw=ifft_dif(xt);
```

```
2 %IFFT z podziałem w częstotliwości
```

Zadanie 6.6
Obliczyć widma dwóch sygnałów o długości *N* i wartościach rzeczywistych jednym *N*-punktowym FFT.

Zadanie 6.7
Obliczyć widmo sygnału o długości 2*N* i wartościach rzeczywistych jednym *N*- punktowym FFT.

Zadanie 6.8
Zaimplementować algorytm Goertzla obliczania DFT z odwróconą kolejnością próbek na wejściu.

Rozwiązanie

Program 6.2

```
1 close all, clear all, clc
2 N=16;
3 x=randn(1,N); %sygnał testowy
4 %% Algorytm Goertzla - odwrócona kolejność próbek na wejściu
5 xf=fliplr(x);
6 for k=1:N
7 WN_k=exp(-j*(2*pi/N)*(k-1));
8     for n=1:N
9           if n-1==0
10                y(n)=xf(n);
11          else
12                y(n)=xf(n)+y(n-1)*WN_k;
13          end
14    end
15    Yw(k)=y(n);
16 end
17 % sprawdzenie
18 Yfft=fft(x);
19 figure,
20 subplot(2,1,1), plot(real(Yw)-real(Yfft), '.-' ), xlabel('n'),
ylabel('blad_R_E'),
21 subplot(2,1,2), plot(imag(Yw)-imag(Yfft), '.-' ), xlabel('n'),
ylabel('blad_I_M'),
```

Zadanie 6.9
Zaimplementować algorytm Goertzla obliczania DFT z naturalną kolejnością próbek na wejściu.

Zadanie 6.10
Zaimplementować algorytm transformacji Chirp-Z.

7. Filtracja sygnałów cyfrowych

Zadanie 7.1
1. Wygenerować sygnał testowy $x[n]$ o długości N_x=256 próbek złożony z sumy dwóch przebiegów sinusoidalnych o częstotliwościach f_1=50 Hz i f_2=150 Hz. Częstotliwość próbkowania wynosi F_s=1000 Hz.
2. Zaprojektować cyfrowy filtr dolnoprzepustowy typu FIR (np. funkcja `fir1`, N=21) i typu IIR (np. funkcja `cheby1`, N=5) o krawędzi pasma przepuszczania 100 Hz. Narysować charakterystyki amplitudowe tych filtrów.
3. Przefiltrować sygnał testowy filtrem FIR i filtrem IIR za pomocą funkcji Matlaba `filter`.
4. Zaobserwować przebiegi czasowe i widma sygnałów przed i po filtracji.

Zadanie 7.2
Obliczyć splot sygnału testowego z zadania 7.1 z filtrem FIR przez FFT.

Zadanie 7.3
Obliczyć splotu sygnału testowego z zadania 7.1 z filtrem FIR stosując podział na sekcje metodą *overlap-add*. Sploty poszczególnych sekcji obliczyć poprzez FFT.

Rozwiązanie
Program 7.1 implementuje metodę sekcjonowanych splotów liniowych *overlap-add*. Implementacja algorytmu zaczyna się w linii 15, gdzie wybierana jest długość sekcji, która powinna być nie mniejsza niż długość filtra. Następnie odpowiedź impulsowa filtra jest uzupełniana zerami w linii 20 i wyznaczane jest widmo filtra. W pętli wybierane są kolejne, niezachodzące na siebie fragmenty sygnału wejściowego i liczone są sploty liniowe przez FFT. Zachodzące na siebie fragmenty splotów liniowych są dodawane w linii 31.

Poprawność implementacji sprawdzana jest w linii 34 przez porównanie wyniku z funkcją Matlaba `conv`.

Program 7.1
```
1 close all, clear all, clc
2 % Splot sekcjonowany metoda overlap-add
3 Nx=256;
4 f1=50; A1=1; fi1=pi/7;
5 f2=150;A2=2; fi2=0;
6 Fs=1000;
7 t=(0:Nx-1)/Fs;
8 x1=A1*sin(2*pi*f1*t+fi1);
9 x2=A2*sin(2*pi*f2*t+fi2);
10 x=x1+x2;                    %sygnał testowy
11 h=fir1(21,100/(Fs/2));      %filtr FDP
12 Nh=length(h);
13 y1=conv(h,x);               %poprawny wynik
14 %% obliczenia
15 L=32;                       %długość sekcji >= Nh
16 L=max([L,Nh]);
17 nL=ceil(Nx/L);              %liczba sekcji
18 x =[x zeros(1,nL*L-length(x))];
19 Ny=Nh+L-1;
20 hz=zeros(1,Ny);
21 hz(1:Nh)=h;
22 Hw=fft(hz); %widmo h[n] jest liczone tylko raz!!!
23 y=zeros(1,Nh+length(x)-1);
```

```
for k=1:nL
   xz=zeros(1,Ny);
   ind=1+(k-1)*L;
   xz(1:L)=x(ind:ind+L-1);
   Xw=fft(xz);
   ys=ifft(Xw.*Hw); %wynik splotu liniowego dla sekcji
   ind=1+(k-1)*L;
   y(ind:ind+Ny-1)=y(ind:ind+Ny-1)+ys;
end
y=y(1:Nx+Nh-1);
blad=sum(abs(y1-y)) %sprawdzenie przez porównanie z funkcją Matlaba conv
figure
   subplot(3,1,1), hold on
         plot(x1,'r')
         plot(x2,'k')
         legend('x_1','x_2')
         axis tight
   subplot(3,1,2)
         plot(x,'b')
         legend('x_1+x_2')
         axis tight
   subplot(3,1,3), hold on
         plot(x1,'r')
         plot(y,'b')
         legend('x_1','y')
         axis tight
```

Zadanie 7.4

Obliczyć splotu sygnału testowego z zadania 7.1 z filtrem FIR stosując podział na sekcje metodą *overlap-save*. Sploty poszczególnych sekcji obliczyć poprzez FFT.

8. Analiza częstotliwościowa z wykorzystaniem DFT

Zadanie 8.1

1. Wygenerować przebieg $x[n]=DC+A\sin(2\pi ft+\varphi)$, gdzie składowa stała DC=1, a częstotliwość f=20 Hz, próbkowany z częstotliwością F_s=100 Hz.
2. Dla tego sygnału obliczyć dyskretną transformatę Fouriera (funkcją Matlaba `fft`) na podstawie N=32 i N=64 próbek. Widma amplitudowe dla obu przypadków przedstawić na jednym wykresie z osią częstotliwości wyskalowaną w Hz i osią OY wyskalowaną w wartościach amplitudy.
3. Na jednym wykresie przedstawić widmo amplitudowe sygnału o długości N=64 próbki i widmo amplitudowe tego samego sygnału uzupełnionego zerami do długości N=1024 próbki. Oś częstotliwości wyskalować w Hz, a oś OY w wartościach amplitudy.
4. Sygnał o długości N=64 próbki przemnożyć (mnożenie indeksowe) przez okno Hamminga (funkcja Matlaba `hamming`), a następnie uzupełnić zerami do długości N=1024 próbki. Narysować widmo amplitudowe tego sygnał. Dla porównania na tym samym wykresie narysować widmo amplitudowe sygnału o długości N=64 próbki z oknem prostokątnym po uzupełnieniu zerami do tej samej długości N=1024 próbki. Oś częstotliwości wyskalować w Hz.

Zadanie 8.2

Dla sygnału w postaci $x[n]=A\sin(2\pi f_1 t+\varphi_1)+A\sin(2\pi f_2 t+\varphi_2)$ o długości N=256 próbek, f_1=20Hz i częstotliwości próbkowania F_s=100 Hz zaobserwować jego widmo amplitudowe dla f_2 z przedziału od 21 Hz do 40 Hz. Przed liczeniem widma uzupełnić sygnał zerami do długości N=2048 próbek.

Rozwiązanie

Program 8.1 przedstawia animację ilustrującą problem rozdzielczości częstotliwościowej DFT. Należy zwrócić uwagę, że nawet w przypadku, gdy częstotliwości f_1 i f_2 różnią się znacznie maksimum charakterystyki amplitudowej przeważnie nie leży w punkcie (20Hz,1). Zjawisko to jest spowodowane przeciekiem widmowym.

Program 8.1

```
close all, clear all, clc
% Rozdzielczość częstotliwościowa DFT
Nx =256;
Nfft = 2048;
f1=20; A1=1; fi1=pi/7;
f2=21:0.05:40; A2=1; fi2=0;
Fs=1000;
t=(0:Nx-1)/Fs;
f=(0:Nfft-1)*(Fs/Nfft);
x1=A1*sin(2*pi*f1*t+fi1);
for k=1:length(f2)
   x2=A2*sin(2*pi*f2(k)*t+fi2);
   x=x1+x2;
   Xw=fft(x,Nfft); Xw=2*Xw/Nx; Xw(1)=Xw(1)/2;
   plot(f,abs(Xw),'.-');
         title(['f_2=' num2str(f2(k),'%2.2f') ' Hz'])
         xlabel('f [Hz]')
         ylabel('|X(e^j^\omega)|')
         axis([0 50 0 1.5]), grid on
         drawnow
```

```
21 end
```

Zadanie 8.3
Dla sygnału w postaci $x[n]=A_1\sin(2\pi f_1 t+\varphi_1)+A_2\sin(2\pi f_2 t+\varphi_2)$ o długości N=256 próbek, f_1=20Hz, f_2=24Hz i częstotliwości próbkowania F_s=100 Hz zaobserwować jego widmo amplitudowe dla A_2 z przedziału od 0 do A_1. Przed liczeniem widma uzupełnić sygnał zerami do długości N=2048 próbek.

Rozwiązanie
Program 8.2 przedstawia animację ilustrującą problem rozdzielczości amplitudowej DFT. Należy zwrócić uwagę, że nawet w przypadku, gdy amplituda A_2 jest na tyle duża, że listek główny sygnału o częstotliwości f_2 nie jest maskowany przez listki boczne sygnału o częstotliwości f_1 to listki główne obu tych sygnałów nie mają ekstremów w częstotliwościach f_1 i f_2, co jest spowodowane przeciekiem widmowym.

Program 8.2
```
1 close all, clear all, clc
2 % Rozdzielczość amplitudowa
3 Nx =256;
4 Nfft = 2048;
5 f1=20; A1=1; fi1=pi/7;
6 f2=24; A2=0:0.01:1; fi2=0;
7 Fs=1000;
8 t=(0:Nx-1)/Fs;
9 f=(0:Nfft-1)*(Fs/Nfft);
10 x1=A1*sin(2*pi*f1*t+fi1);
11 for k=1:length(A2)
12    x2=A2(k)*sin(2*pi*f2*t+fi2);
13    x=x1+x2;
14    Xw=fft(x,Nfft); Xw=2*Xw/Nx; Xw(1)=Xw(1)/2;
15    plot(f,abs(Xw),'.-');
16          title(['A_2=' num2str(A2(k),'%2.2f')])
17          xlabel('f [Hz]')
18          ylabel('|X(e^j^\omega)|')
19          axis([0 50 0 1.5]), grid on
20          drawnow
21 end
```

Zadanie 8.4
Dla sygnału w postaci $x[n]=A\sin(2\pi f_1 t+\varphi_1)+A\sin(2\pi f_2 t+\varphi_2)$, f_1=20Hz, f_2=22Hz, częstotliwość próbkowania F_s=100 Hz zaobserwować jego widmo amplitudowe dla liczby próbek N z przedziału od 256 do 2000. Przed liczeniem widma uzupełnić sygnał zerami do długości N=2048 próbek.

Rozwiązanie
Program 8.3 przedstawia możliwość zwiększenia rozdzielczości częstotliwościowej i amplitudowej DFT przez wydłużenie okna obserwacji.
Należy zwrócić uwagę, że przeciek widmowy może być znacznie ograniczony przez stosowanie okien czasowych innych niż prostokątne. Usunięcie znaku % w linii 14 powoduje zastosowanie okna Hamminga. Sygnał jest dzielony przez wartość średnią okna, żeby prążki DFT miały wysokość amplitudy sygnału.

Program 8.3
```
1 close all, clear all, clc
```

```
% Poprawa rozdzielczość przez wydłużenie obserwacji
Nx =[256:4:2000];
Nfft = 2048;
f1=20; A1=1; fi1=pi/7;
f2=23; A2=1; fi2=0;
Fs=1000;
f=(0:Nfft-1)*(Fs/Nfft);
for k=1:length(Nx)
    t=(0:Nx(k)-1)/Fs;
    x1=A1*sin(2*pi*f1*t+fi1);
    x2=A2*sin(2*pi*f2*t+fi2);
    x=x1+x2;
    %w=hamming(length(x)); x=x(:).*w; x=x/mean(w);
    Xw=fft(x,Nfft); Xw=2*Xw/Nx(k); Xw(1)=Xw(1)/2;
    plot(f,abs(Xw),'.-');
    title(['N=' num2str(Nx(k),'%2.0f')])
    xlabel('f [Hz]')
    ylabel('|X(e^j^\omega)|')
    axis([10 30 0 1.2]), grid on
    drawnow
end
```

Zadanie 8.5
Napisać funkcję do liczenia okien Rifea-Vincenta klasy I dla rzędów M=0,1,2,...,6. Przyjąć następującą deklarację funkcji:

```
function [okno, Am]=window_RV(N,ord);
%okna Rifea-Vincenta klasy I
%N - długość okna
%ord - rząd okna 0,1,2,...,6 0-prostokątne, 1-Hanning
```

Dla wybranej długości okna, np. N=32, zaobserwować przebiegi czasowe okna oraz charakterystyki amplitudowe w skali decybelowej.

Zadanie 8.6
Napisać funkcję do liczenia interpolowanego DFT dla okien Rifea-Vincenta klasy I dla rzędów M=0,1,2,...,6. Przyjąć następującą deklarację funkcji:

```
function [Xw,w0,fi,Xw3p,w3p]=interp_dft_Rif_Vinc(x,ord);
% Interpolowane DFT dla okien Rifea-Vincenta klasy I
% x - sygnał
% ord- rząd okna 0,1,...6, ord=0-okno prostokątne, ord=1-okno Hanninga
% Xw - amplituda, interpolacja dwupunktowa
% w0 - pulsacja cyfrowa [rad], interpolacja dwupunktowa
% fi - faza [rad]
% Xw3p- amplituda, interpolacja trzypunktowa
% w03p-pulsacja cyfrowa [rad], interpolacja trzypunktowa
```

Zadanie 8.7
Wyznaczyć błędy systematyczne estymacji częstotliwości dla dwupunktowej i trzypunktowej interpolacji DFT dla okien Rifea-Vincenta klasy I dla ustalonej długości sygnału, np. N=128. Narysować zależność błędu maksymalnego estymacji częstotliwości od częstotliwości cyfrowej sygnału. Częstotliwość sygnału testowego zmieniać w przedziale $(0, \pi)$ z krokiem $\pi/20$. Dla ustalonej częstotliwości wygenerować sygnały testowe z fazą zmieniającą się w przedziale $(-\pi/2, \pi/2)$ z krokiem $\pi/20$; za wynik przyjąć maksymalną różnicę pomiędzy

zadaną częstotliwością sygnału testowego, a częstotliwością estymowaną dla sygnałów o różnej fazie.

Zadanie 8.8
Napisać funkcję liczącą spektrogram. Przyjąć następującą deklarację funkcji:

```
function [X,n_spe,w_spe]=spektogram(x,w,Nfft,R);
%x - sygnał
%w - okno czasowe - wektor o długości L
%Nfft - długość fft >= L
%R - przesunięcie okna
%X - spektrogram - płaszczyzna czas-częstotliwość
%n_spe - indeks czasu
%w_spe - pulsacja cyfrowa
```

Następnie użyć napisaną funkcję `spektogram` w programie 8.4 do obserwacji widma sygnału w czasie dla modulacji amplitudy i modulacji częstotliwości. Dokonać obserwacji spektrogramu dla różnych długości okna prostokątnego i Hamminga oraz dla różnych przesunięć okna *R*.

Program 8.4
```
close all, clear all, clc
Nx=2048;
n=(0:Nx-1);
if 0 % modulacja AM
    w0=1;
    wm=0.01;
    k =0.9;
    x1=1+k*cos(wm*n);
    x2=cos(w0*n);
    x =x1.*x2;
else % modulacja FM
    w0 = pi/2;
    wm = 0.01;
    k = 1;
    x = cos(w0*n+k*cos(wm*n)/wm);
end
L = 128;
Nfft= 256;
R = 64;
w = ones(1,L);
%w = hamming(L);
[X,n_spe,w_spe]=spektogram(x,w,Nfft,R);
figure,
    plot(x, '.-k','LineWidth',1)
    xlabel('n'), ylabel('x[n]')
    axis tight
figure
    imagesc(n_spe,w_spe,abs(X)), axis xy, colorbar
    xlabel('n'), ylabel('\omega [rad]')
    title(['L=' num2str(L) ', R=' num2str(R) ' , N_F_F_T='
num2str(Nfft)])
    axis tight
```

Zadanie 8.9
Napisać funkcję liczącą periodogram:

$$I[k] = \frac{1}{LU} |\,\mathrm{DFT}\{w[n]x[n]\}\,|^2, U = \frac{1}{L}\sum_{n=0}^{L-1}(w[n])^2\,. \tag{8.1}$$

Przyjąć następującą deklarację funkcji:

```
function [Iw,w] =periodogram_lab(x,okno);
%x        - sygnał
%okno     - wektor okna czasowego o długości sygnału
%I        - periodogram
%w        - pulsacja cyfrowa [rad]
```

Porównać wyniki z funkcją Matlaba `periodogram` wg programu 8.5.

Program 8.5

```
close all, clear all, clc
Nx=1024;
n=0:Nx-1;
x=sin(n*pi/4)+2.5*randn(1,Nx);
w=hamming(Nx);
%w=ones(Nx,1);
[Pxx,w1] = periodogram(x,w); Pw = Pxx*pi; Pw(1) =Pw(1)*2;
[Iw, w2] = periodogram_lab(x,w);
figure
    subplot(2,1,1)
          plot(n, x,'-k')
          xlabel('n')
          ylabel('x[n]')
          axis tight, box on
    subplot(2,1,2), hold on
          plot(w2,Iw,'.-k')
          plot(w1,Pw,'o-b')
          xlabel('\omega'), ylabel('I(\omega)')
          axis tight
```

Zadanie 8.10
Napisać funkcję liczącą uśredniony periodogram metodą Welcha:

$$\hat{I}[k] = \frac{1}{K}\sum_{r=0}^{K-1} I_r[k]\,, \tag{8.2}$$

gdzie $I_r[k]$ są periodogramami (8.1) dla kolejnych przesunięć okna analizy.
Przyjąć następującą deklarację funkcji:

```
function [Iw,w] = periodogram_usredniony(x,okno,R, Nfft);
%x        - sygnał
%okno     - wektor okna czasowego
%Nfft     - długość FFT
%R        - przesunięcie okna (liczba próbek)
%I        - periodogram
%w        - pulsacja cyfrowa [rad]
```

Porównać wyniki z funkcją Matlaba `pwelch` wg programu 8.6.

Program 8.6

```
close all, clear all, clc
```

```
Nx=1024;
L =128; %długość okna
R =round(L/2); %przesunięcie
n=0:Nx-1;
x=sin(n*pi/4)+2.5*randn(1,Nx);
%w=hamming(L);
w=ones(L,1);
 [Iw, w1] = periodogram_usredniony(x,w,R,Nx);
[Pxx,w2] = pwelch(x,w,L-R,Nx); Pxx=Pxx*pi; Pxx(1)=2*Pxx(1);
figure
   subplot(2,1,1)
         plot(n, x,'-k')
         xlabel('n')
         ylabel('x[n]')
         axis tight, box on
   subplot(2,1,2), hold on
         plot(w1, Iw ,'.-k')
         plot(w2, Pxx,'o-b')
         xlabel('\omega'), ylabel('I(\omega)')
         axis tight, box on
```

9. Zmiana częstotliwości próbkowania. Sygnał analityczny

Zadanie 9.1
1. Wygenerować sygnał testowy $x[n]=A\sin(2\pi ft+\varphi)$ o częstotliwości f=100 Hz, długości N=256 próbkowany z F_s=1000 Hz. Napisać program do zmiany częstotliwości próbkowania tego sygnału z F_s=1000 Hz na F_s=1200 Hz.
2. Narysować przebiegi czasowe sygnału dla F_s=1000 Hz i F_s=1200 Hz na jednym wykresie z osią *OX* wyskalowaną w sekundach.
3. Narysować widma amplitudowe sygnału dla F_s=1000 Hz i F_s=1200 Hz na jednym wykresie z osią *OX* wyskalowaną w hercach.
4. Porównać wyniki z funkcją Matlaba `resample`.

Zadanie 9.2
Zastosować sygnał analityczny, obliczony funkcją Matlaba `hilbert`, do demodulacji amplitudy i częstotliwości sygnałów testowych z programu 8.4.

Zadanie 9.3
Napisać program do obliczania sygnału analitycznego metodą filtracji w dziedzinie czasu. Odpowiedź impulsowa idealnego filtra Hilberta dana jest wzorem:

$$h[n]=\begin{cases}\dfrac{2\sin^2(\pi n/2)}{\pi n}, & n\neq 0\\ 0, & n=0\end{cases}. \tag{9.1}$$

Zastosować wyznaczony sygnał analityczny do demodulacji amplitudy i częstotliwości sygnałów testowych z programu 8.4.

Zadanie 9.4
Napisać program do obliczania sygnału analitycznego metodą modyfikacji widma:

$$Z[k]=\begin{cases}X[k], & k=0\\ 2X[k], & k=1,2,\ldots,\dfrac{N}{2}-1\\ X[k], & k=\dfrac{N}{2}\\ 0, & k=\dfrac{N}{2}+1,\ldots,N-1\end{cases}. \tag{9.2}$$

Zastosować wyznaczony sygnał analityczny do demodulacji amplitudy i częstotliwości sygnałów testowych z programu 8.4.

10. Filtry adaptacyjne

Zadanie 10.1
Napisać funkcję realizującą algorytm filtra adaptacyjnego RLS.

Algorytm RLS jest następujący:

I. Inicjalizacja

1. Wybór długości filtra M,
2. Początkowe współczynniki filtra można ustawić na zero $\mathbf{h}_0=\mathbf{0}$,
3. Początkowa macierz odwrotna $\mathbf{P}_0=\delta^{-1}\mathbf{I}$, gdzie δ jest małą wartością dodatnią.

II. Obliczenia dla N=1,2,3,...

1. $\mathbf{k}_N = \dfrac{\mathbf{R}_{N-1}^{-1}\mathbf{q}_N}{1+\mathbf{q}_N^H\mathbf{R}_{N-1}^{-1}\mathbf{q}_N}$,
2. $\varepsilon_N = d_N - \mathbf{q}_N^H\mathbf{h}_{N-1}$,
3. $\mathbf{h}_N = \mathbf{h}_{N-1} + \mathbf{k}_N\varepsilon_N$,
4. $\mathbf{P}_N = \mathbf{P}_{N-1} - \mathbf{k}_N\mathbf{q}_N^H\mathbf{P}_{N-1}$.

Rozwiązanie

Program 10.1

```
function [hrls, h_n, e_n] = rls_lab(x,d,hrls,delt);
%Filtr adaptacyjny RLS
%x        - sygnał wejściowy
%d        - sygnał odniesienia
%hrls     - współczynniki filtra
%delt     - mała wartość dodatnia, np.0.01
%h_n      - kolejne wektory współczynników filtra
%e_n      - kolejne wartości błędu

M = length(hrls) ;
P = 1/delt*eye(M,M);
q = zeros(M,1);
h_n=hrls;
e_n=[];
for k=1:length(x)
   q = [q(2:M); x(k)];
   kg= (P*q)/(1+q'*P*q);
   e = d(k)-q'*hrls;       e_n = [e_n e];
   hrls = hrls + kg*e;     h_n = [h_n hrls];
   P = P - kg*q'*P;
end
```

Zadanie 10.2
Napisać funkcję realizującą algorytm filtra adaptacyjnego LMS.

Algorytm LMS jest następujący:

I. Inicjalizacja

1. Wybór długości filtra M,
2. Początkowe współczynniki filtra można ustawić na zero $\mathbf{h}_0=\mathbf{0}$,
3. Wybór wartości μ.

II. Obliczenia dla N=1,2,3,...

1. $e_N = d_N - \mathbf{x}_N^T\mathbf{h}_N$,
2. $\mathbf{h}_{N+1} = \mathbf{h}_N + \mu\mathbf{x}_N e_N$.

Przyjąć następującą deklarację funkcji:

```
function [hlms, h_n, e_n] = lms_lab(x,d,hlms,mi);
%Filtr adaptacyjny LMS
%x         - sygnał wejściowy
%d         - sygnał odniesienia
%hlms      - współczynniki filtra
%mi        - krok adaptacji, np.0.01
%h_n       - kolejne wektory współczynników filtra
%e_n       - kolejne wartości błędu
```

Zadanie 10.3
Zastosować napisane funkcje `rls_lab` i `lms_lab` do identyfikacji odpowiedzi impulsowej układu zgodnie z programem 10.2.

Program 10.2

```
close all, clear all, clc
%% identyfikacja odpowiedzi impulsowej h[n]
h = fir1(10,0.5);      %odpowiedź impulsowa do identyfikacji
x = randn(1,500);      %wymuszenie
d = conv(x,h);         %odpowiedź układu identyfikowanego na x
d = d + 0.1*randn(size(d)); %pomiar odpowiedzi zakłócony szumem
%% zadanie: zastosować RLS i LMS do wyznaczenia h na podstawie x i d
```

11. Transformacja falkowa

Zadanie 11.1
Napisać funkcję implementującą transformację falkową w wersji predykcyjnej z opcją transformacji całkowitoliczbowej.

Rozwiązanie

Program 11.1

```
1 function [c,d,P,U]=lwt_lab(x,P,U,int);
2 %LWT x musi mieć parzystą dlugosć
3 %P,U - numer predyktora i updata od 1 do 4;
4 %int=1 - transformacja całkowitoliczbowa
5
6 switch P
7     case 1
8           P=[1];
9     case 2
10          P=[1/2 1/2];
11    case 3
12          P=[-1/2^4 9/2^4 9/2^4 -1/2^4];
13    case 4
14          P=[3/2^8 -25/2^8 150/2^8 150/2^8 -25/2^8 3/2^8];
15    otherwise
16          P=[1/2 1/2];
17 end
18 switch U
19    case 1
20          U=[1];
21    case 2
22          U=[1/2 1/2];
23    case 3
24          U=[-1/2^4 9/2^4 9/2^4 -1/2^4];
25    case 4
26          U=[3/2^8 -25/2^8 150/2^8 150/2^8 -25/2^8 3/2^8];
27    otherwise
28          U=[1/2 1/2];
29 end
30 x=x(:); N =length(x);
31 %% podział
32 ce=x(1:2:N);
33 co=x(2:2:N);
34 %% predykcja
35 if int==1
36    d = co-round(lwt_step(ce,P));
37 else
38    d = co-lwt_step(ce,P);
39 end
40 %% uaktualnienie
41 if int==1
42    c = ce+round(lwt_step(d,U)/2);
43 else
44    c = ce+lwt_step(d,U)/2;
45 end
```

Zadanie 11.2
Napisać funkcję implementującą odwrotną transformację falkową w wersji predykcyjnej z opcją transformacji całkowitoliczbowej.
Zastosować następującą deklarację funkcji:

```
function x=ilwt_lab(c,d,P,U,int);
%odwrotna LWT
%P,U predyktor i update z funkcji lwt_lab
%int=1 transformacja całkowitoliczbowa
```

Sprawdzić błąd rekonstrukcji sygnału dla transformacji zmiennoprzecinkowej i całkowitoliczbowej w programie 11.2.

Program 11.2

```
close all, clear all, clc
%% generator sygnału EKG
x = ecg(128);
x = repmat(x,[1 9]);
x = x+0.05*randn(size(x));  %szum pomiarowy
x = sgolayfilt(x,0,5);      %wygładzenie
%% kwantowanie na Lb bitów
Lb=8;
x = x-min(x); x=x/max(x); x=round((2^Lb-1)*x);
%% Transformacja falkowa
P=2;
U=2;
int=0;
[c,d,P1,U1] = lwt_lab(x,P,U,int);      %analiza
xr = ilwt_lab(c,d,P1,U1,int);          %synteza
blad=x(:)-xr(:);
figure
   subplot(2,1,1)
         plot(x);
         xlabel('n')
         ylabel('x[n]')
         axis tight,
   subplot(2,1,2)
         plot(blad,'.-k')
         xlabel('n')
         ylabel('x[n]-x_r[n]')
         axis tight
```

Zadanie 11.3
Napisać funkcję do wielopoziomowej falkowej dekompozycji sygnału.

Rozwiązanie

Program 11.3

```
function [c,dall,P1,U1]=lwt_level(x,P,U,int,lv);
%Wielopoziomowa dekompozycja LWT
%długość x musi dzielić się przez 2^lv
%P,U - numer predyktora i updata od 1 do 4;
%int=1 - transformacja całkowitoliczbowa
%lv - liczba poziomów dekompozycji

c=x;
dall=[];
for k=1:lv
```

```
11     [c,d,P1,U1]=lwt_lab(c,P,U,int);
12     dall=[d; dall];
13 end
```

Zadanie 11.4
Napisać funkcję do wielopoziomowej falkowej syntezy sygnału. Przyjąć następującą deklarację funkcji:

```
1 function c=ilwt_level(c,dall,P,U,int,lv);
2 %Wielopoziomowa synteza LWT
3 %długość x musi dzielić się przez 2^lv
4 %P,U predyktor i update z funkcji lwt_lab
5 %int=1 - transformacja całkowitoliczbowa
6 %lv - liczba poziomów dekompozycji
```

Sprawdzić błąd rekonstrukcji dla dekompozycji i analizy wielopoziomowej analogicznie jak w programie 11.2.

Zadanie 11.5
Napisać program do rysowania falki i funkcji skalującej.

Zadanie 11.6
Napisać program do rysowania płaszczyzny czas-skala dla wielopoziomowej, dyskretnej analiz falkowej sygnału.

Rozwiązanie

Program 11.4

```
1 close all, clear all, clc
2 %% generator sygnału EKG
3 x = ecg(128);
4 x = repmat(x,[1 9]);
5 x = x+0.05*randn(size(x));  %szum pomiarowy
6 x = sgolayfilt(x,0,5);      %wygładzenie
7 %% kwantowanie na Lb bitów
8 Lb=8; x = x-min(x); x=x/max(x); x=round((2^Lb-1)*x);
9 %% LWT
10 P=2; U=2;
11 lv = 5;
12 int=0;
13 [c,dall,P1,U1]=lwt_level(x,P,U,int,lv); %analiza wielopoziomowa
14 %% Prezentacja detali na płaszczyźnie czas-skala
15 k1=1;
16 Nc = length(c);
17 TS = zeros(lv,length(c)+length(dall));
18 for k=1:lv
19     d = dall(k1:k1+Nc-1); k1=k1+Nc; Nc=2*Nc;
20     d=repmat(d(:),[1 2^(lv-k+1)]); d=d.'; d=d(:);
21     TS(lv-k+1,:)=d.';
22 end
23 figure
24     imagesc(abs(TS)), axis xy, colorbar
25     xlabel('n')
26     ylabel('j')
27     axis tight
```

12. Kompresja sygnałów

Zadanie 12.1
Napisać funkcję do liczenia entropii dla wektora liczb całkowitych wg wzoru:

$$H(p_1,...,p_n) = H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i \,. \tag{12.1}$$

Zadanie 12.2
1. Obliczyć entropię sygnału rzeczywistego (np. sygnału EKG) o długości N=4096.
2. Obliczyć entropię tego sygnału po kodowaniu różnicowym:

$$x[n] = \begin{cases} x[n], & n = 1 \\ x[n] - x[n-1], & n > 1 \end{cases} \quad n = 1,2,...,N \,. \tag{12.2}$$

3. Obliczyć entropię tego sygnału na 5 poziomie całkowitoliczbowej dekompozycji falkowej.

Zadanie 12.3
Zakodować koderem Huffmana i koderem arytmetycznym sygnał rzeczywisty (np. EKG) o długości N=4096. Określić średnią liczbę bitów na symbol dla każdego z koderów, liczby te porównać z entropią sygnału.

Zadanie 12.4
1.Wykonać pięciopoziomową, całkowitoliczbową dekompozycję falkową sygnału rzeczywistego (np. EKG) o długości N=4096. Współczynniki detali, których amplituda jest mniejsza od wartości przyjętego progu T_r ustawić na wartość zero.
2. Obliczyć entropię współczynników falkowych po operacji progowania.
3. Obliczyć błąd PRD (*Percent Residual Difference*) rekonstrukcji sygnału:

$$PRD = \sqrt{\sum_n \left(x[n] - x_r[n]\right)^2 \Big/ \sum_n x[n]^2} \cdot 100\% \,. \tag{12.3}$$

4. Dla zwiększanych wartości progu T_r={0,1,2,3.....} wyznaczyć charakterystykę $PRD=f\{H\}$ obrazującą zależność pomiędzy współczynnikiem kompresji, a błędem rekonstrukcji, tj. jakością kompresji stratnej.

13. Filtracja obrazów

Zadanie 13.1
1. Wygenerować obraz testowy `I=checkerboard(60,2,2)` i zakłócić go addytywnie szumem.
2. Przefiltrować obraz testowy filtrami uśredniającymi 2D: filtrem Gaussa, filtrem o prostokątnej odpowiedzi impulsowej i filtrem o odpowiedzi impulsowej w kształcie dysku. Użyć funkcji Matlaba `imfilter` lub `conv2`.
3. Przefiltrować obraz testowy filtrem medianowym, funkcja Matlaba `medfilt2`.

Zadanie 13.2
Przefiltrować obraz naturalny (np. `I =imread('cameraman.tif')`) filtrami z zadania 13.1.

Zadanie 13.3
Przefiltrować obrazy testowe z zadania 13.1 i zadania 13.2 filtrami do detekcji krawędzi.

Zadanie 13.4
Napisać funkcję liczącą ciągłe widmo 2D obrazu.

Rozwiązanie

Program 13.1
```
1 function [Xw,w1,w2]=fourier_ciagly_2D(x,dw,wz);
2 % 2D FT
3 for k=1:size(x,1)
4     [H, w1]=fourier_ciagly(x(k,:),dw,wz);
5     Xw1(k,:)= H;
6 end
7 for k=1:size(Xw1,2)
8     [H, w2]=fourier_ciagly(Xw1(:,k),dw,wz);
9     Xw(:,k)= H(:);
10 end
```

Funkcja `fourier_ciagly_2D` liczy transformatę Fouriera wierszy a następnie kolumn obrazu za pomocą funkcji `fourier_ciagly` dla sygnałów 1D.

Program 13.2
```
1 function [Xw, w]=fourier_ciagly(x,dw,wz);
2 %ciągła transformacja Fouriera sygnałów dyskretnych
3 %dw - krok częstotliwości [rad]
4 %wz=[w1 w2]- zakres częstotliwości [rad]
5 %x - sygnał
6 %Xw - widmo sygnału x
7 %w - pulsacje, dla których wyznaczono widmo Xw
8 %
9 %przykładowe wywołanie
10 %[Xw, w]=fourier_ciagly(x,0.01,[-pi pi]);
11
12 w=wz(1):dw:wz(2);
13 nn=0:1:length(x)-1;
14 for k=1:length(w)
15    Xw(k)=exp(-j*w(k)*nn)*x(:);
```

```
16 end
```

Zadanie 13.5
1. Zaprojektować separowalny filtr dolnoprzepustowy o rozmiarze 11x11 o częstotliwości odcięcia π/4.
2. Narysować odpowiedź impulsową i charakterystykę amplitudową tego filtra funkcją Matlaba `surf`.
3. Przefiltrować tym filtrem obrazy testowe z zadania 13.1 i zadania 13.2.

Zadanie 13.6
1. Zaprojektować separowalny filtr górnoprzepustowy o rozmiarze 11x11 o częstotliwości odcięcia π/4.
2. Narysować odpowiedź impulsową i charakterystykę amplitudową tego filtra.
3. Przefiltrować tym filtrem obrazy testowe z zadania 13.1 i zadania 13.2.

Zadanie 13.7
1. Zaprojektować cylindryczny filtr dolnoprzepustowy o rozmiarze 11x11 o częstotliwości odcięcia π/4.
2. Narysować odpowiedź impulsową i charakterystykę amplitudową tego filtra.
3. Przefiltrować tym filtrem obrazy testowe z zadania 13.1 i zadania 13.2.

Zadanie 13.8
1. Zaprojektować cylindryczny filtr górnoprzepustowy o rozmiarze 11x11 o częstotliwości odcięcia π/4.
2. Narysować odpowiedź impulsową i charakterystykę amplitudową tego filtra.
3. Przefiltrować tym filtrem obrazy testowe z zadania 13.1 i zadania 13.2.

14. DFT i DWT obrazów

Zadanie 14.1

1. Napisać funkcję do liczenia DFT sygnałów 2D z wykorzystaniem funkcji Matlaba `fft`.
2. Zamienić ćwiartki płaszczyzny $X[k_1,k_2]$ tak, aby składowa stała $X[0,0]$ była w środku płaszczyzny, w tym celu użyć funkcji Matlaba `fftshift`.
3. Zaobserwować widmo obrazu naturalnego (np. `I =imread('cameraman.tif')`) o rozmiarze 256x256 w skali liniowej i skali decybelowej.
4. Zaobserwować widmo obrazu naturalnego po odjęciu wartości średniej, tj. składowej stałej.
5. Zaobserwować widmo obrazu naturalnego z oknem separowalnym (np. Hamminga).
6. Zwiększyć gęstość próbkowania osi częstotliwości ω_1 i ω_2 przez uzupełnienie obrazu zerami do rozmiaru 512x512.

Zadanie 14.2

Za pomocą IDFT 2D (funkcja Matlaba `ifft2`) wyznaczyć odpowiedź impulsową dolnoprzepustowego filtra cylindrycznego o częstotliwości odcięcia $\pi/4$.

Rozwiązanie

Program 14.1

```
close all, clear all, clc
%% Projektowanie filtra cylindrycznego w dziedzinie DFT
N = 256;        %rozmiar widma 2*N x 2*N
w = pi/4;       %częstotliwość odcięcia [rad]
R = N*w/pi;
Hw=zeros(2*N,2*N);
for k1=1:2*N
    for k2=1:2*N
          promien=sqrt((N-k1+1)^2+(N-k2+1)^2);
          if promien<R
                Hw(k1,k2)=1;
          end
    end
end
%Hw=~Hw; %FGP
phi1=exp(-j*(1:2*N)*pi); %liniowa faza
phi2=exp(-j*(1:2*N)*pi); %liniowa faza
Hw=Hw.*(phi1(:)*phi2);
hr=ifft2(fftshift(Hw));
hr=real(hr);
%% sprawdzenie
Nh = 8;  %rozmiar filtra 2*Nh x 2*Nh
k1 = 64; %rozmiar widma
h = hr(N-Nh+1:N+Nh,N-Nh+1:N+Nh);       %fragment odpowiedzi impulsowej
h=h.*(hamming(2*Nh)*hamming(2*Nh)');   %okno czasowe
hz=zeros(k1,k1); hz(1:2*Nh,1:2*Nh)=h;  %zagęszczenie próbkowania osi
częstotliwości
Hzw=fft2(hz);
Hzw=fftshift(Hzw);
w=2*pi*((0:k1-1)-k1/2)/k1;
figure
   surf(h)
   xlabel('n_1'), ylabel('n_2'), zlabel('h[n_1,n_2]')
   axis tight, box on
figure,
```

```
35     surf(w,w,abs(Hzw)), alpha(0.5)
36     xlabel('\omega_1'), ylabel('\omega_2')
37     zlabel('|X(\omega_1,\omega_2)|')
38     axis tight, box on
```

Zadanie 14.3
Przefiltrować dolnoprzepustowo obraz naturalny w dziedzinie częstotliwości, tzn. policzyć widmo obrazu, wyzerować właściwe współczynniki widma $X[k_1,k_2]$ i wrócić do dziedziny czasu za pomocą funkcji Matlaba `ifft2`.

Zadanie 14.4
Napisać program do wielopoziomowej, falkowej dekompozycji i rekonstrukcji obrazów. DWT 2D zaimplementować w postaci predykcyjnej z opcją transformacji całkowitoliczbowej.

Rozwiązanie

Program 14.2 wykorzystuje funkcje `lwt2d` i `ilwt2d`. Implementacje funkcji `lwt_lab` i `lwt_step` podane są w rozdziale 11.

Program 14.2
```
1 close all, clear all, clc
2 Ao = double(imread('cameraman.tif'));
3 Lv=3;     %poziom dekompozycji
4 P=2; U=2; int=1;
5 %% Dekompozycja
6 A = double(Ao);
7 AA = A;
8 for k=1:Lv
9     [AA, AD, DD, DX, P1, U1]=lwt2d(AA,P,U,int);
10    temp=[AA AD; DD DX]; [n1, n2]=size(temp);
11    A(1:n1,1:n2)=temp;
12    figure,imshow(uint8(A));
13 end
14 %% Rekonstrukcja
15 [n1,n2]=size(A);
16 n1=n1/2^Lv;
17 n2=n2/2^Lv;
18 for k=1:Lv
19    AA = A(1:n1 , 1:n2);
20    AD = A(1:n1 , 1+n2:2*n2);
21    DD = A(1+n1:2*n1, 1:n2);
22    DX = A(1+n1:2*n1 , 1+n2:2*n2);
23    AR=ilwt2d(AA, AD, DD, DX, P1, U1, int);
24    A(1:2*n1,1:2*n2)=AR;
25    n1=2*n1;
26    n2=2*n2;
27 end
28 blad=sum(abs(A(:)-Ao(:)))
```

Program 14.3
```
1 function [AA, AD, DD, DX, P1, U1]=lwt2d(B,P,U,int);
2 %Dekompozycja falkowa obrazów
3 % AA | DD
4 % ---------
5 % AD | DX
6
```

```
7 [n1 n2]=size(B); k=round(n2/2); w=round(n1/2);
8 A=zeros(n1,k);
9 D=zeros(n1,k);
10 AA=zeros(w,k); AD=AA; DD=AA; DX=AA;
11 for n=1:n1
12    [A(n,:),D(n,:),P1,U1]=lwt_lab(B(n,:),P,U,int);
13 end
14 [n1 n2]=size(A);
15 for n=1:n2
16    [a,d]=lwt_lab(A(:,n),P,U,int);
17    AA(:,n)=a;
18    AD(:,n)=d;
19 end
20 for n=1:n2
21    [a,d]=lwt_lab(D(:,n),P,U,int);
22    DD(:,n)=a;
23    DX(:,n)=d;
24 end
```

Program 14.4

```
1 function B=ilwt2d(AA, AD, DD, DX, P1, U1, int);
2 %Rekonstrukcja falkowa obrazów
3 % AA | DD
4 % ---------
5 % AD | DX
6
7 [n1 n2]=size(AA);
8 A=zeros(2*n1,n2); D=A;
9 B=zeros(2*n1,2*n2);
10 for n=1:n2
11    a=DD(:,n);
12    d=DX(:,n);
13    x=ilwt_lab(a(:),d(:),P1,U1,int);
14    D(:,n)=x(:);
15 end
16 for n=1:n2
17    a=AA(:,n);
18    d=AD(:,n);
19    x=ilwt_lab(a(:),d(:),P1,U1,int);
20    A(:,n)=x(:);
21 end
22 [n1 n2]=size(A);
23 for n=1:n1
24    B(n,:)=ilwt_lab(A(n,:),D(n,:),P1,U1,int);
25 end
```

Program 14.5

```
1 function x=ilwt_lab(c,d,P,U,int);
2 %odwrotna LWT
3 %P,U predyktor i update z funkcji lwt_lab
4 %int=1 transformacja całkowitoliczbowa
5
6 c=c(:);
7 d=d(:);
8 % odwrócenie uaktualnienia
9 if int==1
10    ce = c-round(lwt_step(d,U)/2);
11 else
12    ce = c-lwt_step(d,U)/2;
13 end
```

```
% odwrócenie predykcji
if int==1
   co = d+round(lwt_step(ce,P));
else
   co = d+lwt_step(ce,P);
end
% łączenie
N=2*length(c);
x = zeros(N,1);
x(1:2:N)=ce;
x(2:2:N)=co;
```

Zadanie 14.6

Korzystając z algorytmu rekonstrukcji DWT narysować funkcję skalującą oraz falkę horyzontalną, wertykalną i diagonalną.

15. Transformacja Radona

Zadanie 15.1
Za pomocą funkcji Matlaba `phantom` wygenerować model Sheppa i Logana przekroju głowy o rozmiarze 256 x 256 pikseli. Wyznaczyć transformatę Radona tego modelu funkcją Matlaba `radon`, a następnie zrekonstruować obraz za pomocą funkcji Matlaba `iradon`. Zaobserwować błąd rekonstrukcji dla różnych liczb projekcji transformacji Radona dla kąta Θ z przedziału [0, 180).

Zadanie 15.2
Napisać program do detekcji linii w obrazie wykorzystujący transformację Radona.

Rozwiązanie

Program 15.1

```
close all, clear all, clc
%% Syntetyczny obraz testowy zawierający linie
N=128;
I=zeros(N,N);
I(110,:)=ones(N,1); %linia pozioma
a=[0 0.4 -0.3 0.1 -0.5];
b=[120 20 N-10 3 N-2];
for k1=1:length(a);
    x=1:N; y=a(k1)*x+b(k1); y=round(y);
    for k2=1:N;
        if y<=N & y>0
            I(x(k2),y(k2))=1;
        end
    end
end
%% Obraz naturalny
% I = imread('cameraman.tif');
% h = fspecial('prewitt');
% h=h.';
% I = imfilter(I,h, 'replicate','same');
%% Transformata Radona
dte=1; theta = 0:dte:180-dte;
[R,xp] = radon(I,theta);
%% Detekcja linii tj. maksimów transformaty Radona
NL=8; %liczba linii
x=1:size(I,2); x=x-size(I,2)/2;
y=1:size(I,1); y=y-size(I,1)/2;
figure(1), hold on
   imagesc(theta,xp,R); colorbar
   title(['P_{\theta} (t), d\theta=' num2str(dte)]);
   xlabel('\theta'); ylabel('t'); axis tight
figure(2); hold on
   imagesc(x,y,I), colormap gray
   set(gca,'YDir','reverse')
   axis tight, box on, axis on
for k=1:NL
   [val, ind] = max(R(:));
   [n1,n2] = ind2sub(size(R),ind);
   % wyznaczenie parametrów prostej
   if theta(n2)==0 %prosta równoległa do OY
        x_line=ones(size(y))*xp(n1);
```

```
42          y_line=y;
43    elseif theta(n2)==90 %prosta równoległa do OX
44          y_line=-ones(size(y))*xp(n1);
45          x_line=y;
46    else
47          a=-tand(theta(n2)+90);
48          b=xp(n1)/sind(theta(n2));
49          y_line=a*x-b;
50          ind1=find(y_line<min(y) | y_line>max(y));
51          y_line(ind1)=[]; x_line=x; x_line(ind1)=[];
52    end
53    figure(1),plot(theta(n2),xp(n1),'ow','MarkerSize',15),
54    figure(2),plot(x_line,y_line,'g')
55    R(n1,n2)=0; %usunięcie aktualnego maksimum
56 end
```