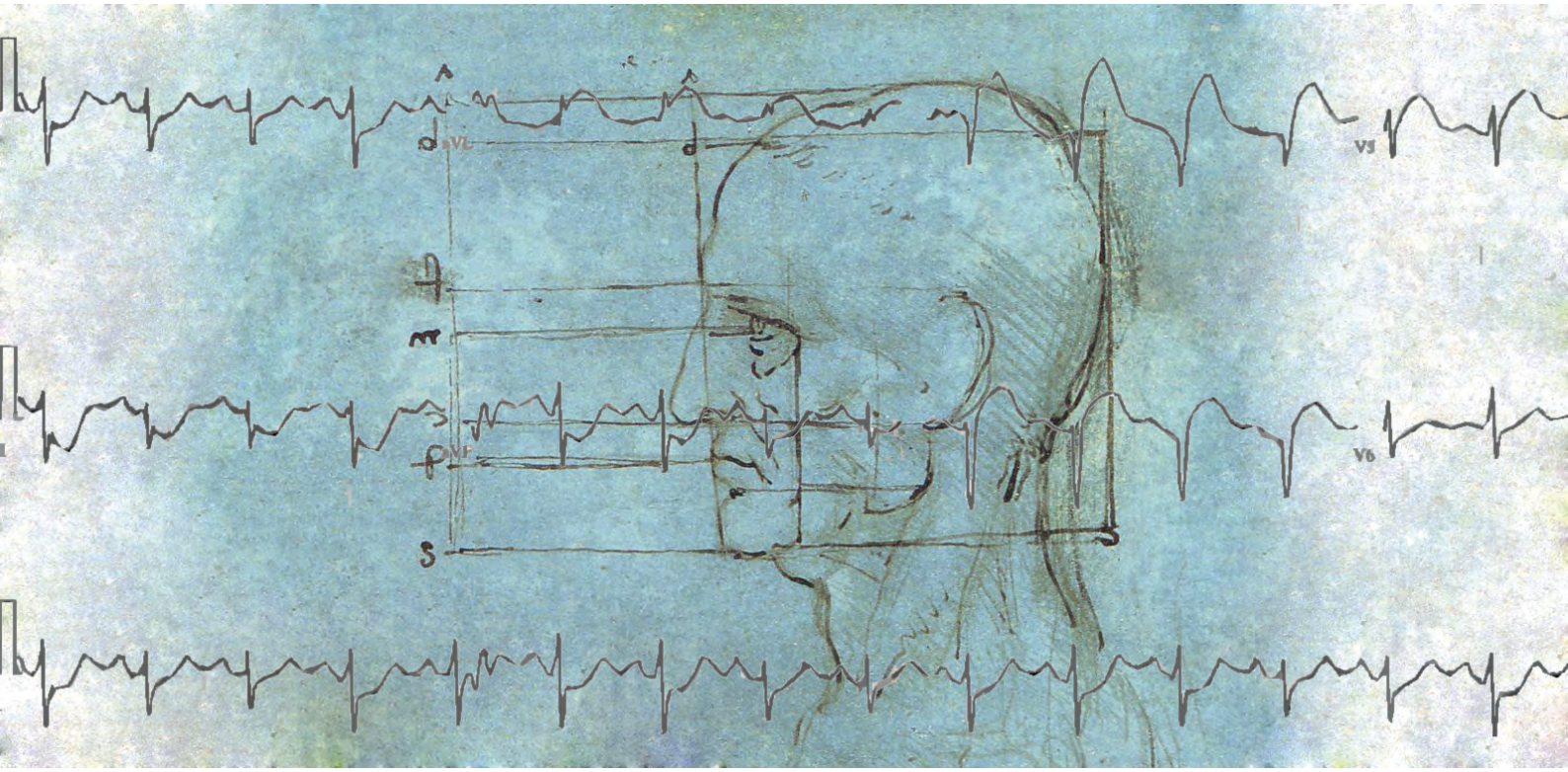


Laboratorium sygnałów medycznych

laboratorium

Marek Krośnicki

Gdańsk, 2013

Wydział Matematyki, Fizyki i Informatyki
Uniwersytetu Gdańskiego

Skrypt powstał w ramach projektu „Kształcimy profesjonalistów – kompleksowy program kształcenia skierowany do studentów, absolwentów oraz studentów niepełnosprawnych Uniwersytetu Gdańskiego” współfinansowanego przez Unię Europejską w ramach środków Europejskiego Funduszu Społecznego.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIWERSYTET GDAŃSKI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Marek Krośnicki

Laboratorium sygnałów medycznych

Materiały pomocnicze dla fizyki medycznej

Spis treści

Spis rysunków 2

1. Pakiet matlab 5

1.1. Podstawy użytkowania pakietu matlab 5

1.2. Graficzna prezentacja danych w matlabie 9

1.3. Elementy programowania w pakiecie matlab 15

2. Sygnały analogowe i dyskretne. 21

2.1. Podstawowe definicje 21

2.2. Przetwarzanie sygnału analogowego na cyfrowy. 23

3. Podstawowe metody analizy sygnałów. 27

3.1. Autokorelacja 33

3.2. Konwolucja 40

3.3. Interpolacja 44

4. Transformata Fouriera oraz widmo mocy 46

4.1. Filtrowanie i analiza widmowa sygnałów 50

4.2. Analiza widmowa rytmu serca. 67

5. Podstawy analizy obrazów 2D 72

5.1. Formaty zapisu obrazów dostępne w pakiecie MATLAB. 73

6. Nieliniowe metody analizy obrazów 2D. 87

7. Dodatki 88

Bibliografia 90

Spis rysunków

1.1 $f = \exp(\sin(x^2))$ 8
1.2 Rysunek $f(t) = \exp(-t^2)$, $f(t-3)$ 10
1.3 Rysunek $f(x,y) = \sin(x)\exp(-y^2)$ 11
1.4 Interfejs programu Plot Catalog 14
1.5 Wykres odchylania obliczonej wartości liczby π od wartości tablicowej w funkcji m. 20

2.1 (a) Sygnału analogowy (b) Sygnał dyskretny spróbkowany z odstępem czasowym $\pi/6$ (c) Sygnał dyskretny przetworzony na sygnał cyfrowy . . . 22
2.2 Wykres amplitudy dwukanałowego sygnału dźwiękowego w funkcji numeru próbki. 25

3.1 Fragment sygnału EKG z zanznaczonymi głównymi składowymi. 28
3.2 Tachogram dla sygnału RR k20_11.55.ce zawierającego $N = 20000$ punktów. 30
3.3 Histogram dla sygnału RR k20_11.55.ce z podziałem na 31 podzbiory. . . . 32
3.4 Wykres funkcji autokorelacji dla sygnału $y = 2\sin(2\pi\, 2t)$ próbkowanego z częstotliwością $f_s = 100$Hz w przedziale $t \in [0, 4]$. 35
3.5 Wykres funkcji autokorelacji dla losowego sygnału 36
3.6 Wykresy funkcji autokorelacji dla losowego sygnału harmonicznego poddanego różnemu stopniowi zaszumienia. 39
3.7 Układ LTI. 40
3.8 (a) Wykres sygnału wejściowego (b) Odpowiedź układu (c) sygnał wychodzący 43
3.9 Interpolacja funkcji sinus przeprowadzona metodami: liniową, kubiczną, splajnami, najbliższego punktu. 45

4.1 Sygnał okresowy w czasie 46
4.2 Analiza fourierowsa sygnały cyfrowego próbkowanego z częstotliwością $f_s = 1000Hz$. (a) składowa sygnału $y_1(t) = 1.5\sin(2\pi\, 2t)$, (b) składowa sygnału $y_2(t) = \sin(2\pi\, 4t)$, (c) sygnał złożony $y(t) = y_1(t) + y_2(t)$ (d) widmo mocy sygnału. 53
4.3 Sygnał $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$ o długosci (zaczynając od góry rysunku) T=10s, T=2s, T=1s, T=0.5s. 56
4.4 Widmo mocy dla sygnału $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$ o długosci (zaczynając od góry ryciny) T=10s, T=2s, T=1s, T=0.5s 57
4.5 Sygnał $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$ o długosci T=100s próbkowany z częstotliwościami (zaczynając od góry rysunku), (a) $f_s = 16$Hz, (b) $f_s = 8$Hz, (c) $f_s = 7$Hz, (d) $f_s = 5$Hz. 60
4.6 Widmo mocy sygnału $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$ próbkowanego z częstotliwością $f_s = 16$Hz, $f_s = 8$Hz, $f_s = 7$Hz $f_s = 5$Hz. 61
4.7 Powtarzający się periodycznie zdigitalizowany sygnał. Kolejne powtórzenia sygnału wyróżniono zmieniając linię ciągłą na przerywaną. 62
4.8 (a) Sygnał $y(t) = 1.5\sin(2\pi\, 2t - 1) + \sin(2\pi\, 4t - 1)$ o długosci T=10.7s próbkowany z częstotliwością $f_s = 100$Hz (b) Sygnał $y(t) = 1.5\sin(2\pi\, 2t - 1) + \sin(2\pi\, 4t - 1)$ przemnożony przez okno Hanninga 65
4.9 Widmo mocy dla sygnału (a) czystego (b) okienkowanego. 65
4.10 Charakterystyka okna Hanninga dla długości $N = 64$ 67
4.11 Periodogram dla sygnał RR 69
4.12 Periodogram Welcha dla sygnału RR z wykorzestaniem okna Hanninga o długości $N = 256$ przy 50% nakładaniu 70
4.13 Periodogramy Welcha dla róznych stopni nakładania się onkan Hanninga (N=256) 71

5.1 Konwersja obrazu RGB na skale szarości 76
5.2 Zmiana kontrastu obrazu oraz negatyw orginału 77
5.3 Periodogram dla sygnał RR 79
5.4 Zmiana mapy obrazu T2 81
5.5 Obróbka obrazu za pomocą funkcji imadjust 83

5.6 Histogram i transformacja 86

1. Pakiet matlab

1.1. Podstawy użytkowania pakietu matlab

Matlab jest pakietem programów, w którym można w prosty sposób wykonywać zaawansowane rachunki numeryczne. W pakiecie tym w intuicyjny sposób przeprowadza się operacje na całych wektorach oraz macierzach jak również na ich poszczególnych elementach.

Zdefiniujmy wektor $\vec{a} = [1, 2, 3, 4]$ w matlabie rozdzielając kolejne elementy wektora spacją

```
1  a=[1  2  3  4]
2  a =
3        1        2        3        4
```

W podobny sposób można zdefiniować macierze, przy czym należy pamiętać o tym, że separatorem wierszy jest średnik ;. Zapiszmy w matlabie macierz

$$A = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \\ 11 & 9 & 7 \end{bmatrix}$$

```
>> A=[ 2  4  6; 1  3  5; 11  9  7]
A =
     2     4     6
     1     3     5
    11     9     7
```

W matlabie aby uzyskać macierz transponowaną

$$A^T = \begin{bmatrix} 2 & 1 & 11 \\ 4 & 3 & 9 \\ 6 & 5 & 7 \end{bmatrix}$$

używamy operatora '

```
>> AT=A'
AT =
     2     1    11
     4     3     9
     6     5     7
```

Mnożenie macierzy realizujemy operatorem $*$.

Przykład:

```
>> B=[1  0  0; 0  0  1; 0  -1  0]
B =
     1     0     0
     0     0     1
     0    -1     0
>> C=A*B
C =
     2    -6     4
     1    -5     3
    11    -7     9
```

Mnożenie macierzy A przez macierz odwrotną B^{-1} wykonujemy używając operatora /

```
>> A/B
ans =
    2    6   -4
    1    5   -3
   11    7   -9
```

Chcąc wykonać działanie mnożenia, dzielenia lub potęgowania na poszczególnych elementach macierzy odpowiednie operatory poprzedzamy kropką (.∗ ./ .^). Działanie tych operacji zilustrujemy przemnażając elementy macierzy A przez elementy macierzy B

```
>>D= A.*B
D =
    2    0    0
    0    0    5
    0   -9    0
```

Można od razu zauważyć, że uzyskana macierz $D = A.*B$ jest różna od macierzy $C = A \cdot B$. Wykorzystując operatory „z kropką" możemy obliczać bez tworzenia pętli wartości funkcji dla wszystkich elementów wektora lub macierzy.

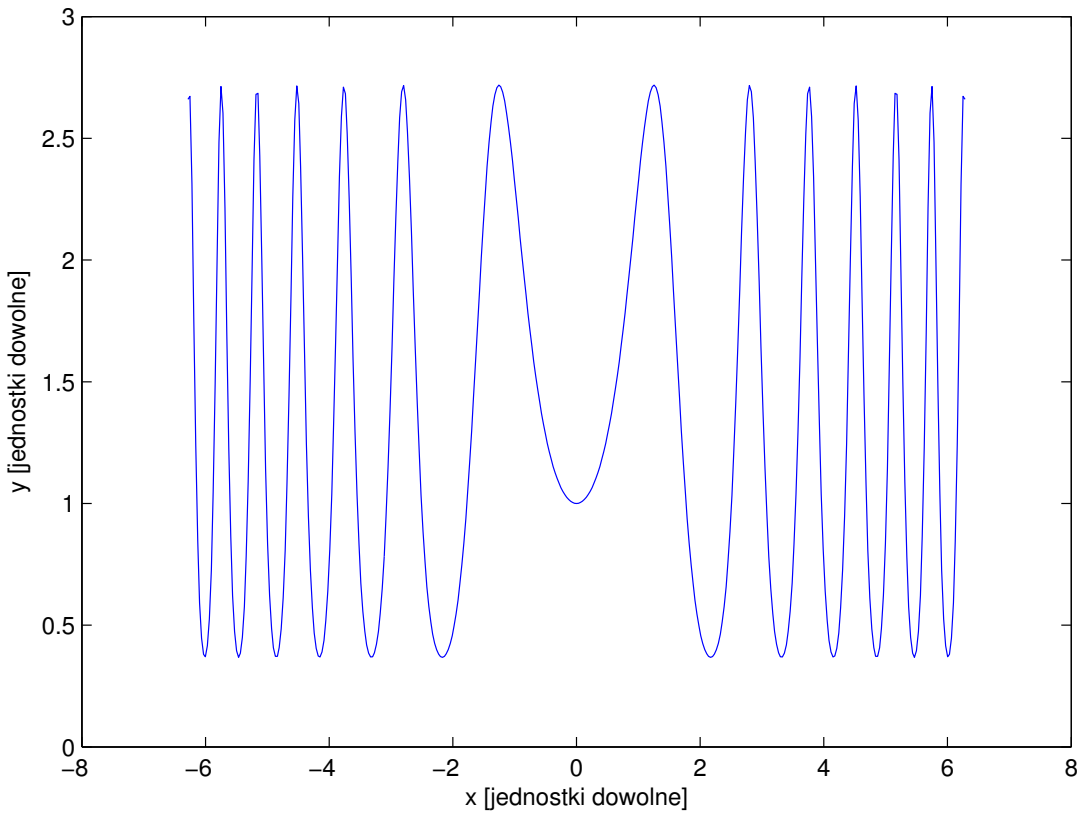
Przykład:

Policzmy dla elementów wektora $x = \left[-2\pi, \frac{-15\pi}{8}, \frac{-14\pi}{8}, \ldots, 0, \ldots, \frac{15\pi}{8}, -2\pi\right]$ funkcje $f(x_i) = \exp(\sin^2(x_i))$

```
>> x=-2*pi:1/8*pi:2*pi;
>> f=exp(sin(x.^2));
```

W powyższym przykładzie, kończąc linię średnikiem zawieszono wypisywanie wyniku działania komendy na ekran. Wektor x został wygenerowany za pomocą konstrukcji x=ograniczenie dolne:inkrement:ograniczenie górne. Sporządźmy wykres funkcji z poprzedniego zadania w przedziale dla $x \in [-2\pi, 2\pi]$, z wartościami wygenerowanymi co $1/100\pi$.

```
>> x=-2*pi:1/100*pi:2*pi;
>> f=exp(sin(x.^2));
>> plot(x,f)
>> xlabel('x [jednostki dowolne]')
>> ylabel('y [jednostki dowolne]')
```



Rysunek 1.1. $f = \exp(\sin(x^2))$

Skalę na osiach x, y można zmienić komendą axis([xmin xmax ymin ymax]), na-

tomiast opisy osi dodaje się komendami xlabel('opis osi x'), ylabel('opis osi y'). Dostęp do dokumentacji danej komendy można uzyskać wydając polecenie: doc nazwa_komendy (np. doc plot). Pisząc help nazwa_komendy dostajemy krótką informację o jej parametrach.

1.2. Graficzna prezentacja danych w matlabie

Po zapoznaniu się z podstawowymi zasadami korzystania z pakietu matlab przejdziemy do omówienia na przykładach wybranych możliwości graficznych pakietu matlab. W tabeli 1.1 zebrano podstawowe komendy służące do tworzenia wykresów

plot(x,y)	tworzy wykresy 2D
semilogy(x,y)	tworzy wykresy 2D z logarytmiczną skalą na osi rzędnych (y)
semilogx(x,y)	tworzy wykresy 2D z logarytmiczną skalą na osi odciętych (x)
loglog(x,y)	tworzy wykresy 2D z logarytmiczną skalą na oby osiach
polar(theta,ro)	tworzy wykres 2D we współrzędnych biegunowych
surf(x,y,z)	tworzy przestrzenny wykres 3D
contour(x,y,z)	tworzy wykres konturowy

Tabela 1.1. Wybrane komendy służące do sporządzania rysunków

Przykład 1.
Narysuj funkcję $f(t) = \exp(-t^2)$. Zaznacz gwiazdką punkty, w których była obliczona wartości funkcji. Na tym samym rysunku wykreśl przerywaną linią koloru czerwonego funkcję $f(t)$ opóźnioną w czasie o 3s.
Rozwiązanie:
Nałożenie jednego wykresu na drugi umożliwia komenda hold on, która wyłącza nadpisywanie aktywnego rysunku.

```
t=-3:0.1:6;  % Wybieramy  przedział
f=exp(-t.^2); % definiujemy funkcję
f3=exp(-(t-3).^2); % opóźniamy funkcje o 3 jednostki
figure  % otwieramy nowy rysunek
plot(t,f,'-bx')  % rysujemy funkcję f ciągłą linią '-'
% w kolorze niebieskim 'b' oznaczając punkty 'x'
hold on  % wyłącza nadpisywanie aktywnego rysunku
plot(t,f3,'--ro')  % rysujemy funkcję f3 linią przerywaną '--'
% w kolorze czerwonym 'r' oznaczając punkty 'o'
xlabel('t [s]');
ylabel('y [jednostki dowolne]');
```

Przykład 2.

Sporządź rysunek trójwymiarowy funkcji $f(x,y) = \sin(x)\exp(-y^2)$ w przedziale $x \in [-2\pi, 2\pi]$, $y \in [-2, 2]$

```
x=-2*pi:1/10*pi:2*pi; % Definiujemy wektor x
y=-2:1/10:2; % definiujemy wektor y
[Xm,Ym]=meshgrid(x,y); % tworzymy siatkę punktów Xm Ym
size(Xm) % Proszę zwrócić uwagę na wymiar macierzy Xm
size(Ym) % Proszę zwrócić uwagę na wymiar macierzy Xm
Zm=sin(Xm).*exp(-Ym.^2); % Obliczamy wartości funkcji
surf(Xm,Ym,Zm) % rysujemy wykres 3D
```

Przykład 3.

Sporządź wykres konturowy funkcji z poprzedniego przykładu $f(x, y) = sin(x) \exp(-y^2)$.

```
x=-2*pi:1/10*pi:2*pi; % Definiujemy wektor x
y=-2:1/10:2; % definiujemy wektor y
[Xm,Ym]=meshgrid(x,y); % tworzymy siatkę punktów Xm Ym
Zm=sin(Xm).*exp(-Ym.^2); % Obliczamy wartości funkcji
contour(Xm,Ym,Zm,30) % rysujemy wykres konturowy
```

Przykład 4.

Sporządź histogram dla liczb losowych normalnie rozłożonych wokół średniej 2.

```
r = 2 + randn(10000,1); % randn generuje liczby losowe (więcej ...
    informacji można znaleźć w helpie)
hist(r,20) % rysuje histogram z podziałem na 20 przedziałów
```

Matlab zawiera bardzo intuicyjne narzędzie graficzne (Plot Catalog) służące do tworzenia wykresów, którego działanie omówimy w poniższym przykładzie.

Przykład 4.

Wprowadzając komendy

```
x=[0:0.1:10];
y=[0:0.1:6];
[X,Y]=meshgrid(x,y);
Z=X.*Y;
```

wygenerujemy zmienne X, Y, Z, których zakresy i nazwy można znaleźć w okienku Workspace.

Jednocześnie trzymając klawisz ctrl oraz klikając myszką na nazwie zmiennej możemy w okienku Workspace wybrać dane, dla których chcemy utworzyć wykres. Następnie klikając prawym klawiszem myszki na zaznaczonych zmiennych wybieramy Plot Catalog i z jego menu wybieramy typ wykresu (patrz rysunek 1.4).

Dostęp do menu Plot Catalog można także uzyskać wykonując podobne operacje jak opisano powyżej na zmiennych w okienku Varible palety rysunku, którą wywołuje się komendami:

```
figure
```

```
figurepalette
```

1.3. Elementy programowania w pakiecie matlab

W matlabie zaimplementowano w pełni wartościowy język skryptowy. Skrypty są zapisywane w plikach tekstowych z rozszerzeniem .m. Pliki te potocznie zwane m-file'ami mogą być prostą serią matlabowskich komend, lub być funkcjami, które będą pracować na zmiennych lokalnych pobierając i zwracając wartości zadanych zmiennych do programu nadrzędnego. Struktura m-faile'a jest następująca:

— linia definiująca funkcję,

— linie z komentarzem pomocy,

— wnętrze funkcji zawierające komendy,

— linia końca funkcji.

Powyższą strukturę realizujemy następująco

```
function [zmienna_wyjściowa_1,zmienna ...
    wyjściowa_2,itd.]=nazwa_funkcji(zmienne_wjściowe,a1,a2,itd.)
% Tu mamy pierwszy wiersz pokazujący się w helpie
% Tu mamy drugi wiersz pokazujący się w helpie
   linia komend
   linia komend
      .
      .
      .
   linia komend
end
% Komentarz do procedury
```

M-file można tworzyć wydając polecenie edit nazwa_m-filea, lub za pomocą zwykłych edytorów dyskowych występujących w używanym przez nas systemie operacyjnym np. notepad, vim.

Przykład 1.

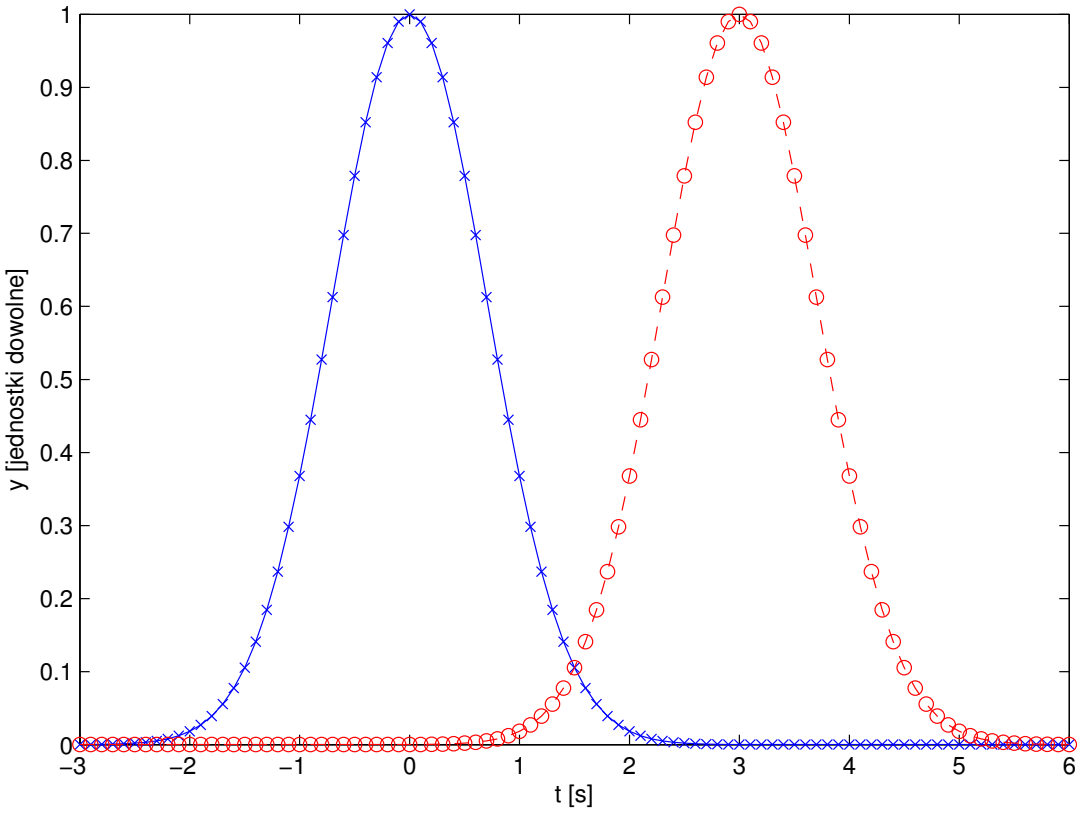
Napiszemy funkcję obliczającą objętość walca. Funkcja ta będzie nosić nazwę polew, będzie pobierać dwa argumenty: promień walca r oraz wysokość walca h. Funkcja zwróci wartość pola p.

```
function p=polew(r,h)
% Funkcja oblicza pole walca pobierając promień oraz wysokość
% r promień
% h wysokość
p=pi*r.^2.*h
end
% Nie wiem jak skomentować tak prostą funkcję
% Cytując klasyków najlepszym komentarzem będzie
% brak komentarza.
```

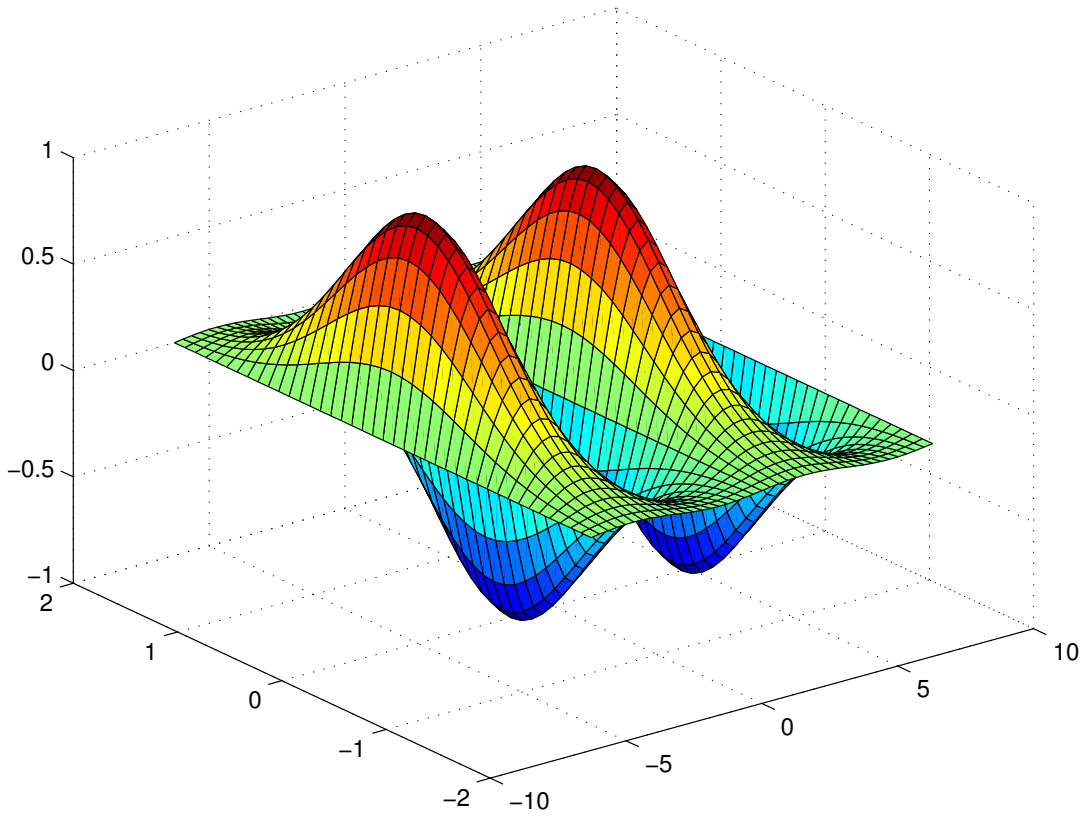
Matlab domyślnie szuka funkcji i skryptów zdefiniowanych przez użytkownika w bieżącym katalogu roboczym następnie je kompiluje. Dlatego pisząc skomplikowane programy należy pamiętać o zapewnieniu dostępu do zdefiniowanych uprzednio funkcji.

Zanim przejdziemy do pisania bardziej rozbudowanych procedur powinniśmy najpierw zapoznać się z podstawowymi komendami służącymi do kontroli przepływu danych[1]. Instrukcje te zebrano w tabeli 1.2. Większość wyrażeń z tabeli 1.2 sprawdza wartości logiczne wyrażeń (np. konstrukcje if *warunek*, while *warunek*) zawierające operacje relacji oraz operatory logiczne. Operacje relacji zestawiono w tabeli 1.3, a operatory logiczne zamieszczono w tabeli 1.4.

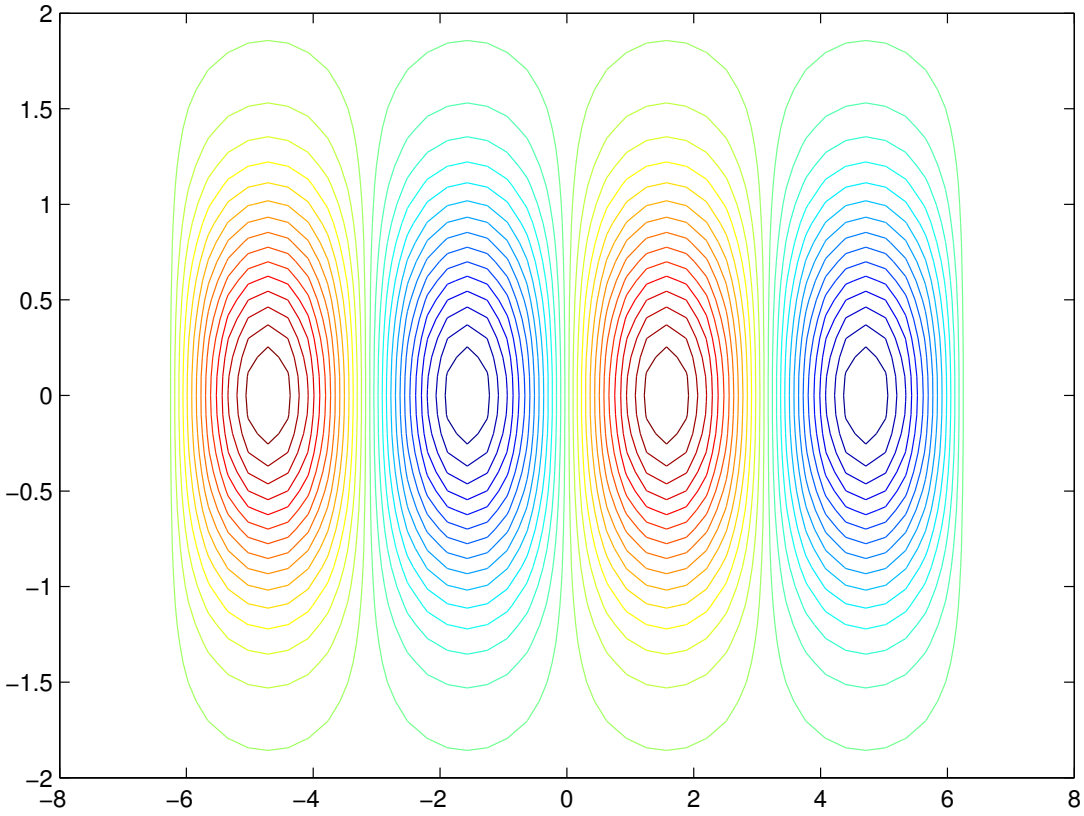
[1] Więcej informacji na temat powyższych komend można znaleźć w dokumentacji programu Matlab w rozdziale „Program Control Statements

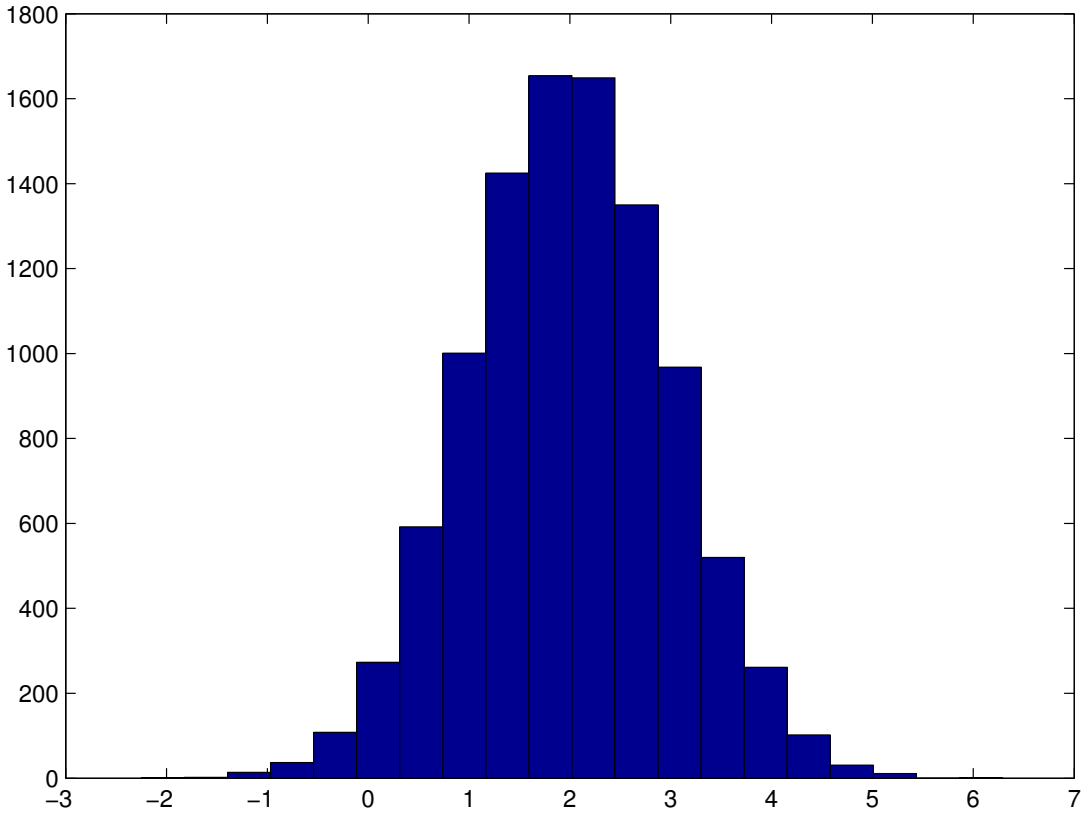


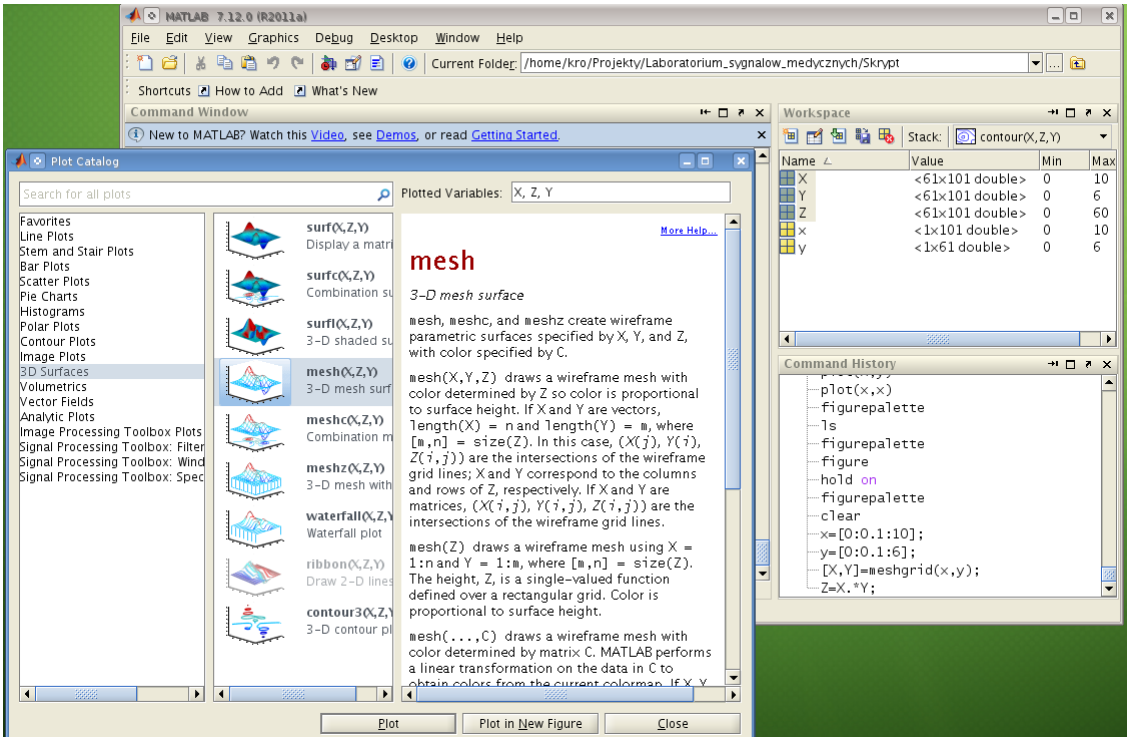
Rysunek 1.2. Rysunek$f(t) = \exp(-t^2)$, $f(t-3)$



Rysunek 1.3. Rysunek$f(x, y) = \sin(x) \exp(-y^2)$


2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−8
−6
−4
−2
0
2
4
6
8


1800
1600
1400
1200
1000
800
600
400
200
0
−3
−2
−1
0
1
2
3
4
5
6
7



Rysunek 1.4. Interfejs programu Plot Catalog

Komendy	Opis
for *index=strat:inkrement:stop* *Komendy* end	wykonuje komendy zawarte we wnętrzu for określoną ilość razy. Inkrement może przyjmować wartości całkowite jak i ułamkowe.
while *WARUNEK* *Komendy* end	wykonuje komendy zawarte we wnętrzu while dopóki *WARUNEK* jest prawdziwy.
break	Przerywa działanie pętli for oraz while. Działanie programu jest kontynuowane od pierwszej instrukcji znajdującej się poza pętlą
continue	powoduje pominięcie instrukcji znajdujących się w pętli za poleceniem continue i przejście do nowej iteracji
switch *wyrażenie_switch* *case wyr_case1* *Komendy* case {*wyr_case2,wyr_case3*} *Komendy* otherwise *Komendy* end	w przypadku gdy wartość wyrażenia *wyrażenie_switch* jest równa któremuś *wyr_case* wykonywane są komendy z odpowiedniego bloku case. W przypadku gdy wartość wyrażenia *wyrażenie_switch* nie zostanie znaleziona wśród *wyr_case* wykonane zostaną komendy z bloku otherwise.
if *Warunek_1* *Komendy* elseif *Warunek_2* *Komendy* elseif *Warunek_3* *Komendy* ⋮ *else* *Komendy* end	Sprawdza prawdziwość warunku a następnie wykonuje zestaw komend zawartych w odpowiednim bloku.

Tabela 1.2. Podstawowe komendy

Operacja	Objaśnienie
A < B	jeśli A jest mniejsze od B wynikiem jest prawda
A > B	jeśli A jest większe od B wynikiem jest prawda
A <= B	jeśli A jest mniejsze równe B wynikiem jest prawda
A >= B	jeśli A jest większe równe B wynikiem jest prawda
A == B	jeśli A jest równe B wynikiem jest prawda
A ∼= B	jeśli A jest różne od B wynikiem jest prawda

Tabela 1.3. Operacje relacji

Operator	Funkcja	Operacja	Priorytet
∼	not(*arg1*,*arg2*)	Negacja	Najwyższy
&	and(*arg1*,*arg2*	(AND) koniunkcja pomiędzy poszczególnymi elementami	
|	or(*arg1*,*arg2*)	(OR) alternatywa między poszczególnymi elementami	
&&		„szybka" koniunkcja (ang. short-circut AND)	
||		„szybka" alternatywa (ang. short-circut OR)	Najniższy

Tabela 1.4. Operatory logiczne.

Przejdziemy teraz do napisania kilku prostych programów, które będą ilustrować działanie poleceń zamieszczonych w powyższych tabelach.

Przykład 1.

Napisać program obliczający przybliżoną wartość liczby π. Skorzystaj z wrażenia $\pi \approx \sum_{k=0}^{m} \frac{(-1)^k}{2k+1}$.

Rozwiązanie:

```
clear % kasuje wszystkie zmienne z workspace
m=10000; % górne ograniczenie sumy
api=0; % Zeruje wartość początkową
for k=0:m
    api=api+4*(-1)^k/(2*k+1);
end
format long; % Zmienia format wydruku na długi
api
```

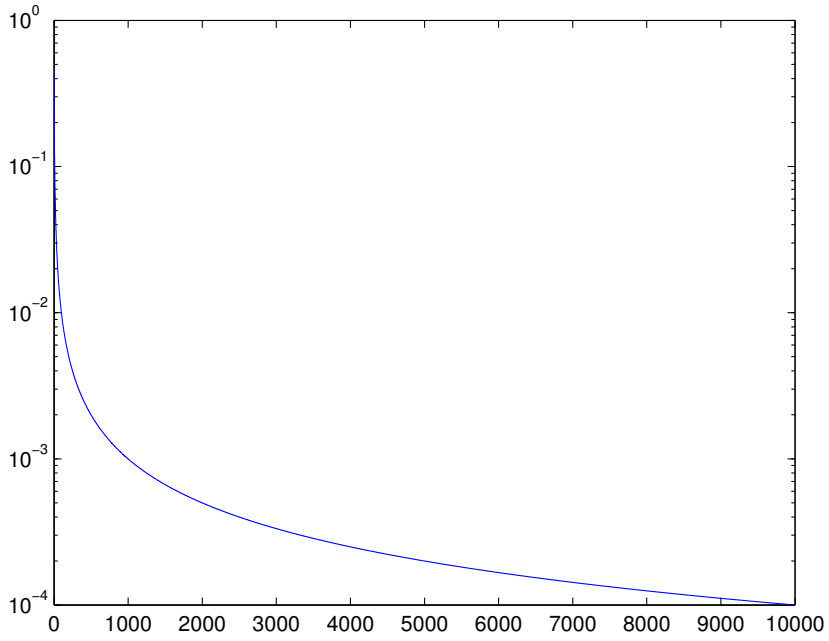
Przykład 2.

Zmodyfikuj poprzedni program tak, aby zbierać do tabeli apitab co setną przybliżoną wartość π. Utwórz wektor, w którym będzie się znajdować wartość bezwzględna odchylenia obliczanej przez nas wartości liczby π od wartości zdefiniowanej w Matlabie.

```
clear % kasuje wszystkie zmienne z workspace
m=10000; % górne ograniczenie sumy
api=0; %Zeruje wartość początkową
ii=0;
for k=0:m
    api=api+4*(-1)^k/(2*k+1);
    if mod(k,100)   % Jeśli k jest podzielne przez 100
        ii=ii+1;    % to wchodzimy do if i zwiększamy ii o 1
        apitab(ii)=api;
```

```
10            apierror(ii,1:2)=[k (abs(api-pi))];% Zapisujemy
11           % wierszami wartości k oraz |api-pi|
12        end
13    end
14    format long; % Zmienia format wydruku na długi
15    api % drukujemy wartość api
16    figure;
17    semilogy(apierror(:,1),apierror(:,2)); % tworzymy wykres błędu
18    %od górnego ograniczenia sumy w skali pół logarytmicznej
```

Z rysunku 1.5 możemy zauważyć, że błąd początkowo maleje bardzo szybko, przy m=2000 poprawa wyników jest już wolniejsza, przy $m = 10000$ błąd jest rzędu 10^{-4}.



Rysunek 1.5. Wykres odchylania obliczonej wartości liczby π od wartości tablicowej w funkcji m.

2. Sygnały analogowe i dyskretne.

2.1. Podstawowe definicje

W literaturze można znaleźć różne definicję sygnałów (patrz na przykład [5]), które są mniej lub bardziej formalne. Dla naszych potrzeb przyjmiemy dość luźną definicję.

Sygnałem nazwiemy informację o danym układzie, która może być zapisana w postaci funkcji zależnej od zmiennych rzeczywistych.

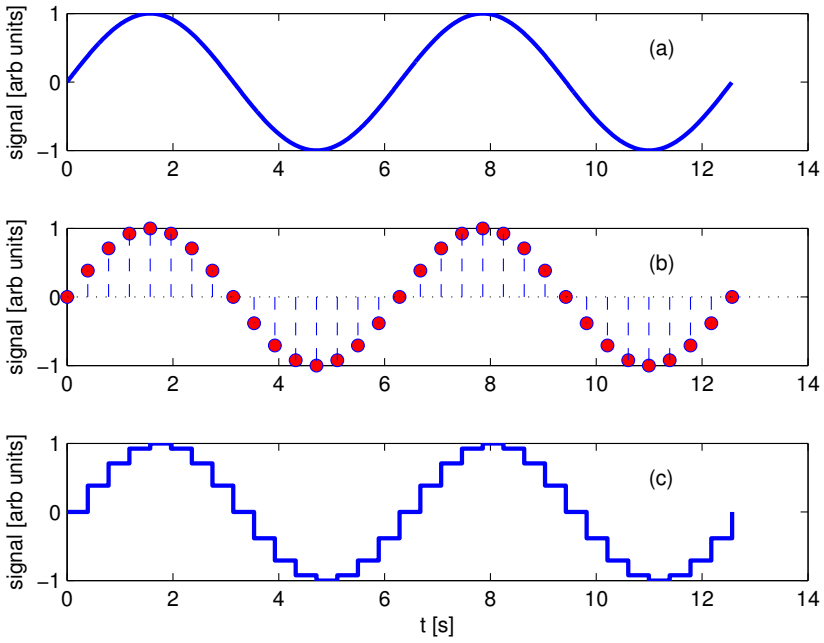
W naszym przypadku będziemy zajmować się sygnałami, które towarzyszą procesom fizjologicznym zachodzącym w organizmach żywych. Sygnały te mogą mieć różną postać, zależną od procesu fizjologicznego któremu towarzyszą. W przypadku układu krwionośnego może to być różnica potencjałów elektrycznych związanych z czynnością serca, bądź zmiany ciśnienia krwi. Oba wyżej wymienione sygnały można zapisać w postaci funkcji zależnej od czasu. Sygnały biologiczne mogą mieć również bardziej „chemiczny" charakter objawiający się pod postacią zmian stężeń hormonów bądź neurotransmiterów.

Badając sygnał biologiczny można znaleźć prawidłowości, które są charakterystyczne dla zdrowej populacji. Schorzenia z reguły prowadzą do procesów patologicznych mających swoje odbicie w zmianie charakterystyki sygnałów pochodzących z danego układu biologicznego. Odnotowywanie i analiza powyższych odchyleń od wartości przyjętych z normalne dostarcza pomocnych informacji dla dalszej diagnostyki.

Temperatura ciała jest jednym z najprostszych sygnałów medycznych, który od wieków stanowi jeden z podstawowych wskaźników stanu zdrowia pacjenta.

Z rozwojem techniki zmieniły się sposoby pomiaru temperatury: począwszy od jakościowej oceny za pomocą dłoni, a skończywszy na pomiarze prowadzanym z określonym interwałem czasowym za pomocą czujnika podczerwieni stosowanym na oddziałach intensywnej opieki medycznej.

W niniejszym skrypcie nie będziemy przedstawiać zjawisk związanych z fizjologią generacji sygnałów medycznych, dlatego zainteresowanych studentów zachęcam do zapoznania się z bardzo dobrym monograficznym podręcznikiem prof. Rangayyan [6], w którym omówiono między innymi mechanizm pompy sodowo-potasowej, potencjału spoczynkowego etc.



Rysunek 2.1. (a) Sygnału analogowy (b) Sygnał dyskretny spróbkowany z odstępem czasowym $\pi/6$ (c) Sygnał dyskretny przetworzony na sygnał cyfrowy

Wprowadźmy teraz formalne definicje:

Sygnał analogowy jest ciągłą funkcją czasu i dla każdego punktu czasu może przyjąć dowolną wartość należącą do continuum (patrz rys. 2.1a).

Sygnał dyskretny w czasie jest funkcją określoną tylko dla dyskretnych czasów przy czym funkcja ta dla danego czasu może przyjąć dowolną wartość należącą do continuum (patrz rys. 2.1b).

Sygnał cyfrowy jest funkcją określoną tylko dla dyskretnych czasów przy czym funkcja ta dla danego czasu może przyjmować dyskretne wartości należące do określonego przez zakres zbioru (patrz rys. 2.1c).

Sygnały medyczne takie jak ECG (elektrokardiogram), EEG (elekroencefalograf) są sygnałami ciągłymi w dziedzinie czasu, aby móc je zarejestrować w postaci cyfrowej należy je poddać procesowi przekształcania sygnału analogowego na cyfrowy.

2.2. Przetwarzanie sygnału analogowego na cyfrowy.

Kluczowym etapem przetwarzania sygnału analogowego na cyfrowy jest wybór jego dyskretnej reprezentacji. W przypadku sygnału zależnego od czasu wybiera się określoną częstotliwość f_s zwaną **częstotliwością próbkowania**, z którą zapisuje się w tablicy (o określonej długości słowa) kolejne następujące po sobie w odstępach czasowych $T_s = 1/f_s$ wartości sygnału analogowego. Obrazy analogowe przetwarza się do postaci cyfrowej przez przyjęcie odpowiedniej rozdzielczości definiującej ilość punktów przypadających na jednostkę długości danego obrazu.

Wierność odtwarzania sygnału analogowego przez sygnał cyfrowy jest ograniczona przez twierdzenie o próbkowaniu

Twierdzenie o próbkowaniu (Nyquist–Shannon)
Jeśli funkcja y(t) (sygnał analogowy) nie zawiera częstotliwości większych niż K hertców, to wtedy może być ona jednoznacznie zdefiniowana w postaci cyfrowej przez podanie ciągu rzędnych pobranych w odległych od siebie o 1/(2K) odstępach czasowych.

Przykład: W pliku `Toke_and_Exhale-Jaime_Os_Rivera-930234911.wav`[1] zapisany jest dźwięk towarzyszący wdechowi i wydechowi próbkowany z częstotliwością $44100Hz$. Komenda wavread pozwala na wczytanie do Matlaba pliku wav. W zmiennej strukturalnej readinfo znajdują się informacje o pliku między innymi dotyczące: autora pliku (Jaime Os Rivera), ilości kanałów (2), częstotliwość próbkowania (44100Hz), ilości bitów przypadających na jedną próbkę (16bit). Po wykreśleniu amplitudy sygnału dźwiękowego możemy zauważyć, że oba kanały dźwiękowe są identyczne.

```
% Wczytywanie plików wav
[breath,Fs1,nbits1,readinfo1]=wavread('Toke_and_Exhale-Jaime_Os_Rivera
-930234911.wav');
sound(breath,Fs1,nbits1)
subplot(2,1,1);plot(breath(:,1))
ylabel('amplitude left channel')
subplot(2,1,2);plot(breath(:,2))
breath_m=breath-breath;
xlabel('sample number')
ylabel('amplitude right channel')
```

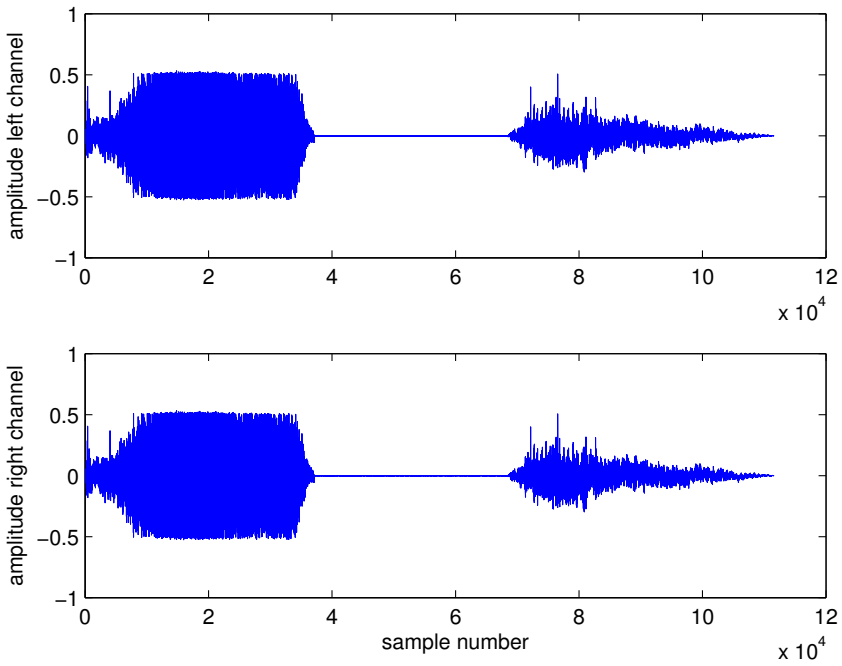
Częstotliwość próbkowania $44100Hz$ jest standardową częstotliwością zapisu danych na płytach CD audio. Z twierdzenia Nyquista wiemy, że przy takiej częstotliwości próbkowania odpowiednio przefiltrowanego sygnału analogowego poprawnie będą odtwarzane częstotliwości do $22050Hz$, co zapewnia nam pokrycie z drobnym nadmiarem całego zakresu słyszalności człowieka[2]

Na sygnałach w postaci cyfrowej można w prosty sposób wykonywać operacje arytmetyczne. Co pokażemy na przykładzie kilku próbek dźwiękowych

Przykład:

[1] plik pochodzi ze z serwisu `http://soundbible.com`

[2] Człowiek słyszy dźwięki od około $16Hz$ do około $20000Hz$.



Rysunek 2.2. Wykres amplitudy dwukanałowego sygnału dźwiękowego w funkcji numeru próbki.

W pliku sample.tar.gz znajduje się kilka próbek dźwięków. Złożymy z nich krótką scenkę muzyczną.

```
% Wczytywanie plików wav
[bass,Fs1,nbits1,readinfo1]=wavread('b_bc_04.wav');
[guitar,Fs1,nbits1,readinfo1]=wavread('HMRhyMix155-02.wav');
[breath,Fs1,nbits1,readinfo1]=wavread('Toke_and_Exhale-Jaime_Os_Rivera-930234911.wav'
%sound(breath,Fs1,nbits1)
[gun,Fs2,nbits2,readinfo1]=wavread('gun_battle.wav');
%sound(gun,Fs1,nbits1)
ngun=size(gun,1);
tgun=ngun/Fs2 ; % Czas trwania tgun ¬20s
nbreath=size(breath,1);
tbreath=nbreath/Fs1 % Czas trwania tbreath 2.5s
rep=int8(ngun/nbreath)
tmp=[];
tmp=breath;
for i=1:rep-1
      tmp=[tmp; breath];
end

samp=0.2*tmp(1:size(gun),:); % ściszenie dźwięku
samp=samp+0.1*gun+0.3*[zeros(size(samp,1)-size(guitar,1),1) ...
    zeros(size(samp,1)-size(guitar,1),1) ;guitar];
sound(samp,Fs1,nbits1)
```

Zanim przejdziemy do wniosków wynikających z twierdzenia o próbkowaniu, oraz omówienia efektów takich jak zjawisko przecieku czy też *aliasing*, będziemy musieli zapoznać się z podstawowymi wiadomościami dotyczącymi transformaty Fouriera oraz podstawowymi metodami analizy sygnałów.

3. Podstawowe metody analizy sygnałów.

Analiza sygnału zależnego od czasu może być prowadzona bezpośrednio w dziedzinie czasu, lub po przeprowadzeniu transformaty Fouriera w dziedzinie częstotliwości. W obu przypadkach można wykorzystać podstawowe metody znane z kursu ze statystyki, takie jak np. odpowiedniej liczenie średniej arytmetycznej, szukanie maksymalnych bądź minimalnych wartości etc. W niniejszym rozdziale będziemy zajmować się analizą sygnału w dziedzinie czasu na przykładzie sygnału biomedycznemu związanemu z czynnością ludzkiego serca.

Pracy serca towarzyszą zmiany potencjałów elektrycznych, które mogą być rejestrowane przy pomocy elektrokardiografu. W krzywej elektrokardiogramu (EKG) można wyróżnić kilka charakterystycznych powtarzających się w czasie elementów takich jak: załamek P, zespół QRS oraz załamek T. Odstęp pomiędzy kolejnymi pikami R nazywamy interwałem RR (ang. *RR interval*, patrz rysunek 3.1). Interwał RR wyznacza czasu, który upłynął pomiędzy kolejnymi uderzeniami serca co powoduje, że jest ściśle związany z pulsem i staje się jednym z podstawowych parametrów wykorzystywanych do oceny zmienności rytmu serca.

Puls zbierany na którejś z tętnic od bardzo dawna był jednym z podstawowych parametrów oceny stanu zdrowia organizmu ludzkiego. Szczególnie łatwo dostępnym do pomiaru parametrem jest średni puls oznaczany jako HR_{60}, który wyznacza się zliczając liczbę uderzeń przypadających na jedną minutę. Pomiar puls średniego i prowadzony w ściśle określonych warunkach przedstawia dużą wartość diagnostyczną. Dla przykładu Hr_{60} mierzony rano tuż po przebudzeniu

(Hr_{60} spoczynkowy) może być używany do kontroli stanu organizmu sportowca poddanego obciążeniom treningowym (poradź [**?**]). Podniesienie wartość Hr_60 spoczynkowego przez kilka dni z rzędu j o kilka lub kilkanaście uderzeń na minute, w odniesieniu do wartości średniej charakterystycznej dla danego osobnika może świadczyć o niewystarczającej regeneracji organizmu spowodowanej nadmiernym obciążeniem treningowy. Utrzymywanie w tej sytuacji wysokich obciążeń treningowych może doprowadzić do przetrenowania organizmu i kompletnego spadku formy.

Dużo bardziej wyrafinowane informacje o zmienności rytmu serca (HRV heart rate variability) możemy otrzymać szeregując kolejne następujące po sobie w sygnale EKG (angielski skrót *ECG*) normalne interwały RR w jednowymiarową tablicę. Wykres tej tablicy funkcji czasu nazywa się **tachogramem**.

Przykład 1.

W pliku RR.ce_2012.zip[1] zawarte są sygnały RR zebrane od osób zdrowych. Weźmy zbiór k20_11.55.ce składający się z $N = 20000$ punktów i utwórzmy z zawartych w nim danych tachogram. Używając metod statystycznych wyznaczymy podstawowe parametry sygnału w funkcji czasu oraz sporządzimy histogram.

```
clear all
close all
ecg=load('k20_11.55.ce');
ecgh=ecg/(60*60*1000); % przeliczamy czas z milisekund na godziny
t(1)=ecgh(1);
% W pętli sumujemy odstepy czasu RR tak
% aby wygenerować wektor który naniesiemy na oś czasu
for i=1:length(ecg)-1
    t(i+1)=t(i)+ecgh(i+1);
```

[1] Dane sygnałów RR zawarte w archiwum są udostępnione do ćwiczeń dzięki uprzejmości prof. dr hab. Danuty Makowiec

```
10 end
11 %tachogram w funkcji czasu
12 figure
13 plot(t,ecg);
14 xlabel('time [h]'); ylabel('RR (ms)');
15 axis([0 inf  400 1600]);
16 % Wyciągamy podstawowe informacje statystyczne
17 ecgmin=min(ecg); % minimalny interwał RR
18 ecgmax=max(ecg); % maksymalny interwał RR
19 ecgaverage=mean(ecg); % Średni interwał RR
20 ecgstd=std(ecg); % odchylenie standardowe RR
21
22 figure
23 hist(ecg,31); % Sporządzamy histogram
24 xlabel('RR interval (ms)'); ylabel('Number of samples');
```

Tachogram zamieszczony na rysunku 3.2 przedstawia zmienność pracy serca w czasie prowadzonego przez około 5.5 godziny pomiaru. Sygnał RR cechuje począwszy od 1.3h się dużą zmiennością. Poniżej w punktach zebraliśmy podstawowe parametry tachogramu:

— najkrótszy interwał RR ma długość ecgmin=500ms,

— najdłuższy interwał jest równy ecgmax=1469ms,

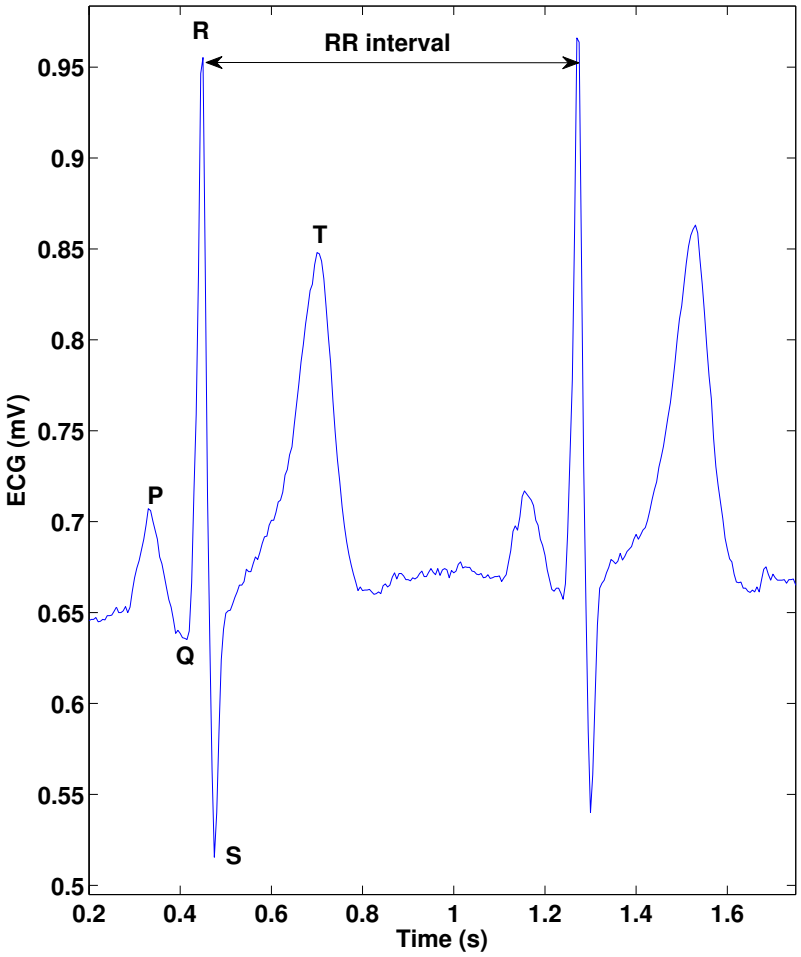
— wartość średnia interwału RR jest równa ecgaveradge=978ms,

— odchylenie standardowe jest równe ecgsdt=143ms.

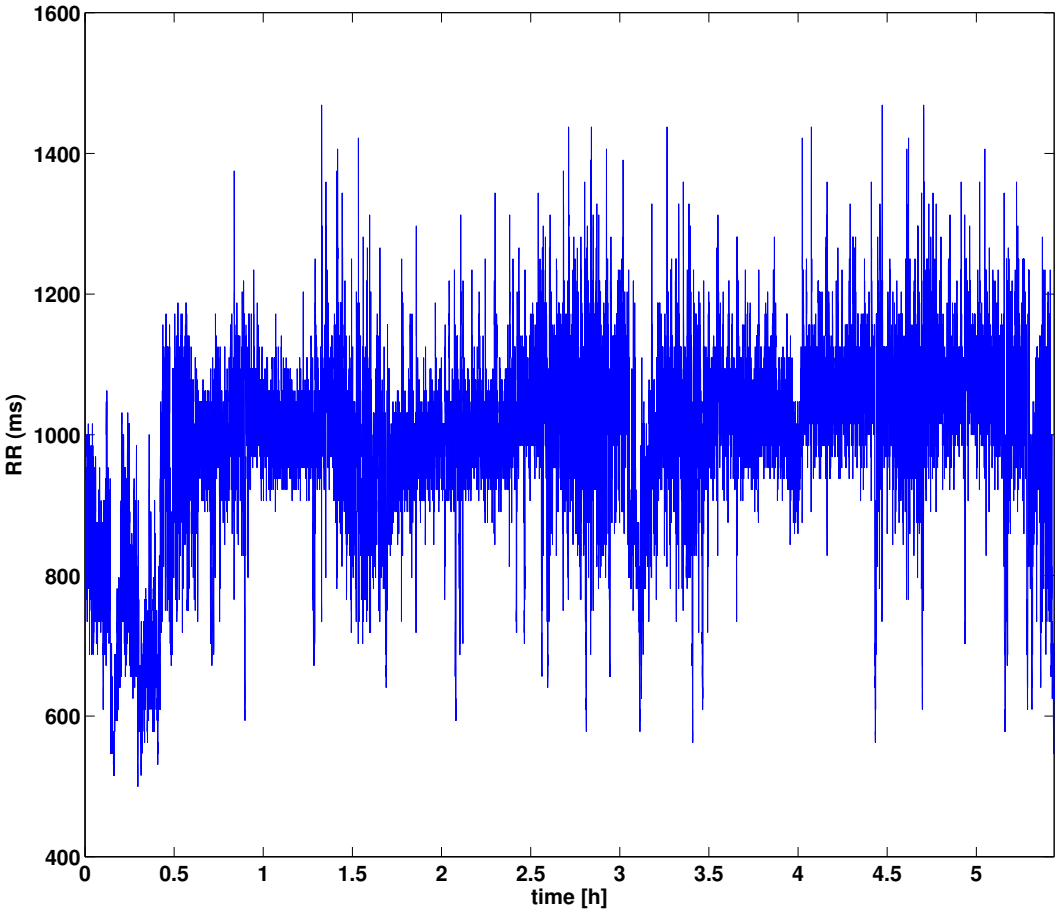
Wartości minimalna i maksymalna amplitudy sygnału są wyznaczone obiektywnie i zrozumienie ich znaczenia nie nastręcza kłopotu, natomiast ostrożnie należy postępować interpretując wartość średniej arytmetycznej i nie należy jej mylić z najliczniej występującymi wartościami w sygnale.

Histogram dla sygnału k20_11.55.ce został przedstawiony na rycinie 3.3. Stanowi on proste jednakże bardzo pomocne narzędzie przy obróbce sygnału. Przedstawia on graficznie liczbę elementów zbioru wpadających do tej samej klasy określonej przez długość przedziału. Analizując rysunek 3.3 możemy spostrzec, iż

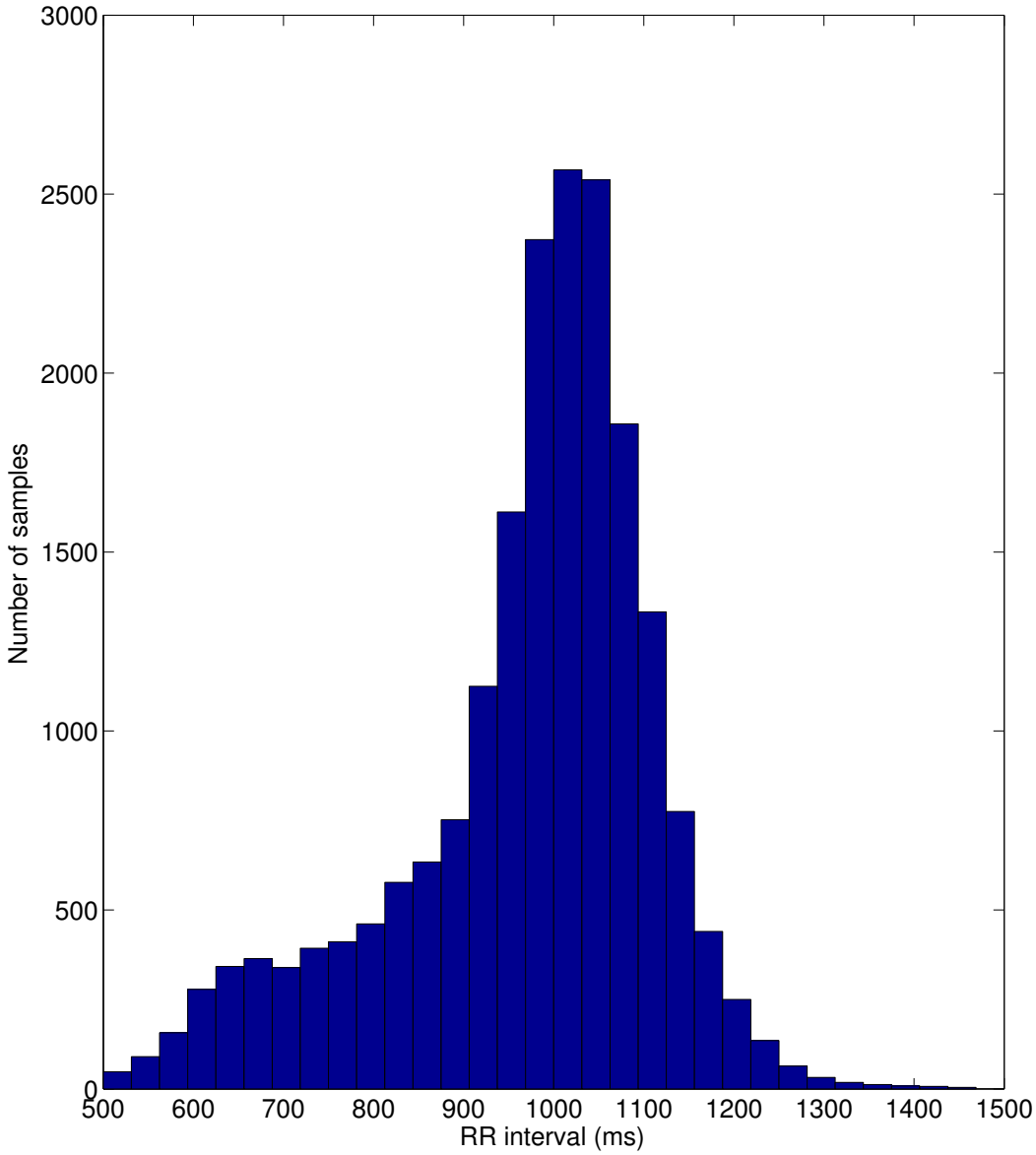
najczęściej w sygnale RR występują interwały których czas trwania jest 1100ms a rozkład liczby próbnej nie jest symetryczny wobec wartości średniej arytmetycznej. W szeregu RR k20_11.55.ce kraniec skrzydła odpowiadające małym interwałom znacznie większy niż kraniec skrzydła odpowiadającego długim interwałom. Wobec tego w tym przypadku należy dość sceptycznie podejść do użyteczności odchylenia standardowego.



Rysunek 3.1. Fragment sygnału EKG z zanznaczonymi głównymi składowymi.



Rysunek 3.2. Tachogram dla sygnału RR k20_11.55.ce zawierającego $N = 20000$ punktów.



Rysunek 3.3. Histogram dla sygnału RR k20_11.55.ce z podziałem na 31 podzbiory.

3.1. Autokorelacja

Dla sygnałów ciągłych spełniających warunki ergodyczności[2] funkcję autokorelacji (poradź również Hussain [5]) definiuje się poprzez całkę

$$R_{xx}(t) = \frac{1}{T_0} \int_0^{T_0} x(\tau)x(\tau + t)\, d\tau, \tag{3.1}$$

natomiast dla sygnałów dyskretnych całkowanie po przedziale czasu T zostaje zastąpione sumowaniem. W pakiecie MATLAB dla sygnałów o długości N do obliczania funkcji autokorelacji wykorzystuje się następujący wzór

$$R_{xx}(m) = \begin{cases} \sum_{n=0}^{N-m-1} x(n+m)x(k) & m \geqslant 0, \\ R_{xx}(-m) & m < 0. \end{cases} \tag{3.2}$$

Funkcja autokorelacji pozwala sprawdzić jak bardzo sygnału w wybranej przez nas chwili czasu jest podobny do tego samego sygnału przesuniętego w czasie o t. Stosując autokorelacje możemy wychwytywać podobieństwa, które występują w zmieniającym się w czasie sygnale. Funkcja autokorelacji jest funkcją parzystą, która zawsze przyjmuję największą wartości dla $t = 0$, co odpowiada zerowemu przesunięciu. Jeśli mamy do czynienia z sygnałem będącym białym szumem, czyli z sygnałem który ze swej definicji jest całkowicie losowy i nie wykazuje samo podobieństwa wraz z upływem czasu, to wtedy funkcja autokorelacji zbliża się kształtem do delty Diraca. Wykorzystując tą własność autokorelacji możemy badać stopień zaszumienia sygnału.

Funkcja autokorelacji jest szczególnym przypadkiem funkcji korelacji wzajemnej zdefiniowanej wzorem

$$R_{xy}(t) = \frac{1}{T_0} \int_0^{T_0} x(\tau)y(\tau + t)\, d\tau, \tag{3.3}$$

która pozwala na badanie wzajemnego podobieństw występujących w dwóch różnych sygnałach $x(t)$ oraz $y(t)$.

[2] dla sygnałów ergodycznych średnia po ansamblu jest równa średniej po czasie. [5]

W pakietach MATLAB oraz Octave funkcję autokorelacji wywołuje się komendą[3] [var,m]=xcorr(x,x) gdzie wektor x jest badanym sygnałem, var jest wektorem zawierającym wartości funkcji autokorelacji, n jest wektorem przesunięć. Jeśli długość próbkowanego sygnału wynosi N to zakres wektora m wynosi od $-N+1$ do $N-1$. Wektor przesunięć może łatwo przeliczyć na czas przesunięcia.

Przykład 2.

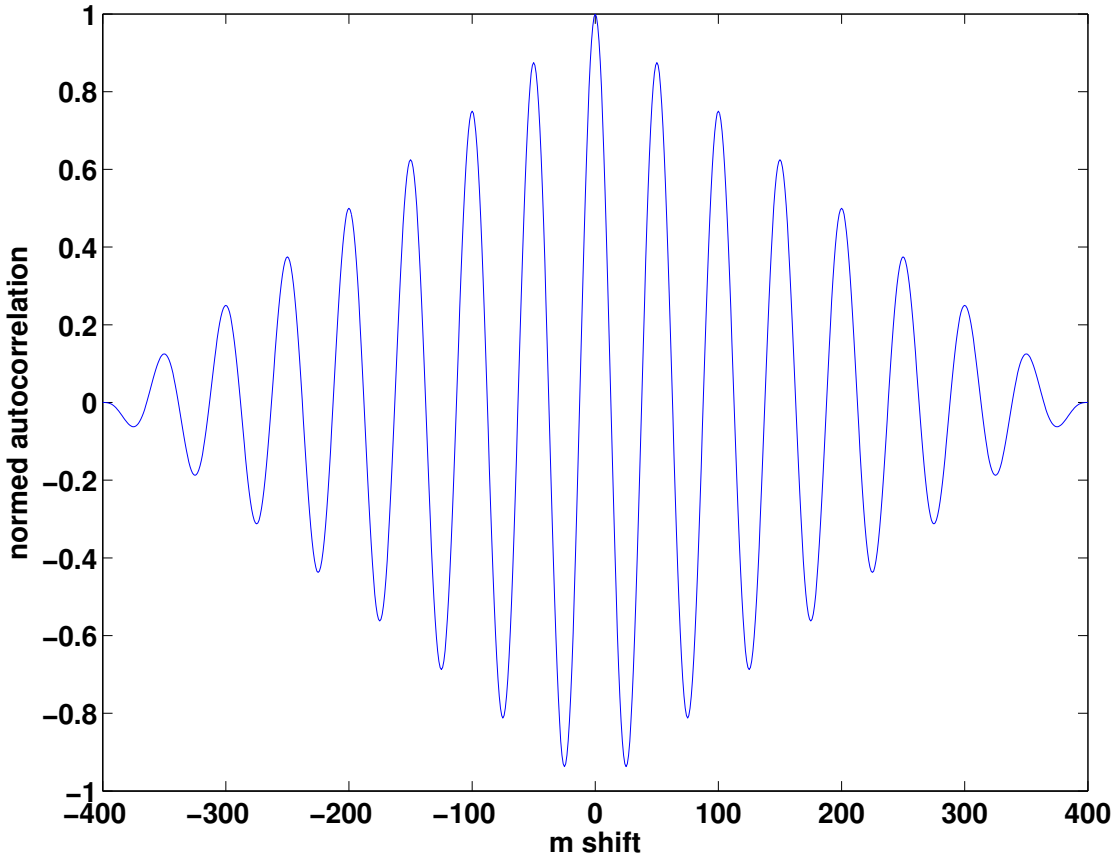
Następujące zdanie jest zawsze prawdziwe:

Funkcja autokorelacji obliczona dla nieskończenie długiego sygnału okresowego jest również funkcją okresową.

Pisząc prosty program sprawdzimy czy powyższe zdanie jest również prawdziwe dla skończonego sygnału.

```
close all;
clear all;
T=4; % Długość sygnału
dt=0.01;
t=0:dt:T;
y=2*sin(2*pi*2*t);
[vec,tt]=xcorr(y,y);
am=max(vec);
plot(tt,vec/abs(am));
xlabel('n shift');
ylabel('normed autocorrelation');

[vec,tt]=xcorr(y,300,'coeff');  % Powyższa komenda
% oblicza unormowaną autokorelację w przedziale
% -300 do 300 próbek
```

[3] funkcję korelacji wzajemnej wywołuje się taka samą komendą xcorr tym, że jako argumenty pobiera się dwa różne sygnały x, y



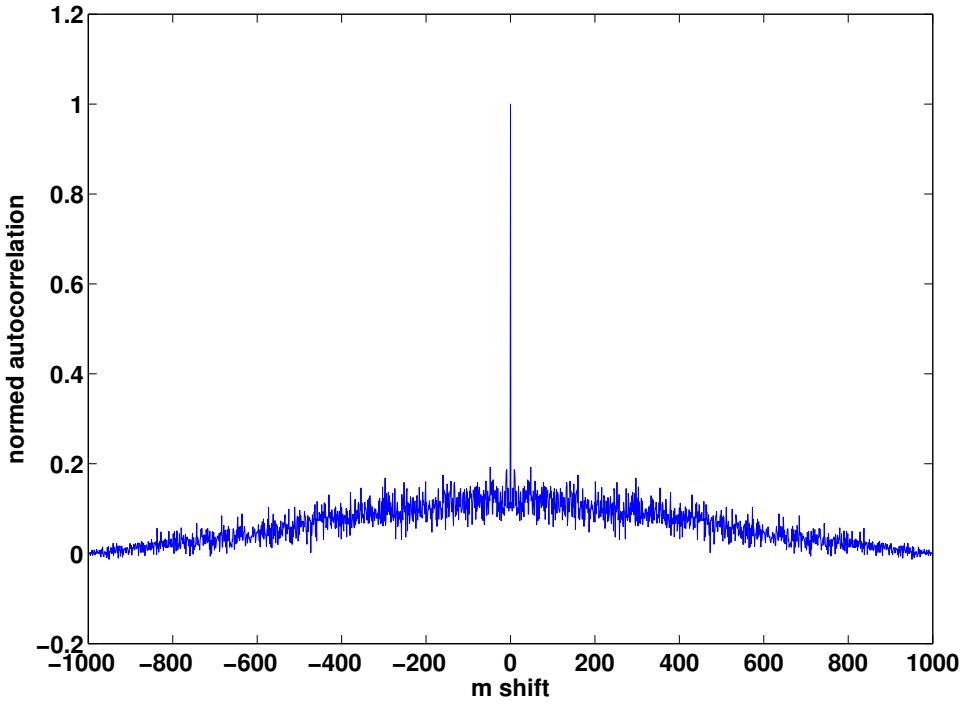
Rysunek 3.4. Wykres funkcji autokorelacji dla sygnału $y = 2\sin(2\pi\, 2t)$ próbkowanego z częstotliwością $f_s = 100$Hz w przedziale $t \in [0, 4]$.

Funkcja autokorelacji obliczona za pomocą wzoru 3.2 dla sygnału okresowego $y = 2\sin(2\pi\, 2t)$ jest funkcją, która zmienia swoje wartości cyklicznie jednak nie jest okresowa. Powolne zmniejszanie się wartości funkcji autokorelacji wraz ze wzrostem przesunięcia m jest konsekwencją tego, że w górnej granicy sumowania liczymy różnicę $N - m - 1$ zmniejszając z każdym krokiem liczbę sumowanych wyrazów.

Przykład 3.

Wygeneruj szum losowy oraz oblicz i wykreśl jego funkcję autokorelacji

```
close all;
clear all;
a=-2; % dolna granica przedziału
b=3; % górna granica przedziału
y = a + (b-a).*rand(1000,1); % generujemy
% tysiąc punktów szum losowy
% obliczamy autokorelację
[vec,tt]=xcorr(y,y);
%znajdujemy maksimum
am=max(vec);
%normujemy i rysujemy funkcję autokorelacji
plot(tt,vec/abs(am));
xlabel('m shift');
ylabel('normed autocorrelation');
```



Rysunek 3.5. Wykres funkcji autokorelacji dla losowego sygnału

Zgodnie z oczekiwaniami obliczona przez na funkcja autokorelacji dla sygnału losowego ma istotne wartości jedynie dla nieprzesuniętego sygnału $m = 0$. Wykres z rysunku 3.5 przypomina swym kształtem złożenie delty Kroneckera z drobnymi fluktuacjami wokół osi odciętych.

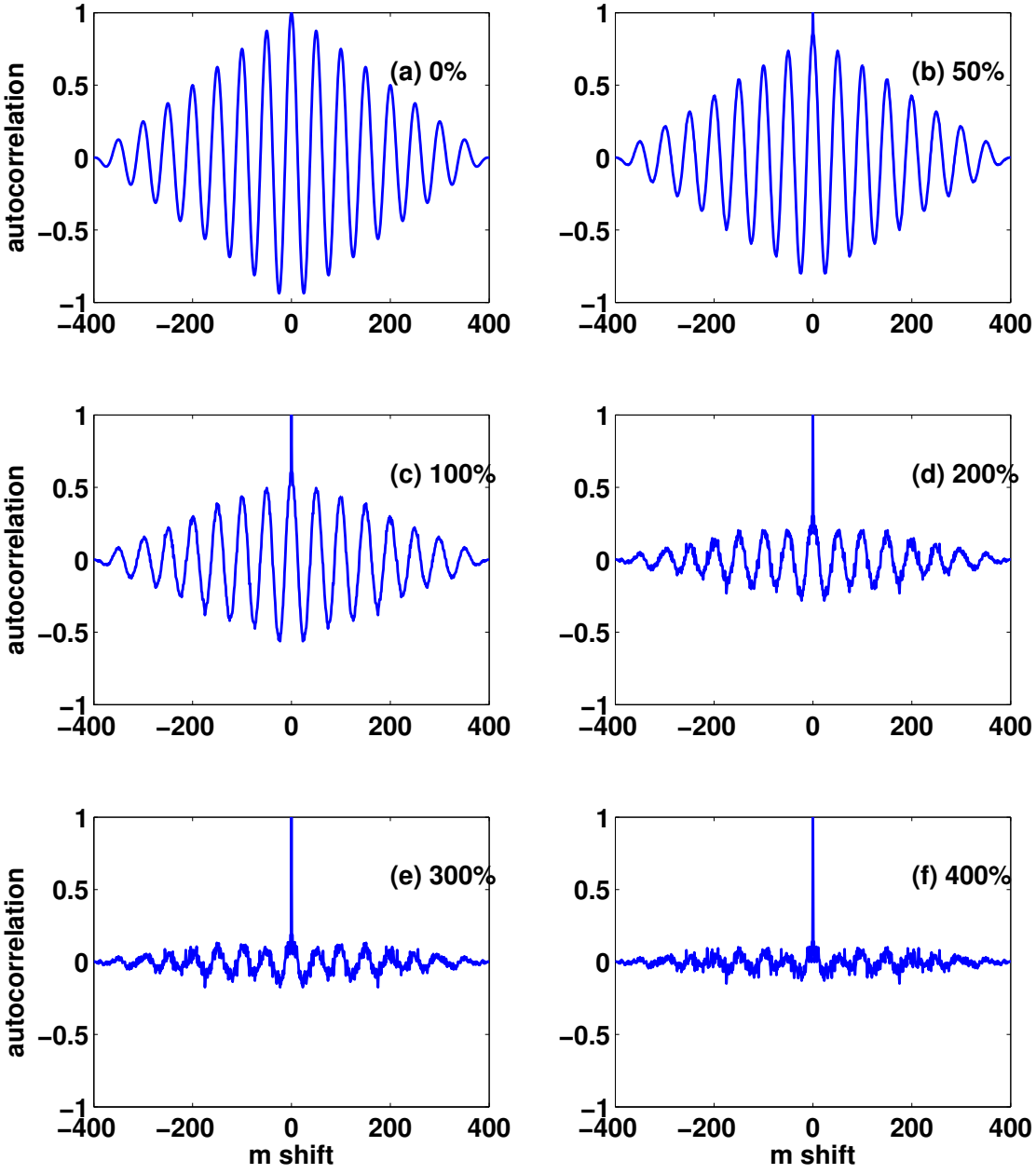
Przykład 4.

Do sygnału okresowego $y = 2\sin(2\pi\, 2t)$ dodaj różne wartości szumu i dla każdego przypadku sporządź wykres funkcji autokorelacji i skomentuj otrzymane wyniki.

```
close all;
clear all;
T=4; % Długość sygnału
dt=0.01;
t=0:dt:T;
% generujemy sygnał okresowy
y=2*sin(2*pi*2*t);
M=length(y); % sprawdzamy długość wektora y
a=-2; % dolna granica przedziału
b=2; % górna granica przedziału
noise= a + (b-a).*rand(M,1); % generujemy
% sygnał losowy
%ustalamy procent szumu w całkowitym sygnale
percent=[0 50 100 200 300 400 500 600];
%generujemy etykiety do rysunku
label=['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' ]  ;
% normujemy poziom szumu
noise_level=percent/100;
% Tworzymy pętlę przebiegającą przez wszystkie zakresy szumu
for i=1:length(noise_level)
    tt=[];
    vec=[];
   % składamy sygnał zaszumiony
    yn=y+noise_level(i)*noise';
```

```
25      %liczymy unormowaną funkcję autokorelacji z maksymalnym
26      %przesunięciem 400
27      [vec,tt]=xcorr(yn,400,'coeff');
28      subplot(4,2,i); plot(tt,vec,'LineWidth', 1.5);
29      text(200,0.6,['(',label(i),') ',num2str(percent(i)),'%']);
30      % generujemy etykiety na oś rzędnych dla i nie parzystych
31      if(mod(i,2))
32        ylabel('autocorrelation');
33      end
34       % generujemy etykiety na oś odciętych dla ostatniej oraz
35       % przedostatniej wartości i
36      if(i==(length(noise_level)-1) || i==length(noise_level))
37            xlabel('m shift');
38      end
39  end
```

Działanie powyższego skryptu zostało zilustrowane rysunkiem 3.6, na którym zamieszczono sześć paneli przedstawiających funkcję autokorelacji sygnału harmonicznego $y = 2\sin(2\pi 2t)$, poddanego różnemu stopniowi zaszumienia, począwszy od sygnału bezszumowego (panel (a)), a skończywszy na sygnale, w którym maksymalna amplituda szumu była czterokrotnie większa od amplitudy sygnału harmonicznego (panel (f) oznaczenie zaszumienia 400%). Analizując rysunek 3.6 możemy zauważyć, że 50% zaszumienie sygnału spowodowało pojawienie się małego piku przy m=0, nie zmieniając przy tym znacznie amplitud oscylacji w stosunku do funkcji autokorelacji sygnału nie zaszumionoego y. Przy poziomie szumu 100% i 200% pojawia się wyraźne maksimum dla m=0 jednakże oscylacje krzywej autokorelacji są wyraźnie widoczne. Przy poziome szumu 400% wykres funkcji autokorelacji przedstawionej na panelu (f) został zdominowany przez jeden pik dla m=0, który świadczy o bardzo wysokim zaszumieniu sygnału. Przy odrobinie dobrej woli, wciąż jednak można dopatrzyć się delikatnych oscylacji będących pochodną drobnego wkładu składowej harmonicznej do całkowitego sygnału.



Rysunek 3.6. Wykresy funkcji autokorelacji dla losowego sygnału harmonicznego poddanego różnemu stopniowi zaszumienia.

3.2. Konwolucja

Przez pewien czas zajmiemy się układami liniowymi niezmienniczymi w czasie (ang. *Linear Time-invariant system* w skrócie LTI), które posiadają własności:

— czasową niezmienniczość, którą rozumiemy następująco:

Jeśli sygnał wejściowy $x(t)$ daje na wyjściu układu sygnał $y(t)$, to wtedy dla dowolnego przesunięcia czasowego $\tau \in \Re$ sygnał wejściowy $x(t+\tau)$, daje na wyjściu przesunięty sygnał wyjściowy $y(t+\tau)$.

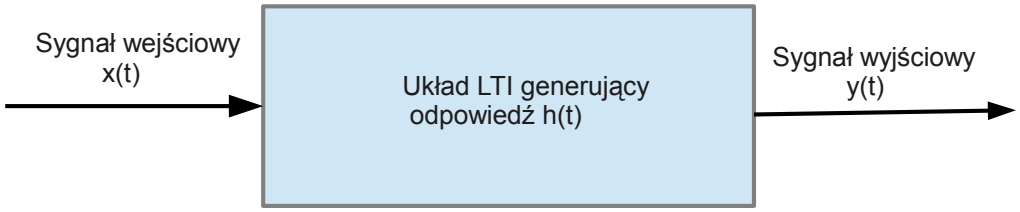
— liniowość rozumianą jako

Jeśli sygnał wejściowy $x_1(t)$ daje sygnał wyjściowy $y_1(t)$ oraz sygnał wejściowy $x_2(t)$ daje na wyjściu $y_2(t)$ to wtedy dla dowolnych stałych $a_1, a_2 \in \Re$ sygnał wejściowy $a_1x_1(t) + a_2x_2(t)$ daje na wyjściu sygnał $a_1y_1(t) + a_2y_2(t)$.

Układy LTI z reguły charakteryzuje się przez podanie sygnału wejściowego $x(t)$ oraz odpowiedzi układu $h(t)$, które dają wystarczającą ilość informacji do tego, aby licząc konwolucję (ang *convolution*)

$$y(t) = h(t) * x(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)\,d\tau \tag{3.4}$$

wyznaczyć sygnał wyjściowy $y(t)$ (patrz rysunek 3.7). Zdefiniowaną w ten sposób



Rysunek 3.7. Układ LTI.

konwolucję nazywa się w polskiej literaturze splotem. Dla wejściowego dyskret-

nego sygnału $x(n)$ długości N_1 oraz odpowiedzi układu $h(n)$ długości N_2 splot definiuje się następująco

$$y(n) = h * y = \sum_{k=1}^{N} h(k)x(n-k) \tag{3.5}$$

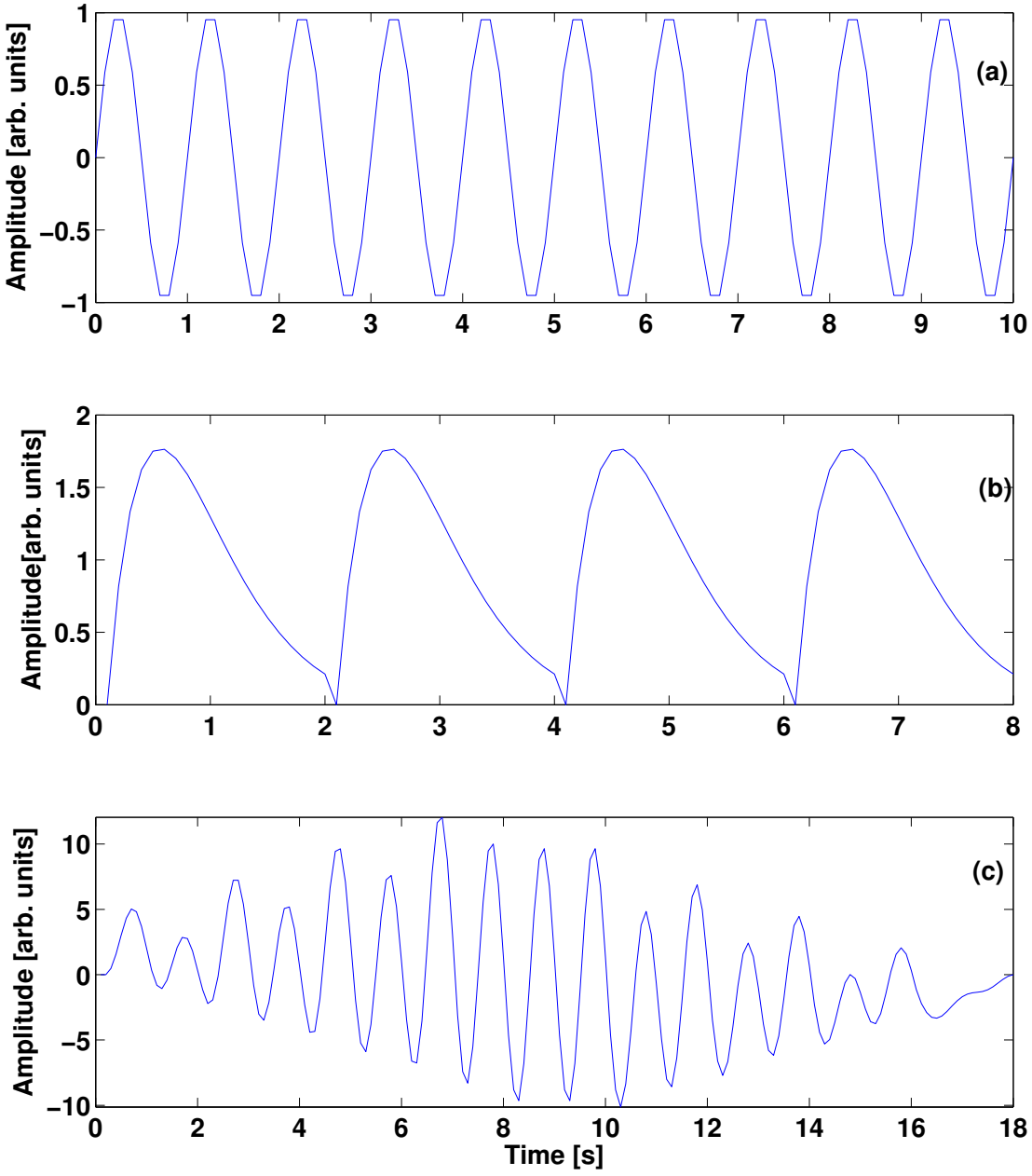
gdzie $N = \min(N_1, N_2)$. Długość $y(n)$ jest równa $L = N_1 + N_2 - 1$.

Przykład 5. W pakiecie MATLAB konwolucję obliczą się komendą conv(h,x). Przy użyciu tej komendy oblicz kształt sygnału wychodzącego z urządzenia o charakterystyce h=10*sin(t(1:20)).*exp(-2.*t(1:20)) przez które jest przepuszczany sygnał sinusoidalny o częstotliwości 1Hz.

```
close all;
clear all;
% x  sygnał wejściowy
T=10;
dt=0.1;
t=0:dt:T;
%generujemy sygnał wchodzący
x=sin(2*pi*t);
% Generujemy wektor z odpowiedzią sygnału
h=10*sin(t(1:20)).*exp(-2.*t(1:20));
for i=1:2
    h=[h h]; % przedłużamy odpowiedź urządzenia w czasie
end
% obliczamy splot
y=conv(h,x)
%Przeliczamy numer próbki na  czas odpowiedzi
th=1:length(h);
th=th*dt;
%Przeliczamy numer próbki na  czas sygnału wychodzącego
tc=1:length(y);
tc=tc*dt;
```

```
22 subplot(3,1,1); plot(t,x);
23 ylabel('Amplitude [arb. units]','LineWidth', 1.5);
24 text(9.6,0.6 ,['(a)']);
25 subplot(3,1,2); plot(th,h);
26 ylabel('Amplitude[arb. units]','LineWidth', 1.5);
27 text(7.7,1.5 ,['(b)']);
28 subplot(3,1,3); plot(tc,y);
29 ylabel('Amplitude [arb. units]','LineWidth', 1.5);
30 text(17.2,8 ,['(c)']);
31 xlabel('Time [s]');
32 axis([0 inf -inf inf])
```

W powyższym przykładzie przepuszczono trwający 10s sygnał sinusoidalny próbkowany z częstotliwością 10Hz (patrz rysunek 3.8(a)) przez układ LTI, którego charakterystyka jest przedstawiona na rysunku 3.8(b). Obliczony za pomocą konwolucji impuls wyjściowy $y(t)$ został umieszczony na rysunku 3.8. Porównując sygnały $x(t)$ oraz $y(t)$ możemy stwierdzić, że układ z naszego przykładu zmienia w znaczny sposób kształt sygnału wejściowego, opóźniając wyjście o osiem sekund.



Rysunek 3.8. (a) Wykres sygnału wejściowego (b) Odpowiedź układu (c) sygnał wychodzący

3.3. Interpolacja

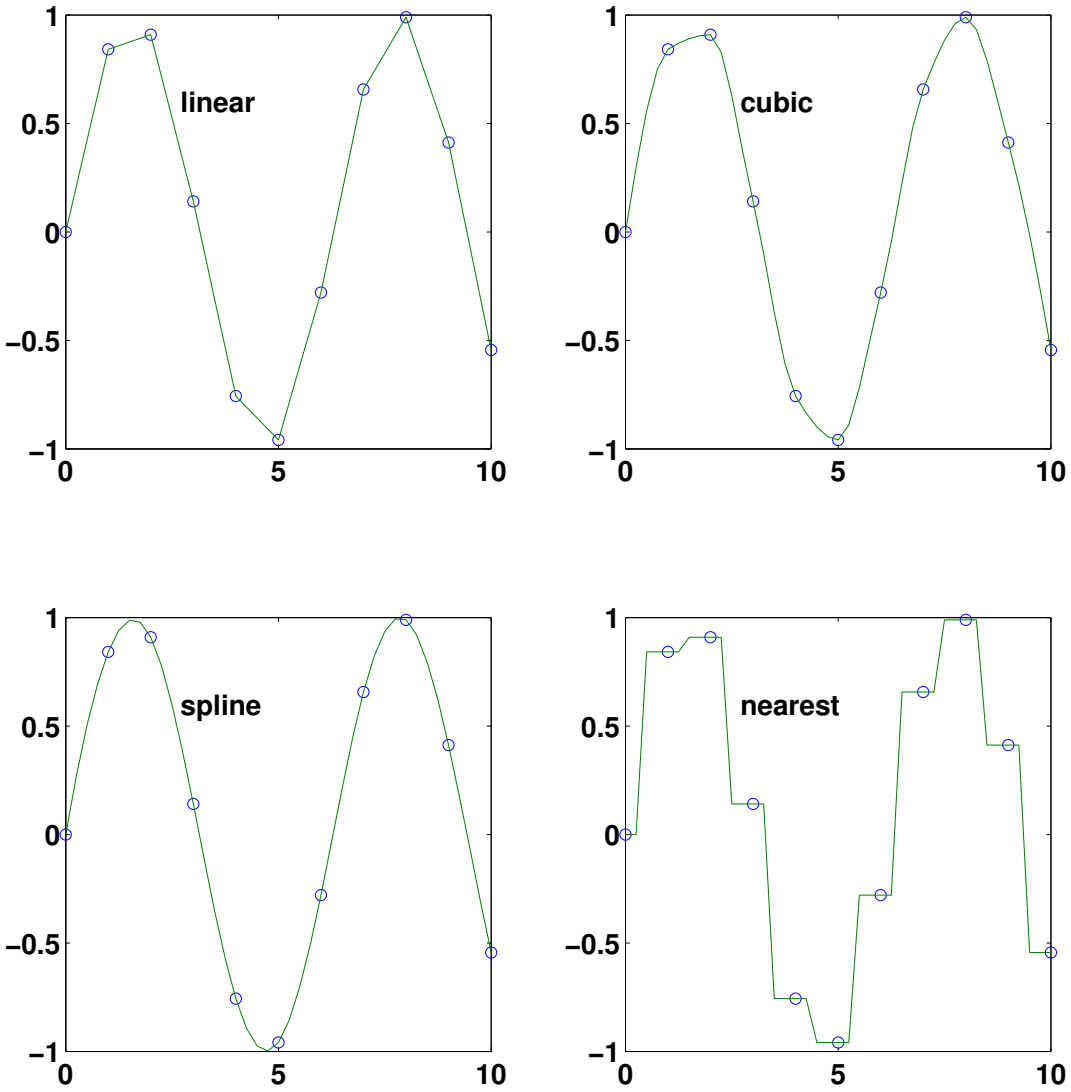
W przypadku gdy dysponujemy sygnałem, który jest próbkowany stosunkowo rzadko, możemy spróbować wygenerować dodatkowe punkty w sygnale używając metody interpolacji. Wszystkie algorytmy interpolacyjne przeprowadzają krzywą interpolująca przez występujące w sygnale wyjściowym punkty. W tym miejscy nie będziemy zajmować omawianiem struktury algorytmów, ponieważ jest to poza zakresem niniejszego wykładu[4] zademonstrujemy jedynie na pojedynczym przykładzie działanie różnych algorytmów interpolujących.

Przykład 6.

Wygenerujmy sygnał cyfrowy $y = sin(x)$ próbkowany z częstotliwością 1Hz, następnie dokonajmy interpolacji sygnału co $0.25s$.

```
clear all;
close all;
% generujemy sygnał co jedną jednostkę
x= 0:10;
y = sin(x);
% wybieram siatkę do interpolacji
xi = 0:.25:10;
ya = sin(xi);
Δ=[];
method={'linear' 'cubic' 'spline' 'nearest' };
label=['a' 'b' 'c' 'd'];
for i=1:length(method)
  % przeprowadzamy interpolację wybraną metodą
   yi = interp1(x,y,xi,method{i});
   subplot(2,2,i); plot(x,y,'o',xi,yi);
   text(2.7, 0.6, [method(i)]);
   Δ(i)=sum((ya-yi).^2)
 end
```

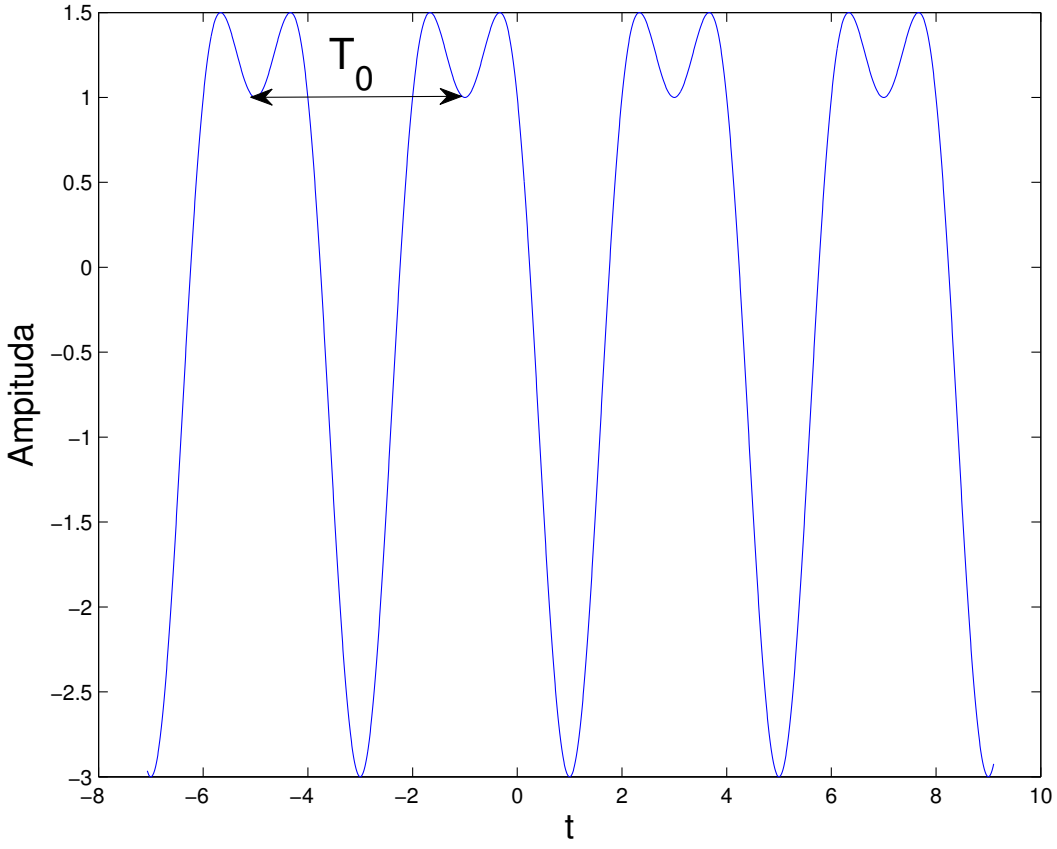
[4] Czytelnika proszę o przejrzenie notatek z metod numerycznych.



Rysunek 3.9. Interpolacja funkcji sinus przeprowadzona metodami: liniową, kubiczną, splajnami, najbliższego punktu.

4. Transformata Fouriera oraz widmo mocy

Załóżmy, że mamy dany sygnał $x(t)$, który jest sygnałem zależnym od czasu. Jeśli chcemy wyróżnić w sygnale elementy powtarzające się z jakąś częstotliwością to wtedy musimy odwołać się do formalizmu analizy Fourierowskiej. W przypadku



Rysunek 4.1. Sygnał okresowy w czasie

gdy mamy pewność, że analizowany przez nas sygnał jest jakąś okresową funkcją czasu (patrz rysunek 4.1) to do analizy poszczególnych składowych częstotliwościowych wystarczy nam rozwiniecie sygnału w zwykły szereg Fouriera

$$x(t) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cos(\omega_n t) + b_n \cos(\omega_n t) \right], \tag{4.1}$$

gdzie częstość $\omega_0 = 2\pi f_0 = 2\pi T_0$ oraz

$$\begin{aligned} a_0 &= \frac{1}{T_0} \int_0^{T_0} x(t) dt, \\ a_n &= \frac{2}{T_0} \int_0^{T_0} x(t) \cos(n\omega_0 t) dt, \\ b_n &= \frac{2}{T_0} \int_0^{T_0} x(t) \sin(n\omega_0 t) dt. \end{aligned} \tag{4.2}$$

Współczynniki a_n oraz b_n dają informację o wkładzie częstości ω_0 do sygnału $x(t)$. W przypadku sygnałów nieokresowych powyższe wzory nie obowiązują. Przechodząc z $T_0 \to \infty$ można rozszerzyć zakres stosowalności formalizmu Fourierowskiego na sygnały nieperiodyczne. Przejście z granicą $T_0 \to \infty$ we wzorach 4.2 prowadzi do transformat Fouriera, które w przypadku funkcji ciągłych, całkowalnych z kwadratem modułu są zdefiniowane następującymi wzorami

$$X(\omega) = \mathcal{F}\left\{x(t)\right\} = \int_{-\infty}^{\infty} x(t) \exp\left(-i\omega t\right) dt \tag{4.3}$$

$$x(t) = \mathcal{F}\left\{X(f)\right\} = \int_{-\infty}^{\infty} X(\omega) \exp\left(+i\omega t\right) d\omega. \tag{4.4}$$

Transformatę Fouriera (4.7) możemy rozumieć jako rzutowanie sygnału $x(t)$ na funkcje $\exp\left(-i\omega t\right) = \cos\left(\omega t\right) - i \sin\left(\omega t\right)$, które stanowią bazę zupełną $\{\exp\left(i\omega t\right), \quad x \in \Re\}$. Transformata Fouriera $X(\omega)$ pokazuje rozkład częstotliwościowy (często nazywany spektrum) funkcji $x(t)$.

W przypadku sygnałów dyskretnych dla których kolejne zapisy są dokonywane z równym odstępem czasowym T_s (częstotliwością próbkowania $f_s = 1/T_s$) całka w transformacie Fouriera (4.7) przechodzi w sumę po kolejnych próbkowanych czasach nT_s

$$X(\omega) = \sum_{n=0}^{N-1} x(n) \exp(-i\omega n T_s), \tag{4.5}$$

przy czym częstość może przyjmować ciągłe wartości $0 \leqslant \omega \leqslant 2\pi T_s$, a N oznacza liczbę próbek sygnału $x(t)$. Wprowadzając oznaczenie na częstość próbkowania $\omega_s = 2\pi/T_s$ przedział częstości możemy zapisać jako $0 \leqslant \omega \leqslant \omega_s$. Sumowanie we wzorze (4.5) zaczynamy od zera ponieważ dla sygnałów fizycznych, obecna wartość zależy jedynie od stanów przyjętych przez układ w przeszłości lub teraźniejszości. Kiedy przetwarzamy częstotliwości sygnałów cyfrowych na komputerze musimy również zdyskretyzować również częstości ω. Zastępując ciągłe wartości $\omega \in \langle 0, \omega_e \rangle$ wartościami dyskretnymi

$$\omega_k = \frac{2\pi}{NT_s}k, \text{ gdzie } k \in \{0, 1, 2, \ldots, N-1\}. \tag{4.6}$$

Ostatecznie ze wzoru (4.5) otrzymujemy wzór

$$X(k) = X(\omega_k) = \sum_{n=0}^{N-1} x(n) \exp\left(-i\frac{2\pi}{N}kn\right), \quad k = 1, 2, \ldots, N-1 \tag{4.7}$$

na Dyskretną Transformatę Fouriera (DFT). Powyższy wzór w sposób jednoznaczny definiuje transformatę Fouriera dyskretnego sygnału składającego się z N próbek[1].

Odwrotna transformata określona jest wzorem

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k) \exp\left(i\frac{2\pi}{N}kn\right), \quad n = 1, 2, \ldots, N-1. \tag{4.8}$$

Zastanówmy się od czego zależy zdolność rozdzielcza częstotliwości. Biorąc różnice dwóch sąsiednich częstości

$$\Delta\omega = \omega_{k+1} - \omega_k = \frac{2\pi}{NT_s} \tag{4.9}$$

możemy zauważyć, że zwiększając długość próbki NT_s jesteśmy w stanie powiększyć naszą zdolność rozdzielczą wychwytując coraz mniejsze różnice częstości $\Delta\omega$.

W zastosowaniach bardzo często zamiast przedziału częstości $\omega_k \in \left\langle 0, \frac{2\pi}{T_s} \right\rangle$ wprowadza się znormalizowany przedział częstości, w którym redefiniuje się ω_k następująco

$$\omega_k = \frac{2\pi}{N}k,$$

[1] Zainteresowanego dowodem czytelnika odsyłam do [7].

tak aby

$$\omega_k \in \langle 0, 2\pi \rangle .$$

Co przekład się w następujący sposób

$$f_k = \frac{\omega_k}{2\pi} = \frac{k}{N}$$

na znormalizowaną częstotliwość, która należy do przedziału $f_k \in \langle 0, 1 \rangle$.

Wprowadźmy pojęcie mocy chwilowej sygnału $p(t)$, którą definiujemy jako moduł kwadratu amplitudy

$$p(t) = |x(t)|^2. \tag{4.10}$$

Energię chwilową definiujemy zaś jako

$$e(t) = p(t)\, dt = |x(t)|^2\, dt, \tag{4.11}$$

zaś energia całkowita jest definiowana

— całką dla sygnału ciągłego

$$E = \lim_{T \leftarrow \infty} \int_{-T}^{T} |x(t)|^2\, dt, \tag{4.12}$$

— sumą sygnału dyskretnego

$$E = \sum_{n=0}^{N-1} |x(n)|^2 . \tag{4.13}$$

Moc średnia sygnału jest średnią czasową całkowitej energii

$$P = lim_{T \leftarrow \infty} \frac{E}{T}. \tag{4.14}$$

Jednym z ważniejszych twierdzeń w analizie fourierowskiej jest **twierdzenie Parsevala**. Mówi ono, że całkowita energia sygnału przed i po transformacie Fouriera jest taka sama, co można zapisać wzorami odpowiednio:
dla sygnału ciągłego

$$\int_{-\infty}^{\infty} |x(t)|^2\, dt = \int_{-\infty}^{\infty} |X(\omega)|^2\, d\omega, \tag{4.15}$$

oraz

$$\sum_{n=0}^{N-1} |x(n)|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X(k)|^2 \tag{4.16}$$

dla sygnału dyskretnego. Licząc dla danego sygnału transformatę Fouriera dokonujemy rozkładu spektralnego na odpowiednie częstotliwości. Z kolei na podstawie twierdzenie Parsevala wiemy, w które częstotliwości lokuje się największa część energii sygnału.

Wykres $|X(k)|^2$ w funkcji częstotliwości f_k nazywa się **widmem mocy**. W literaturze angielskiej na widmo mocy stosuje się dwa terminy *power spectrum* lub *power spectral density* (w skrócie **PSD**) [5].

W pakietach Matlab oraz Octave[2] dzięki zaimplementowaniu algorytmu szybkiej transformaty Fouriera można w bardzo wygodny sposób prowadzić analizę spektralną sygnałów cyfrowych. W następnym podrozdziale przedstawimy przykłady zastosowań analizy fourierowskiej.

4.1. Filtrowanie i analiza widmowa sygnałów

W niniejszym rozdziale będziemy ilustrować przykładamy ważniejsze elementy analizy fourierowskiej sygnałów cyfrowych.

Przykład 7.

Wyznaczmy moc periodycznego sygnału określonego zależnością funkcyjną $y(t) = 1.5\sin(2\pi 2t) + \sin(2\pi 4t)$ o długości $T = 100s$ próbkowanego z częstotliwością $100Hz$.

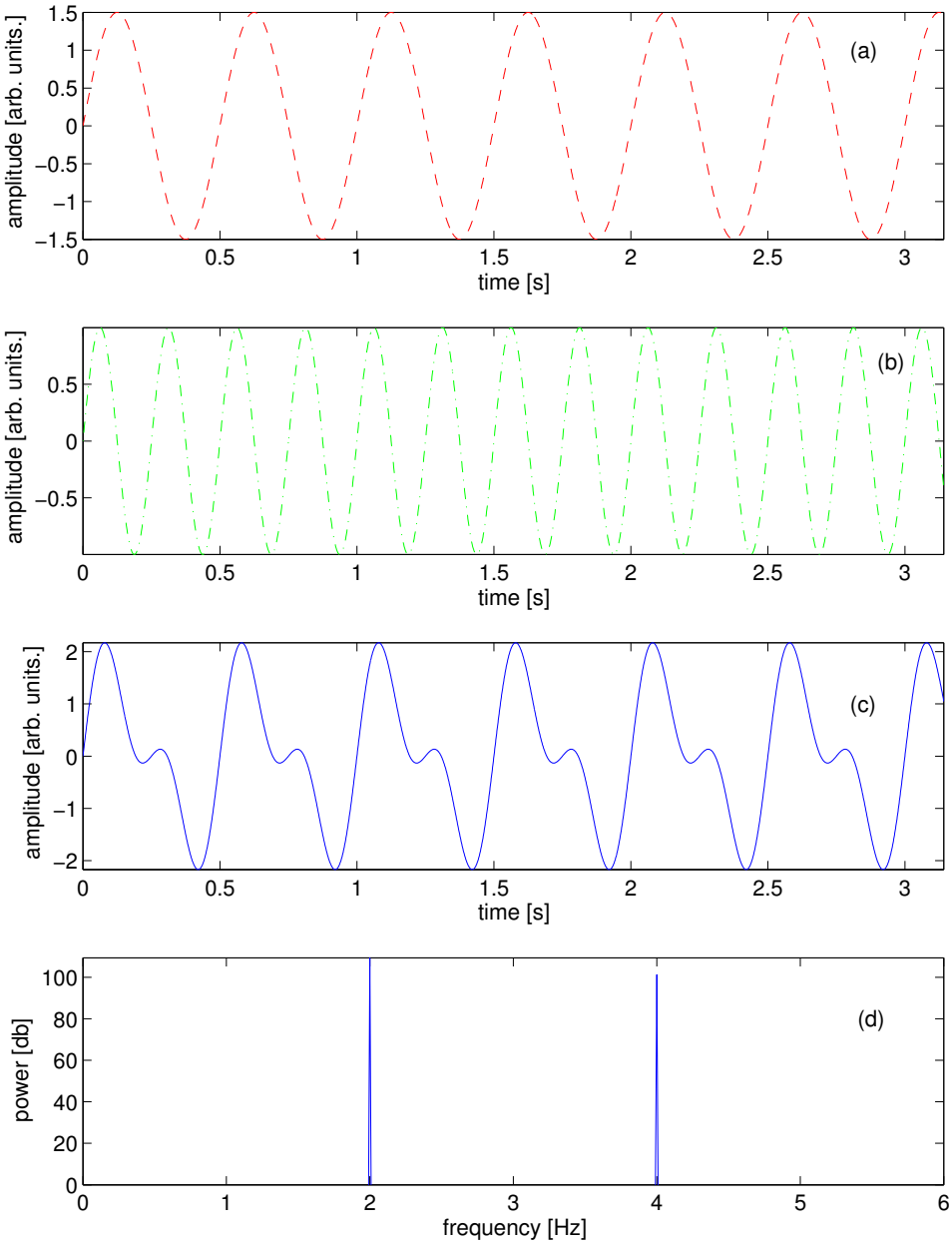
```
1 close all
2 clear all
3 T=100; % Długość sygnału, tak dobrana aby sygnał
```

[2] Podobny do Matlaba pakiet obliczeniowy dystrybuowany na licencji GNU

```
4 % przyjmował wartości równe zeru na końcach przedziału
5 dt=0.001;
6 t=0:dt:T;
7 y=1.5*sin(2*pi*2*t)+sin(2*pi*4*t); %sygnał mający
8 % składowe o częstotliwościach odpowiednio 2Hz i 4 Hz
9 figure(1);
10 % wykres sygnału o częstotliwości 2Hz
11 subplot(4,1,1);plot(t,1.5*sin(2*pi*2*t),'--r');
12  text(2.8,1,['(a)']);
13  axis([0 pi -inf inf]);
14  xlabel('time [s]');
15  ylabel('amplitude [arb. units.]');
16 hold on
17 % wykres sygnału o częstotliwości 4Hz
18 subplot(4,1,2);plot(t,sin(2*pi*4*t),'-.g');
19 text(2.9,0.7,['(b)']);
20 axis([0 pi -inf inf]);
21 xlabel('time [s]');
22  ylabel('amplitude [arb. units.]');
23 % wykres wypadkowego sygnału
24 subplot(4,1,3);plot(t,y);
25 text(2.8,1,['(c)']);
26 axis([0 pi -inf inf]);
27 xlabel('time [s]');
28 ylabel('amplitude [arb. units.]');
29
30 fy=fft(y); % Obliczamy DTF
31 freq=linspace(0,1,length(t))*1/dt; % Przeliczamy częstotliwość
32 % z przedziłu znormalizowaneg <0,1> na przedział rzeczywisty
33
34 % wykres mocy sygnału w decybelach dla poziom odniesienia
35 % Po=1
36 subplot(4,1,4);plot(freq,10*log(abs(fy).^2/length(y)));
37 axis([0 6 0 inf]);
```

```
38 text(5.4,80,['(d)']);
39 xlabel('frequency [Hz]');
40 ylabel('power [db]');
```

Rysunek 4.2 pokazuje fragment sygnału cyfrowego y_n dla dyskretnych czasów należących do przedziału $t_n \in \langle 0, 100s \rangle$. Przy czym początek i koniec przedziału zostały tak wybrane, aby w tych miejscach $y(t) = 0$. Sygnał analogowy $y(t)$ składa się z dwóch harmonicznych o częstościach odpowiednio 2Hz i 4Hz. Sygnał $y(t)$ jest w długim przedziale czasu bardzo gęsto próbkowany z częstotliwością $f_s = 1000Hz$. Tak przyjęte warunki analizy pozwalają na uzyskanie na rysunku 4.2(d) przedstawiającym widmo mocy dwóch bardzo ostrych pików, które są dobrze zlokalizowane na prawdziwych częstotliwościach 2Hz oraz 4Hz występujących w sygnale analogowym $y(t)$. Z rysunku 4.2(c) możemy zauważyć, że w czasie 100s sygnał $y(t)$ wykona bardzo wiele oscylacji. Oscylacje, które wykonuje $y(t)$ zachodzą z dużo mniejszą częstotliwością niż częstotliwość próbkowania.



Rysunek 4.2. Analiza fourierowsa sygnały cyfrowego próbkowanego z częstotliwością $f_s = 1000Hz$. (a) składowa sygnału $y_1(t) = 1.5\sin(2\pi\ 2t)$, (b) składowa sygnału $y_2(t) = \sin(2\pi\ 4t)$, (c) sygnał złożony $y(t) = y_1(t) + y_2(t)$ (d) widmo mocy sygnału.

Dla cyfrowego sygnału określenia „długi przedział czasu” („długi sygnał”) oraz duża częstotliwość próbkowania powinny być zawsze odniesione do składowych częstotliwościowych jakie spodziewamy się znaleźć w analizowanym sygnale.

W poprzednim rozdziale pokazaliśmy, że zdolność do rozdzielenia sąsiednich częstotliwości jest związana z długością sygnału i może być poprawiona poprzez wydłużenie długości sygnału. Zademonstrujemy ten efekt skracając długość sygnału pochodzącego z przykładu 7 przy pozostawieniu pozostałych parametrów bez zmian.

Przykład 8.

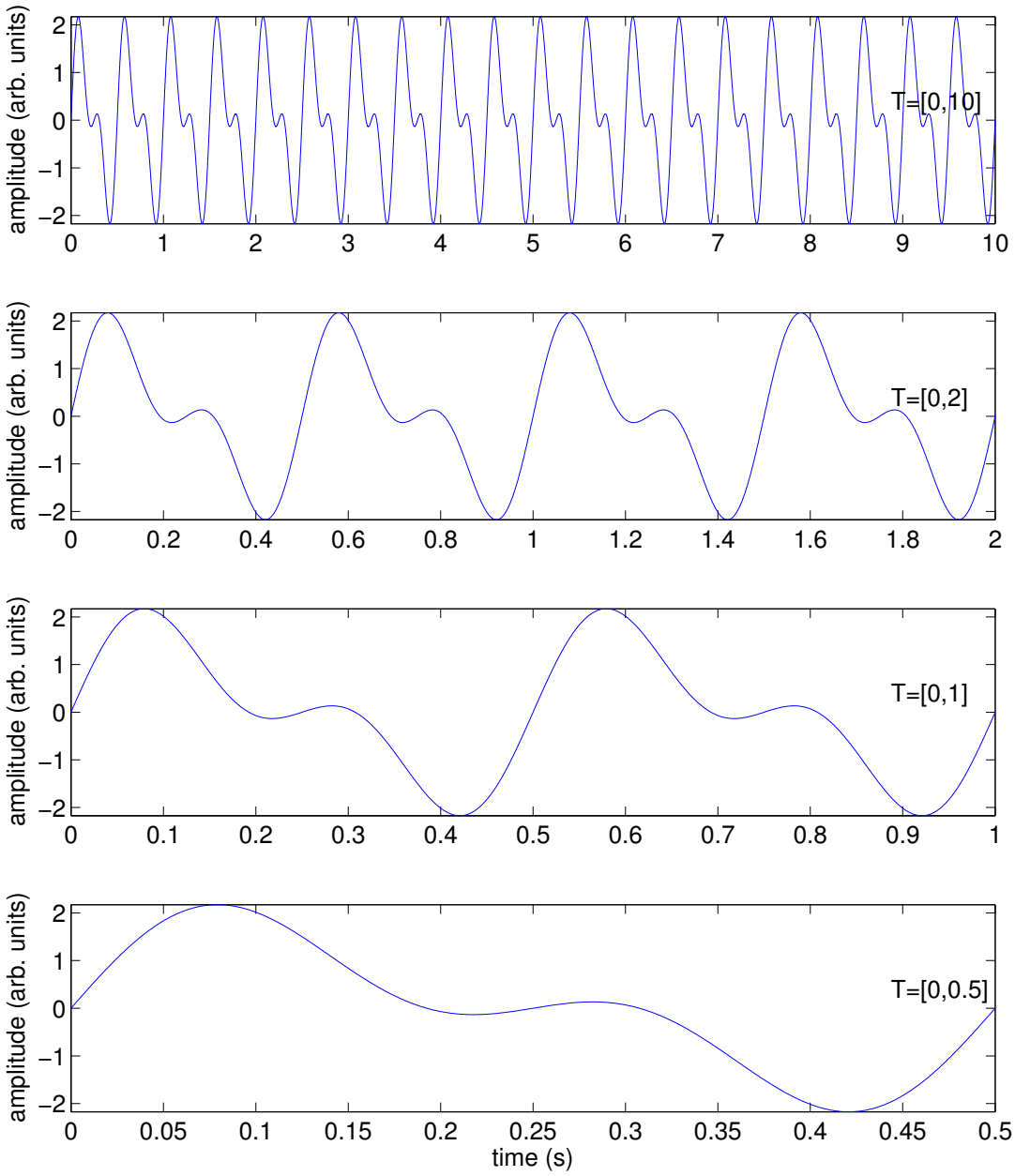
Przeprowadź analizę Fourierowską dla sygnału z poprzedniego przykładu zmniejszając jego długość trwania do $T = 10s$, $T = 2s$, $T = 1s$, $T = 0.5s$. Poniżej zamieszczony jest listing z komentarzami.

```
clear all;
close all;
fs=1000; % Częstotliwość próbkowania
dt=1/fs; % odstęp między kolejnymi próbkami
vecdt=[10 2  1 0.5 ]; % wektor zawierający
% długość sygnału
for i=1:length(vecdt)
  T=vecdt(i);
  t=0:dt:T;
  y=1.5*sin(2*pi*2*t)+sin(2*pi*4*t);
  figure(1);
  hold on
  subplot(4,1,i);plot(t,y);
  axis([0 T -inf inf]); % skalowanie osi
  ylabel('amplitude (arb. units)')
  text(T-0.12*T,0.4  ,[' T=[0,',num2str(T),']']);
  %kładziemy etykietki
  fy=fft(y); % transformata Fouriera
```

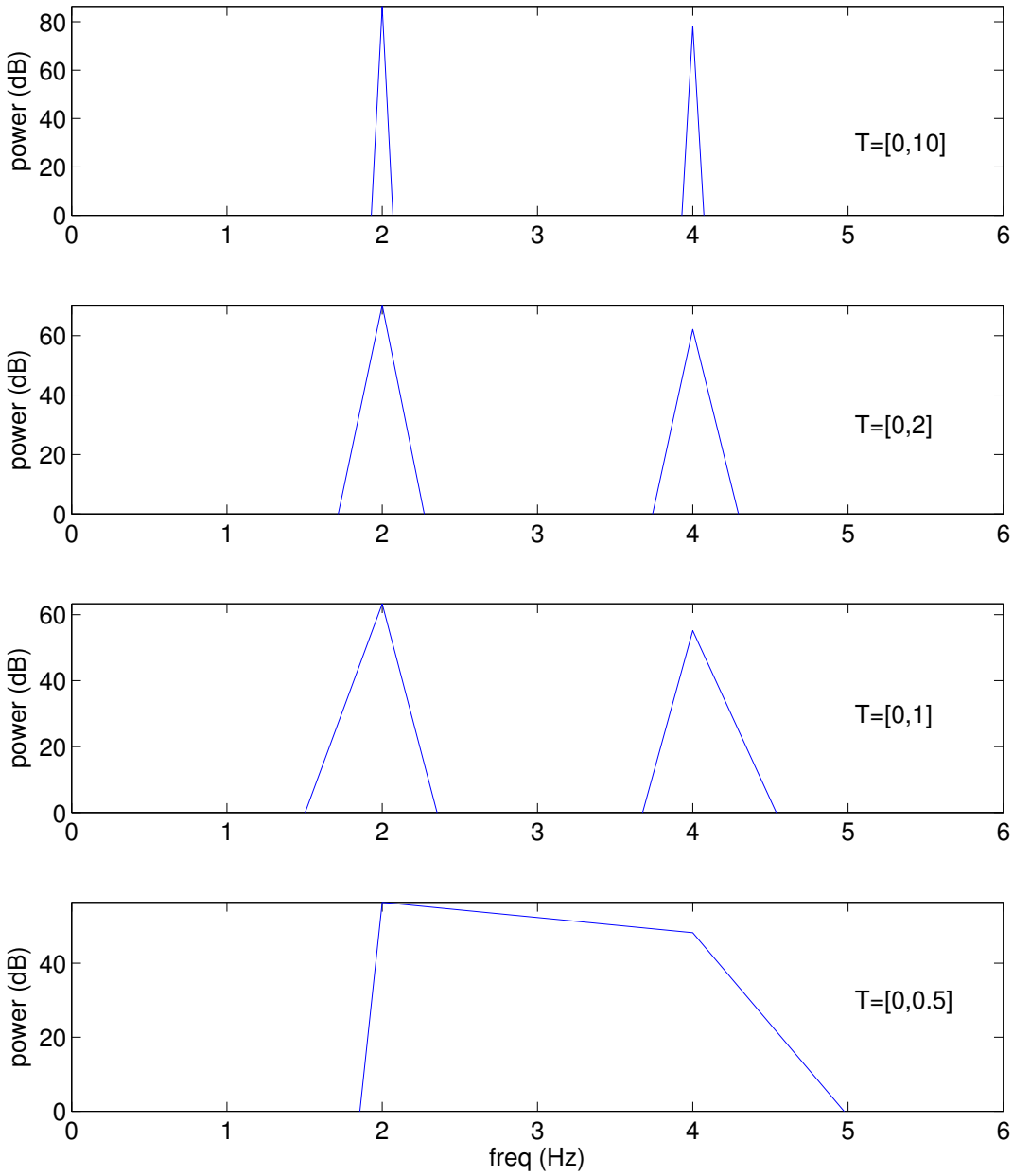
```
19      freq=linspace(0,1,length(t))*1/dt;
20      figure(2);
21      subplot(4,1,i);plot(freq,10*log(abs(fy).^2/length(y)));
22      axis([0 6 0 inf]);
23      text(5,30,[' T=[0,',num2str(T),']']);
24      ylabel('power (dB)')
25  end
26  figure(2);
27  xlabel('freq (Hz)');
28  figure(1);
29  xlabel('time (s)');
```

Na rycinie 4.3 pokazano kształt sygnału dla czasów T=10s, T=2s, T=1s oraz T=0.5s, natomiast na rycinie 4.4 zamieszczono odpowiednie widma mocy sygnału. Porównując odpowiadające sobie wykresy z rycin 4.3 oraz 4.4 możemy zauważyć stopniowe pogorszenie się rozdzielczości częstotliwościowej postępujące wraz ze zmniejszeniem długości sygnału. Dla długości (T=0.5s) równej jednemu okresowi sygnału rozdzielczość jest tak słaba, że rozróżnienie częstotliwości składowych 2Hz oraz 4Hz staje się nie możliwe.

Mając w pamięci twierdzenie Nyquista, które mówi że sygnał cyfrowy poprawnie odtwarzane te częstości sygnału analogowego które są mniejsze od połowy częstotliwości próbkowania, można stwierdzić że próbkowanie sygnału, w którym mamy jedynie komponenty o częstotliwościach 2Hz i $4Hz$ z częstotliwością 1000Hz jest sporym zbytkiem. Wybieranie dużo większej częstotliwości niż jest potrzebna nazywa się nadpróbkowaniem (ang. *oversampling*). Tą technikę stosuje się czasami gdy zależy nam na otrzymaniu „gęstego sygnału", aby bezpiecznie móc obrabiać go numerycznie (np. licząc różniczki).



Rysunek 4.3. Sygnał $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$ o długosci (zaczynając od góry rysunku) T=10s, T=2s, T=1s, T=0.5s.



Rysunek 4.4. Widmo mocy dla sygnału $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$ o długosci (zaczynając od góry ryciny) T=10s, T=2s, T=1s, T=0.5s

Przykład 9.

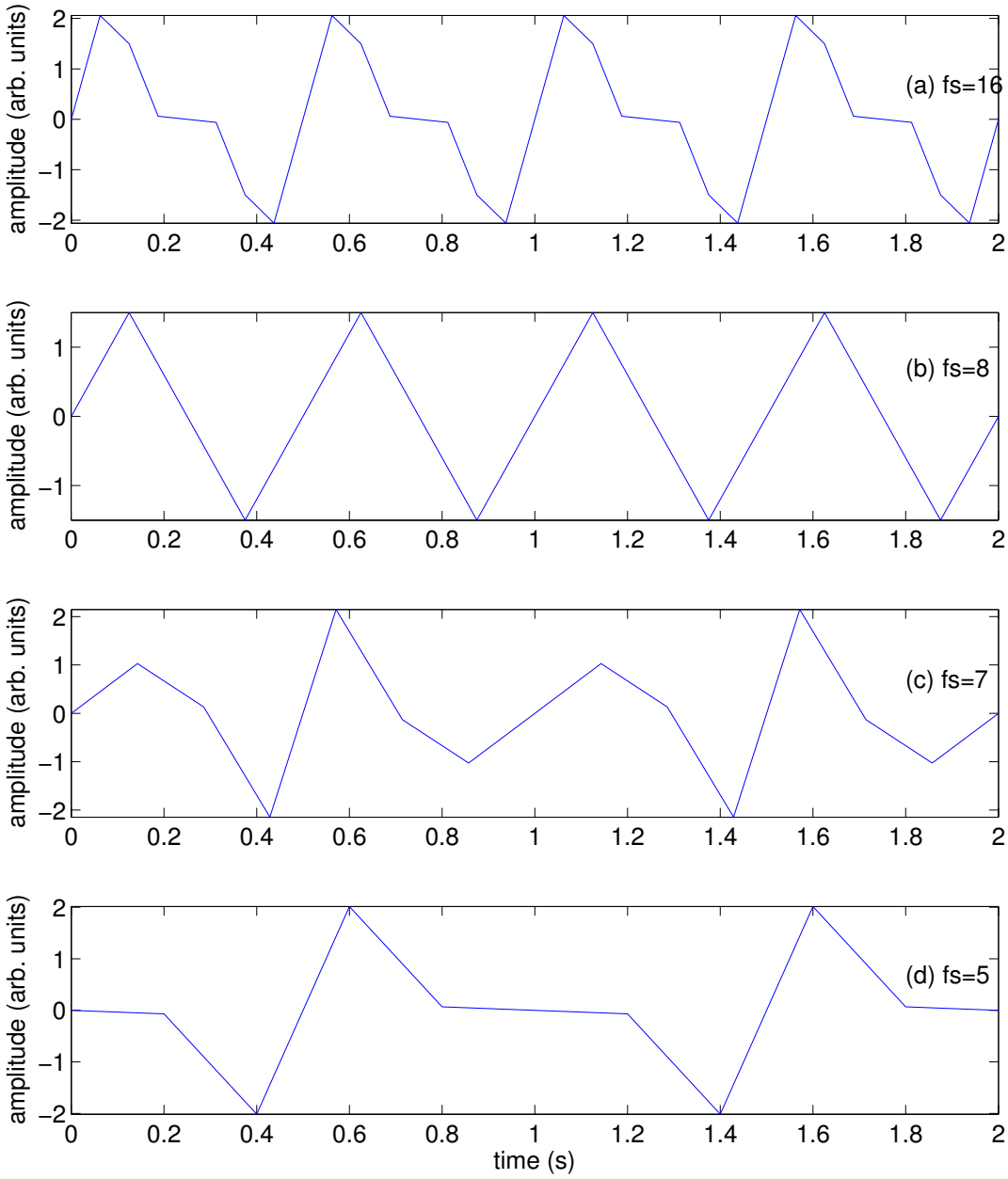
Weźmy znany już nam sygnału $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$. Będziemy go próbkować przez 100s z różnymi częstościami: 9Hz, 8Hz, 7Hz, 6Hz, co się stanie gdy sygnał będzie próbkowany z częstotliwością nie spełniającą warunków twierdzenia Nyquista.

```
clear all;
close all;
T=100 % długość sygnału
vfreq=[16  8  7  5 ]; % wektor zawierający częstotliwości ...
    próbkowania
labels=['a' 'b' 'c' 'd'];
for i=1:length(vfreq)
  fs=vfreq(i);
  dt=1/fs; % odstęp między kolejnymi próbkami
  t=0:dt:T;
  y=1.5*sin(2*pi*2*t)+sin(2*pi*4*t);
  figure(1);
  hold on
  subplot(4,1,i);plot(t,y);
  axis([0 2 -inf inf]); % skalowanie osi
  ylabel('amplitude (arb. units)')
  text(1.8,0.7 ,['(',labels(i),') fs=',num2str(fs)]);
  %kładziemy etykietki
  fy=[];
  freq=[]
  fy=fft(y); % transformata Fouriera
  freq=linspace(0,1,length(t))*1/dt;
  figure(2);
  subplot(4,1,i);plot(freq,10*log(abs(fy).^2/length(y)));
  axis([0 fs 0 inf]);
  text(fs/2-fs*0.05,30,['(',labels(i),') fs=',num2str(fs)]);
```

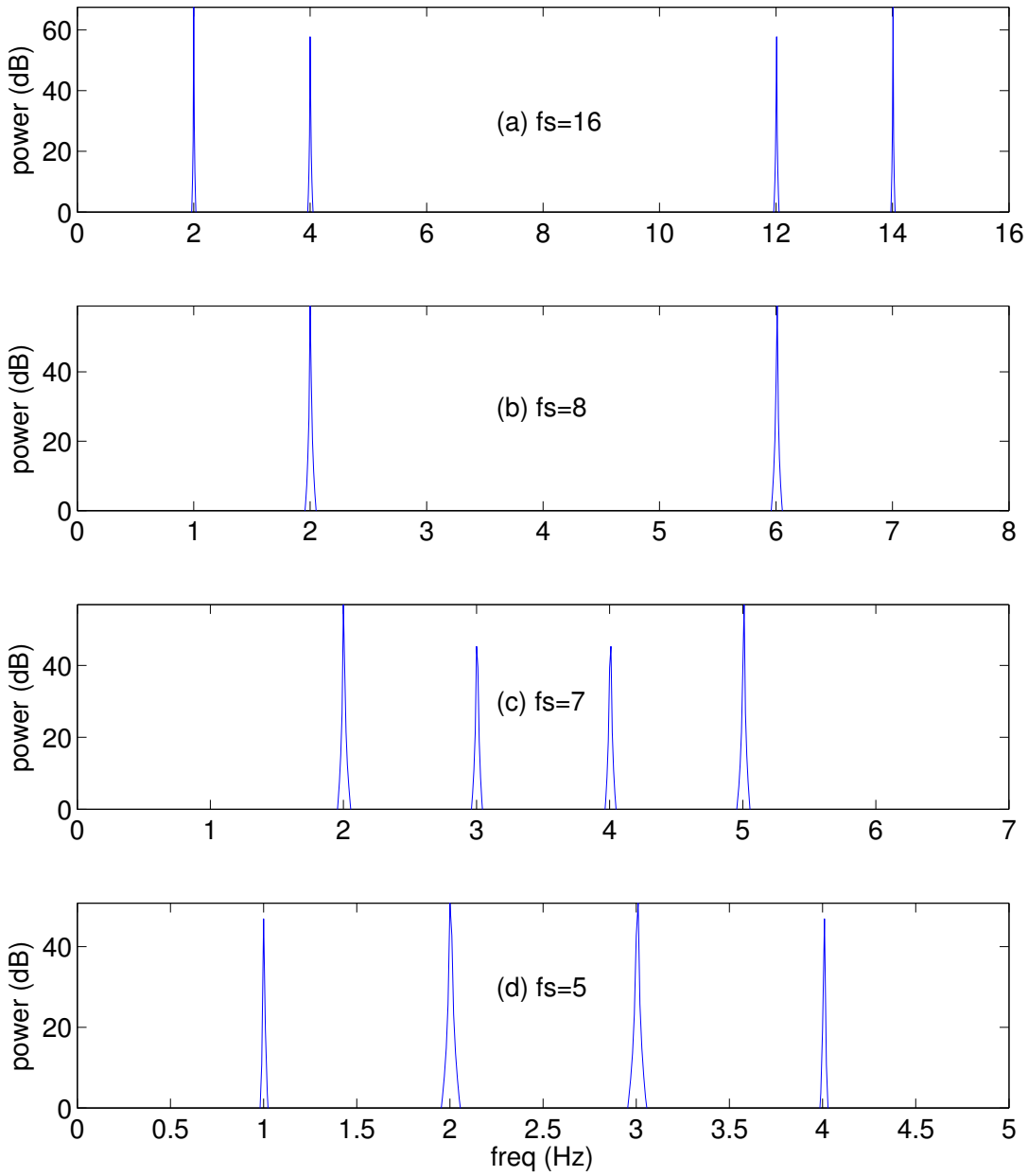
```
26     ylabel('power (dB)')
27 end
28 figure(2);
29 xlabel('freq (Hz)');
30 figure(1);
31 xlabel('time (s)');
```

Graficznie wyniki działania powyższego skrypty przedstawiono na rysunkach 4.5 oraz 4.6. Jak już wspomnieliśmy sygnał $y(t)$ składa się z dwóch harmonicznych o częstościach 2Hz oraz 4Hz, więc zgodnie z twierdzeniem Nyquista, aby poprawnie odtworzyć charakterystykę częstotliwościową sygnał powinien być próbkowany z częstotliwością większą niż 8Hz. Panele (a) rysunków 4.5, 4.6 prezentują odpowiednio sygnał próbkowany z częstotliwością spełniającą warunki twierdzenia Nyquista. Częstotliwość Nyquista dla panelu 4.6(a) jest równa 8Hz i względem niej zgodnie z własnościami symetrii transformaty Fouriera rozłożone są poprawnie odtworzone częstotliwości 2Hz, 4Hz oraz ich odpowiedniki 12Hz oraz 14Hz. Dla paneli (c) i (d) rysunku 4.6 częstotliwość Nyquista $f_s/2$ równa jest odpowiednio 3.5Hz oraz 2.5Hz. Obie wartości są mniejsze od 4Hz, czyli częstotliwości wyższej harmonicznej występującej w sygnale $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$. Konsekwencją nie spełnienia warunków twierdzenia Nyquista jest pojawienie się na panelach (c) oraz (d) fałszywych częstotliwości o wartościach odpowiednio 3Hz oraz 1 Hz. Fałszywe częstości powstały wskutek przyjęcia zbyt małej częstotliwości próbkowania co spowodowało niedopróbkowanie składowej harmonicznej o częstotliwości 4Hz.

Zjawisko „zawinięcia się" niedopróbkowanych częstotliwości wokół częstotliwości Nyquista nazywa się **alliasingiem**. Prowadzi ono do trwałego zanieczyszczenia sygnału cyfrowego przez niefizyczne niedopróbkowane częstotliwości, które o ile nie znamy skądinąd ich pozycji są nie usuwalne. *Alliasingowi* można zapobiegać na etapie digitalizacji sygnału przepuszczając sygnał analogowy przez filtry.



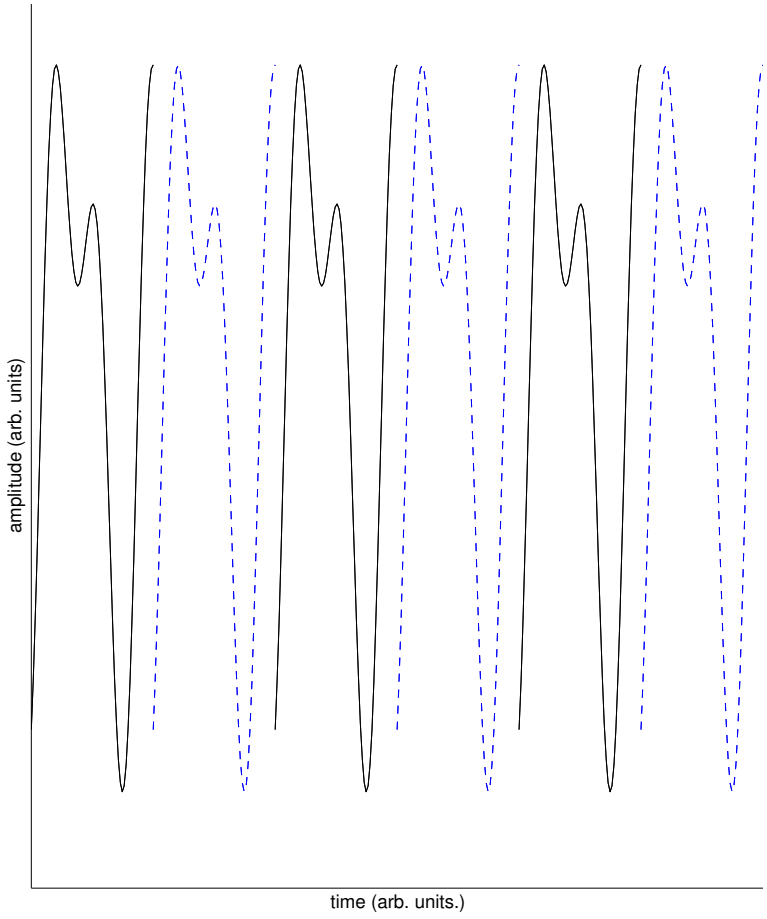
Rysunek 4.5. Sygnał $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$ o długosci T=100s próbkowany z częstotliwościami (zaczynając od góry rysunku), (a) $f_s = 16$Hz, (b) $f_s = 8$Hz, (c) $f_s = 7$Hz, (d) $f_s = 5$Hz.



Rysunek 4.6. Widmo mocy sygnału $y(t) = 1.5\sin(2\pi\, 2t) + \sin(2\pi\, 4t)$ próbkowanego z częstotliwością $f_s = 16$Hz, $f_s = 8$Hz, $f_s = 7$Hz $f_s = 5$Hz.

Filtry analogowe używane do oczyszczania sygnału analogowego z niechcianych wysokich częstotliwości przepuszczają sygnał jedynie do określonej maksymalnej częstotliwości. Z tego powodu tego typu filtry nazywa się filtrami dolnoprzepustowymi.

Algorytmy numeryczne służące do obliczania transformat Fouriera[3] zakładają, że że przygotowany przez nas sygnał cyfrowy powtarza się okresowo w czasie. Takie przekształcenie zostało zilustrowane na rysunku 4.7. Jeśli koniec i początek



Rysunek 4.7. Powtarzający się periodycznie zdigitalizowany sygnał. Kolejne powtórzenia sygnału wyróżniono zmieniając linię ciągłą na przerywaną.

[3] między innymi algorytm szybkiej transformaty Fouriera, w wersji zaimplementowanej w pakietach Matlab oraz Octave

próbki cyfrowej daje się zszyć w sposób ciągły [4], to wtedy powielanie sygnału nie będzie dawać zmieniać jego widma. Właśnie takie warunki przyjęliśmy w przykładzie 7. W przypadku gdy początek i koniec próbku nie daje się zszyć w sposób ciągły tak jak ma to miejsce na rysunku 4.7 dochodzi do rozmycia częstotliwości w widmie mocy[5]. W praktyce można zmniejszyć przeciek obwodząc sygnał funkcją, która wymusi przy osiąganiu końców sygnału gładkie przejście wartości sygnału do zera. Procedurę tego typu nazywamy okienkowaniem, a samą funkcję przez którą mnożymy sygnał nazywamy okienkiem. Efekt przecieku zilustrujemy w następnym przykładzie jednocześnie postaramy się zmniejszyć rozmiar przecieku stosując okno Hanninga.

Przykład 10.

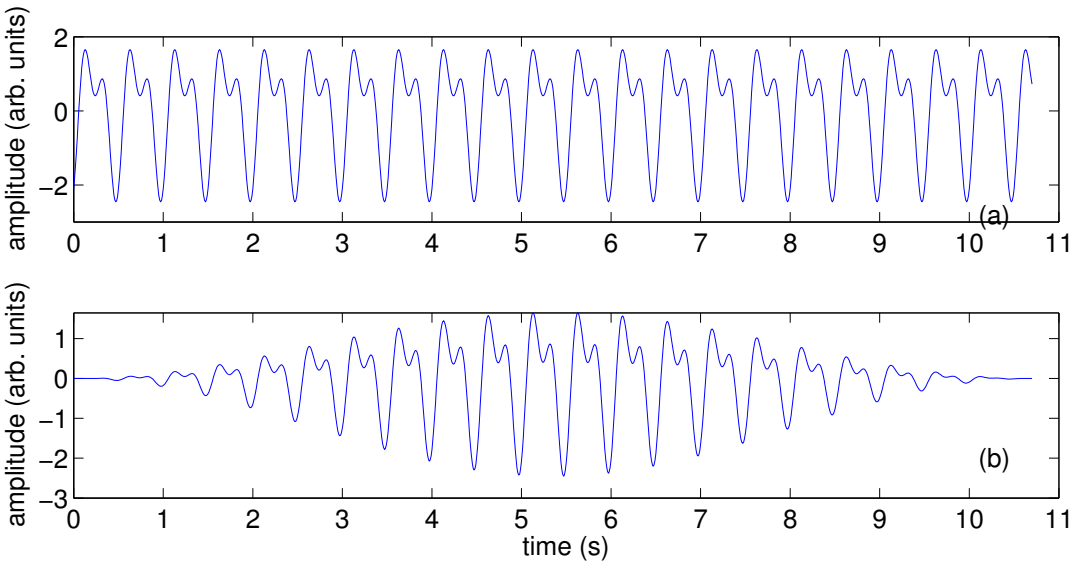
Weźmy sygnał $y = 1.5sin(2\pi\, 2t - 1) + sin(2\pi\, 4t - 1)$, dobierzmy przedział próbkowania tak, aby niemożliwe było zszycie w sposób ciągły początku i końca próbki. Przeprowadzimy analizę Fourierowską sygnału czystego oraz okienkowanego oknem Hanninga.

```
clear all;
close all;
T=10.7 % długość sygnału
fs=100 ; % wektor zawierający częstotliwości próbkowania
dt=1/fs; % odstęp między kolejnymi próbkami
t=0:dt:T;
y=1.5*sin(2*pi*2*(t)-1)+sin(2*pi*4*(t)-1);
w=hanning(length(y),'periodic'); % okno Hanninga o długości
%sygnału y
ywind=y.*w'; %okienkowanie sygnału
figure(1);
hold on
```

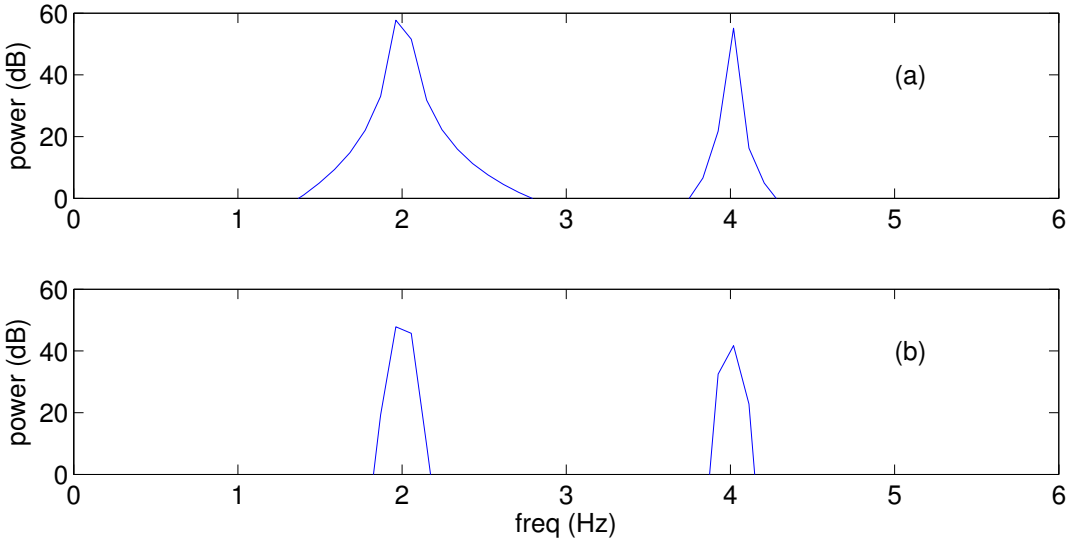
[4] w sensie ciągłości funkcji

[5] w literaturze określa się to mianem przecieku (ang. *frequency leakage*)

```
13 subplot(2,1,1);plot(t,y);
14 text(10.1,-2.8 ,['(a)']);
15 axis([0 11 -3 2]); % skalowanie osi
16 ylabel('amplitude (arb. units)')
17
18 subplot(2,1,2);plot(t,ywind);
19 axis([0 11 -3 inf]); % skalowanie osi
20 ylabel('amplitude (arb. units)')
21 text(10.1,-2 ,['(b)']);
22
23 fy=[];
24 freq=[]
25 fy=fft(y); % transformata Fouriera
26 fyw=fft(ywind); % transformata Fouriera sygnału okienkowanego
27 freq=linspace(0,1,length(t))*1/dt;
28 figure(2);
29 subplot(2,1,1);plot(freq,10*log(abs(fy).^2/length(y)));
30 axis([0 6 0 60]);
31 text(5,40 ,['(a)']);
32 ylabel('power (dB)')
33 subplot(2,1,2);plot(freq,10*log(abs(fyw).^2/length(y)));
34 axis([0 6 0 60]);
35 text(5,40 ,['(b)']);
36 ylabel('power (dB)')
37
38 figure(2);
39 xlabel('freq (Hz)');
40 figure(1);
41 xlabel('time (s)');
```



Rysunek 4.8. (a) Sygnał $y(t) = 1.5\sin(2\pi\, 2t - 1) + \sin(2\pi\, 4t - 1)$ o długosci T=10.7s próbkowany z częstotliwością $f_s = 100$Hz (b) Sygnał $y(t) = 1.5\sin(2\pi\, 2t - 1) + \sin(2\pi\, 4t - 1)$ przemnożony przez okno Hanninga



Rysunek 4.9. Widmo mocy dla sygnału (a) czystego (b) okienkowanego.

Sygnał przedstawiony na rysunku 4.8a jest przesunięty w czasie tak, aby dla czasu $t = 0$ wartość sygnału była ujemna, natomiast koniec próbki celowo wybrano tak, aby wartość sygnału była większa od zera. W takim przypadku nie jest możliwe ciągłe zszycie sygnału i dochodzi do rozmycia częstotliwościowego, które jest uwidacznia się na rysunku 4.9a poprzez poszerzenie pików odpowiadających częstotliwości 2Hz i 4 Hz. W celu określenia skali przecieku, czytelnikowi zaleca się porównanie widma mocy z rysunku 4.9a z widmem mocy (patrz górny panel rysunku 4.4) nie przesuniętego sygnału (patrz górny panel rysunku 4.3), który ma takie same, równe zeru, wartości na krańcach przedziału próbkowania i jest próbkowany w podobnych warunkach.

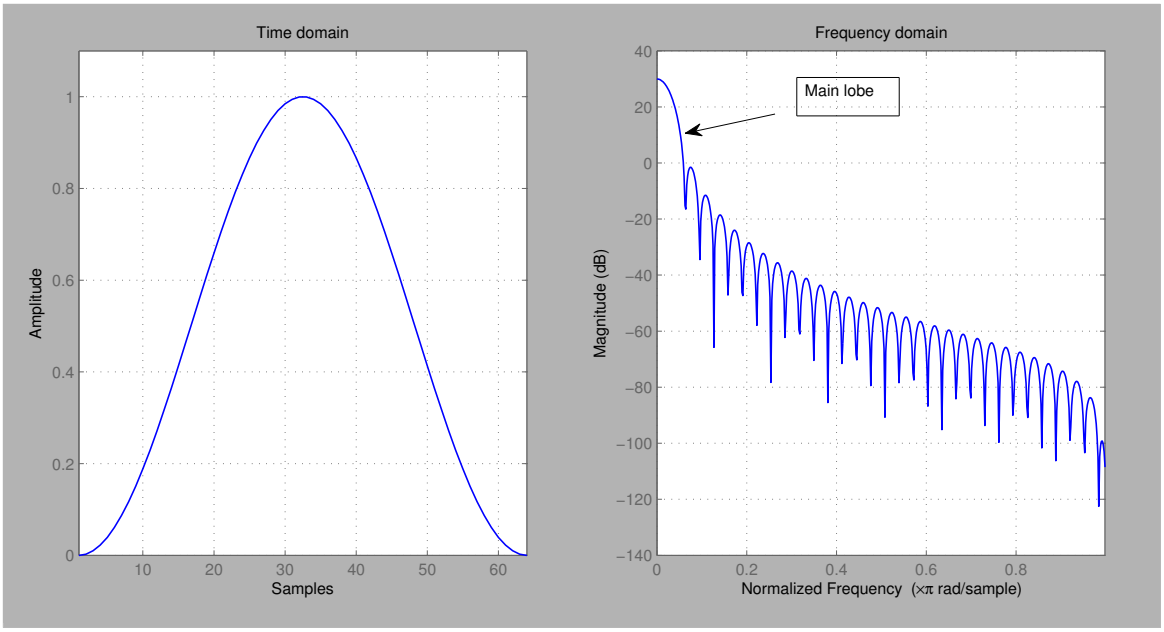
Panel rysunku 4.8 (b) przedstawia sygnał poddany procedurze okienkowania. Pamiętając, że całkowita moc sygnału jest całką z kwadratu modułu, możemy spostrzec porównując panele (a) i (b), że moc sygnału okienkowanego zmalała w stosunku do mocy sygnału wejściowego (mniejsze pole pod kwadratem z krzywej). Porównując widmo mocy sygnału wejściowego panelu (a) rysunku 4.9 z widmem mocy sygnału okienkowanego przedstawionym na panelu (b) tego samego rysunku, możemy za obserwować wyraźne zmniejszenie rozmycia częstotliwościowego dla sygnału poddanego przefiltrowanego przez okno Hanninga. Ograniczenie przecieku, które obserwujemy na rysunku 4.9b odbywa się kosztem spadku całkowitej mocy sygnału okienkowanego. Zmniejszenie mocy sygnału okienkowanego jest bardzo dobrze widoczne przy porównaniu pól pod krzywymi mocy na rysunku4.9a oraz 4.9b.

W praktycznych zastosowaniach stosuje się okna mające różny kształt w dziedzinie czasu np. trójkątne, prostokątne (brak okna), flat-top, Hanninga. Przy projektowaniu nowego rodzaju okna wykorzystuje się z reguły własności spektralne danego okna, ponieważ na podstawie charakterystyki częstotliwościowej okna można określić jego rozdzielczość częstotliwościową. Poszczególne piki występujące w widmie mocy sygnału nazywamy listkami, a pierwszy listek rozkładu nosi nazwę listka głównego (*ang. main lobe*). Przy pomocy wbudowanej funkcji

wvtool można w prosty sposób wykreślić dowolne okno w funkcji czasu oraz jego charakterystykę częstotliwościową. Zilustrujemy działanie powyższej funkcji na przykładzie okna Hanninga. Poniższy skrypt

```
1 clear all
2 close all
3 N=64; % Długośc okna (ilość próbkowań)
4 wvtool(hann(N))
```

daje jako wynik swojego działania rysunek 4.10, na podstawie, którego można



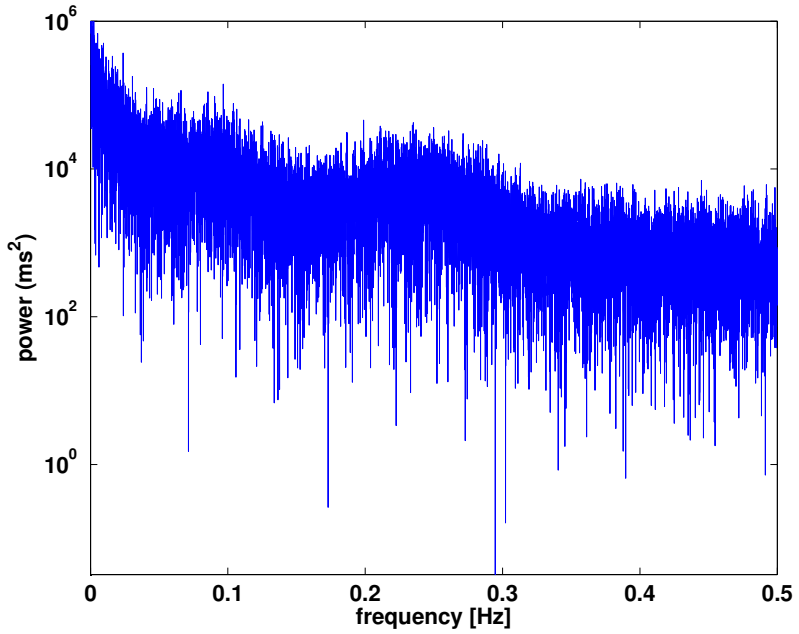
Rysunek 4.10. Charakterystyka okna Hanninga dla długości $N = 64$

zapoznać się z charakterystyką okna. Czytelnik jest zachęcany do zapoznania się się z innymi typami okien zaimplementowanymi w pakiecie Matlab. Dostęp do odpowiedniej dokumentacji można otrzymuje się wydając komendę doc window.

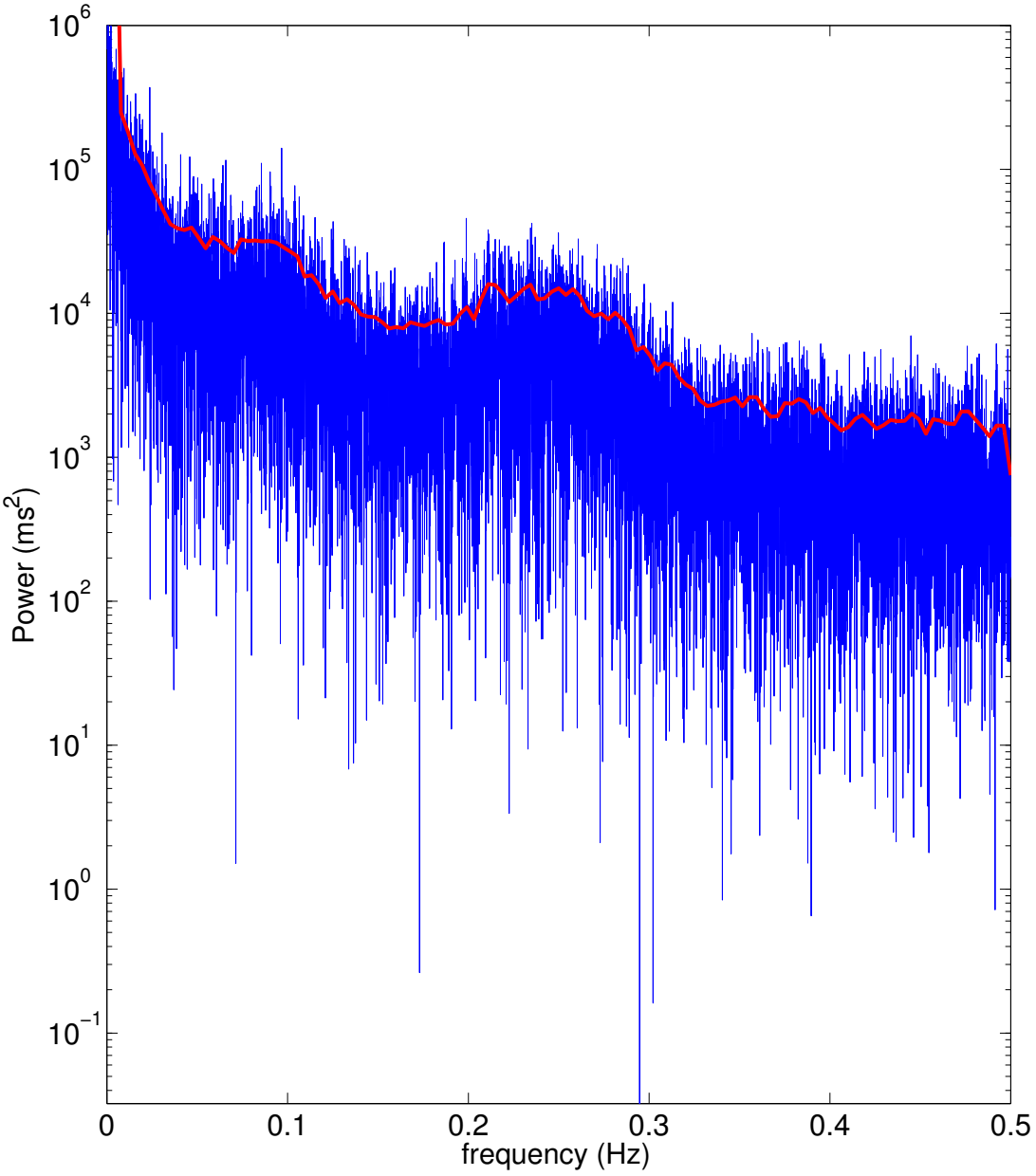
4.2. Analiza widmowa rytmu serca.

```
clear all
close all
ecg=load('k20_11.55.ce');
frequency=linspace(0,1,length(ecg));
figure
% obliczamy transformate fouriera
yecg=fft(ecg);
nfh=length(yecg);
%wykreślamy periodogram
semilogy(frequency(1:nfh/2),abs(yecg(1:nfh/2)).^2/nfh);
axis([0 0.5 0 10^(6)]);
xlabel('frequency [Hz]'); ylabel('power (ms^2)');
fftheart.pdf
% Tworzymy okno Hanninga składającę się z 256 punktów
winhw=hanning(256,'periodic');
% Obliczamy periodogram metodą Welcha przy 50% nakładaniu się okien
% (256/2=128)
[y,freq]=pwelch(ecg,winhw,128,[],1);
figure
semilogy(frequency(1:nfh/2),abs(yecg(1:nfh/2)).^2/nfh); hold on;
semilogy(freq,y,'-r','LineWidth', 2.5)
axis([0 0.5 0 10^(6)]); ;

xlabel('frequency (Hz)');
ylabel('Power (ms^2)');
figure
label=[0 20 40 50]
overlap=256*[10 20 40 50]/100;
% generujemy wykresy dla róznych stopni nakładania się
for i=1:length(overlap)
[y,freq]=pwelch(ecg,winhw,overlap(i),[],1);
subplot(2,2,i);semilogy(freq,y,'-r','LineWidth', 2.) ;
axis([0 0.5 0 10^(6)]);
text(0.2,10^5,['overlap=(',num2str(label(i)),'%)']);
```

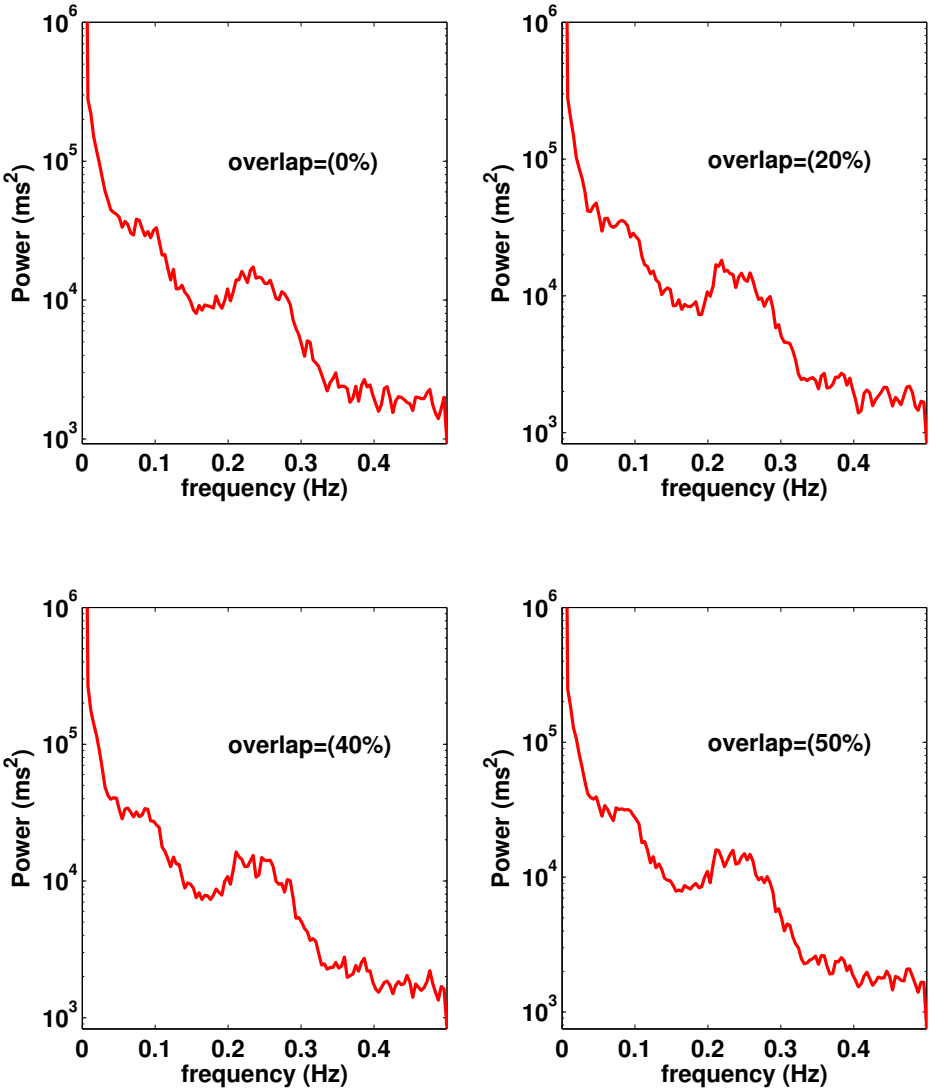
```
35
36 xlabel('frequency (Hz)');
37 ylabel('Power (ms^2)');
38 end
```



Rysunek 4.11. Periodogram dla sygnał RR



Rysunek 4.12. Periodogram Welcha dla sygnału RR z wykorzestaniem okna Hanninga o długości $N = 256$ przy 50% nakładaniu



Rysunek 4.13. Periodogramy Welcha dla róznych stopni nakładania się onkan Hanninga (N=256)

5. Podstawy analizy obrazów 2D

Samo pojęcie obrazu jest mocno ogólne i początkowo wiązało się ściśle ze zmysłem wzroku oraz sposobem postrzegania przez daną osobę otoczenia. Człowiek wraz z rozwojem cywilizacji stosował różne sposoby zapisywania obrazów. Obrazy były malowane na ścianach jaskiń, przenoszone na płótno, a po wprowadzeniu do użytku aparatu, na kliszę fotograficzną. Wraz z wynalezienia aparatu rentgenograficznego człowiek uzyskał dostęp do obrazów pochodzących nie tylko z wąskiego spektrum światła widzialnego 380–740nm, a po upowszechnieniu komputerów obrazy zaczęły być zapisywane w postaci cyfrowej, która jest łatwa do przechowywania na pamięciach masowych. We współczesnej medycynie metody obrazowania są już osobną gałęzią dysponującą pokaźnym rodzajem metod, z których wymienimy miedzy innymi:

— komputerowa tomografia
— PEC
— SPECT
— MRI

Taki wachlarz metod pozwala na praktycznie bezinwazyjne zapisanie w postaci cyfrowej obrazu kości, tkanek i narządów.

Obrazy uzyskane w trakcie badań medycznych, po poddaniu ich obróbce cyfrowej trafiają w końcowym etapie do lekarza, który jest odpowiedzialny za ich analizę i postawienie diagnozy. Dlatego niezmiernie ważna staje, aby w trakcie obróbki cyfrowej nie stracić ważnych szczegółów przetwarzanego obrazu.

W niniejszym skrypcie nie będziemy omawiać sposobów pozyskiwania obra-

zów medycznych, zajmiemy się jedynie ich cyfrową obróbką do której będziemy wykorzystywać pakiet MATLAB oraz Octave.

5.1. Formaty zapisu obrazów dostępne w pakiecie MATLAB.

W trakcie zapisu cyfrowego obraz dwu wymiarowy dzielimy na dyskretną siatkę punktów (x, y) , w której współrzędna x przyjmuje wartości ze zbioru $x \in \{1, 2, \ldots, M\}$, a współrzędna $y \in \{1, 2, \ldots, N\}$ co daje obraz obraz składający się z $M \times N$ pikseli. Każdemu pikselowi przyporządkowujemy funkcję $f(x, y)$, której wartości odpowiadają intensywności obrazu. W pakiecie MATLAB większość komend operujących na obrazach pobiera współrzędne punktu podane w dyskretnym układzie współrzędnych opartym o piksele:

	1	2	3	4		M	x
1	f(1,1)	f(1,2)	f(1,3)	f(1,4)	...	f(1,M)	
2	f(2,1)	f(2,2)	f(2,3)	f(2,4)	...	f(2,M)	
3	f(3,1)	f(3,2)	f(3,3)	f(3,4)	...	f(3,M)	
	⋮	⋮	⋮	⋮	⋱	⋮	
N	f(N,1)	f(N,2)	f(N,3)	f(N,4)	...	f(N,M)	
y							

W zależności od sposobu zapisu intensywności $f(x, y)$ może przybierać wartości skalarne lub wektorowe. Ze względu na sposób realizacji intensywności obrazu możemy wyróżnić cztery podstawowe kategorie zapisu obrazu:

— binarny (ang. *binary*)

W formacie binarnym funkcja $f(x, y)$ może przyjmować jedynie dwie wartości 0 (czarny) lub 1 (biały).

— intensywności (ang. *intensity*)

Format intensywności służy do zapisu rysunków w skali szarości definiując intensywność odcienia szarości liczbą z przedziału $\langle 0, maxn \rangle$. Maxn jest maksymalną liczbą jaką mozna zapisać przy określonej długości bitów przypadających na jeden piksel. Na przykład dla uint8 maxm=255, a dla uint16 maxn=65535

— RGB

w formacie RGB każdemu pikselowi przyporządkowujemy wektor $[R, G, B]$, którego składowe odpowiadają intensywnością podstawowych kolorów Red, Green, Blue.

— indeksowany (ang. *indexed*)

W formacie indeksowanym funkcja f(x,y) zwraca liczbę naturalną k będącą indeksem oddzielnej tablicy, której elementu definiują barwy za pomocą kolorów podstawowych [R,G,B]. Taką tablicę asocjacyjną nazywa się mapa kolorów.

Przykład 11. Wczytaj do przestrzeni roboczej MATLABA obraz dog.jpeg, który ma wymiar 800x600 pikseli i jest zapisany w formacie True Color RGB. Skonwertuj obraz na skalę szarości szarości.

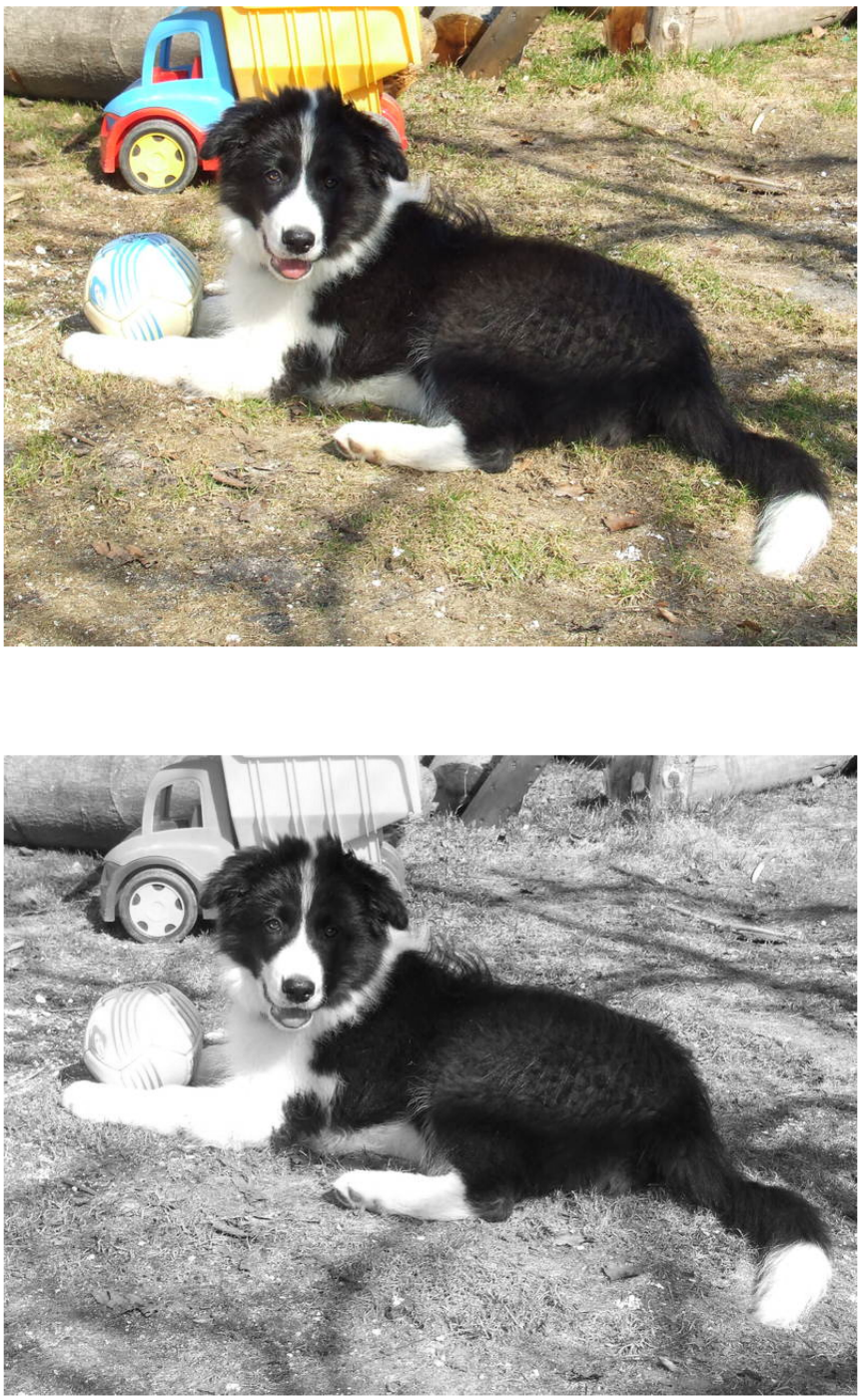
```
close all;
clear all;
% Pod zmienną I ładujemy rysunek w formacie RGB
% wymiar rysunku 800x600
[I]=imread('dog_s.jpeg');
% Sprawdzamy wymiar rysunku
picture_dimension=size(I)
%  w wyniku dostajemy 600x800x3
% proszę zwrócić uwagę  na zmianę kolejności 800 i 600
% oraz na fakt, że w przypadku formatu RGB, na każdy kolor
% przypada jeden wymiar tablicy

% wyświetlamy rysunek
```

```
14 imshow(I);
15 % przekonwertujemy rysunek z formatu RGB na skale szarości
16 % zgodnie ze standardem NTSC
17  IG= .2989*I(:,:,1) ...
18  +.5870*I(:,:,2) ...
19  +.1140*I(:,:,3);
20 figure;
21 imshow(IG);
```

W powyższym przykładzie należy zwrócić szczególna uwagę na format indeksowania tablicy I, która ma wymiar 600x800x3. Źródłowy rysunek składa się z 800 pikseli w poziomie (oś x) oraz 600 pikseli w pionie (oś y). Standardowo w pakietach MATLAB oraz OCTAVE współrzędne z osi y zapisywane są w wierszach, natomiast współrzędne z osi x zapisywane są w kolumnach. W formacie RGB każdy z podstawowych kolorów Red, Green, Blue jest dla każdego piksela zapisywany osobno w dodatkowym wymiarze tablicy. Obraz I został skonwertowany do skali szarości zgodnie ze standardem NTSC, w którym przyjmuje się, dla każdego z kolorów podstawowych $[R, G, B]$, współczynniki korekcji $[.2989, 0.5870, .1140]$ skalujące intensywności poszczególnych kolorów zgodnie z czułością oka ludzkiego dla danej barwy.

Obrazem w skali szarości można bardzo łatwo operować. Wstawiając dwie dodatkowe komendy do wcześniejszego przykładu

```
1 %
2 % Skopiuj zawartość poprzedniego listingu
3 %
4 IC=imcomplement(IG);
5 IG=immultiply(IG,0.9);
6 figure;
7 imshow(IC)
8 figure
9 imshow(IG);
```



Rysunek 5.1. Konwersja obrazu RGB na skale szarości



Rysunek 5.2. Zmiana kontrastu obrazu oraz negatyw orginału

możemy zmniejszyć kontrast oraz stworzyć negatyw. Pliki z obrazami zapisanymi w standardowych formatach (typu bmp, png, jpeg, etc.), mogą być bez problemu wczytane komendą imread. Natomiast pliki formatu DICOM (Digital Imaging and Communications in Medicine) wczytuje się komendą dicomread (od wersji 2013).

Przykład 12. Wykonując badania MRI czasami można uzyskać dostęp do surowych danych (ang. *raw data*) zapisanych w binarnym pliku. Do pliku binarnego dołączone są informacje tekstowe na temat wymiaru obrazu oraz obszaru pamięci zajmowanego przez jeden piksel.

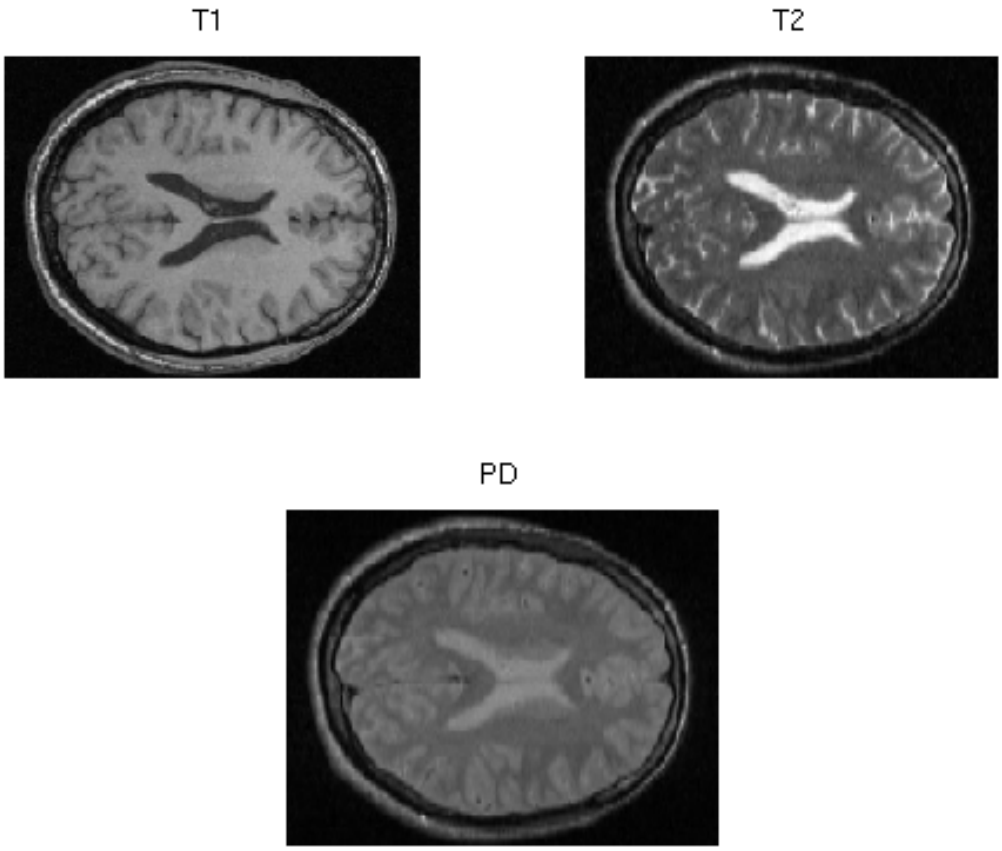
W plikach t-t1.img, t-t2.img, t-pd.img udostępnionych dzięki uprzejmości prof. GUMED dr. hab. Piotra Bogusi, znajdują się surowe dane z MRI mózgu dla sygnałów odpowiednio T1, T2 oraz PD. Pliki opatrzone są komentarzem

```
171 x 220 = 37620
gray: 0 - 255
1 pixel = 1 unsigned char = 1 byte
143 slides PD T1 T2
171 x 220 x 143   -   ok. 5 MB
```

Wczytaj i wykreśl te dane w Matlabie i wyeksportuj obraz T2 do pliku w formacie PNG.

```
close all;
clear all

% Tworzymy rączkę do pliku (handle)
fid = fopen('t-t1.img');
% wczytujemy plik w formacie 171 na 220
% 1 pixel ma długość 8bitów
T1= fread(fid, [171, 220],'*uint8');
subplot(2,2,1);imshow(T1)
title('T1');
fid = fopen('t-t2.img');
T2 = fread(fid, [171, 220],'*uint8');
% konewrtujemy pixel na podwójną precyzję
T2_dub=im2double(T2);
subplot(2,2,2);imshow(T2)
title('T2');
fid = fopen('t-pd.img');
PD = fread(fid, [171, 220],'*uint8');
subplot(2,2,3.5);imshow(PD)
title('PD');
% tworzymy mapę kolorów
map=colormap('Gray')
%zapisujemy do pliku
imwrite(T2,map,'T2.png');
```

Obrazay T1-weighted i T2-weighted są łatwo rozróżnialne. Podczas obrazowania T2-weighted bardzo silny sygnał MRI daje woda, dlatego płyn mózgowo-rdzeniowy jest intensywnie widoczny (wybarwia się na biały kolor patrz rysunek 5.3T2), natomiast na obrazach T1 płyn jest ciemnego koloru (patrz rsunek 5.3T1). W literaturze na temat MRI można znaleźć klasyfikacje odpowiedzi różnych struktur na odpowiednie sygnały T1, T2, PD. Jednoczesna analiza trzech sygnałów pozwala przyporządkować zobrazowane obszary mózgu do odpowiednich grup [8].



Rysunek 5.3. Periodogram dla sygnał RR

Przykład 13. Przeprowadź konwersje obrazu T2 z poprzedniego przykładu do formatu indeksowanego zmieniając mapę kolorów.

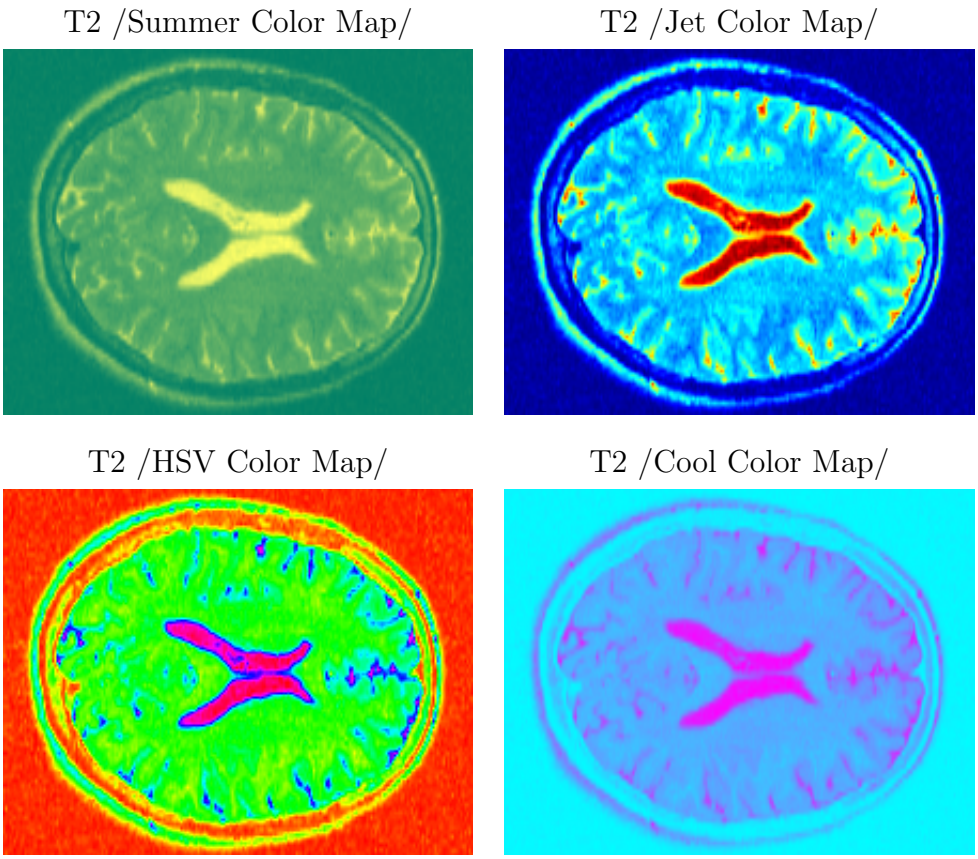
```
1 close all;
2 clear all
3 fid = fopen('t-t2.img');
4 T2 = fread(fid, [171, 220],'*uint8');
5 % konewrtujemy pixel na podwójną precyzję
6 T2_dub=im2double(T2);
```

```
imshow(T2)
title('T2 /Summer Color Map/');
colormap('Summer');
% zmieniamy mapę kolorów
figure
imshow(T2)
title('T2 /Jet Color Map/');
colormap('Jet');

figure
imshow(T2)
title('T2 /HSV Color Map/');
colormap('HSV');

figure
imshow(T2)
title('T2 /Cool Color Map/');
colormap('Cool');
```

Proszę zwrócić uwagę, że mapa kolorów jest zwykłą tablicą w której liczby w kolumnach odpowiadają intensywności kolorów:

```
      Red         Green       Blue
map =
            0           0           0
       0.0039      0.0039      0.0039
       0.0078      0.0078      0.0078
       0.0118      0.0118      0.0118
       0.0157      0.0157      0.0157
       0.0196      0.0196      0.0196
            .           .           .
            .           .           .
            .           .           .
       0.9882      0.9882      0.9882
```

```
13      0.9922    0.9922    0.9922
14      0.9961    0.9961    0.9961
15      1.0000    1.0000    1.0000
```



Rysunek 5.4. Zmiana mapy obrazu T2

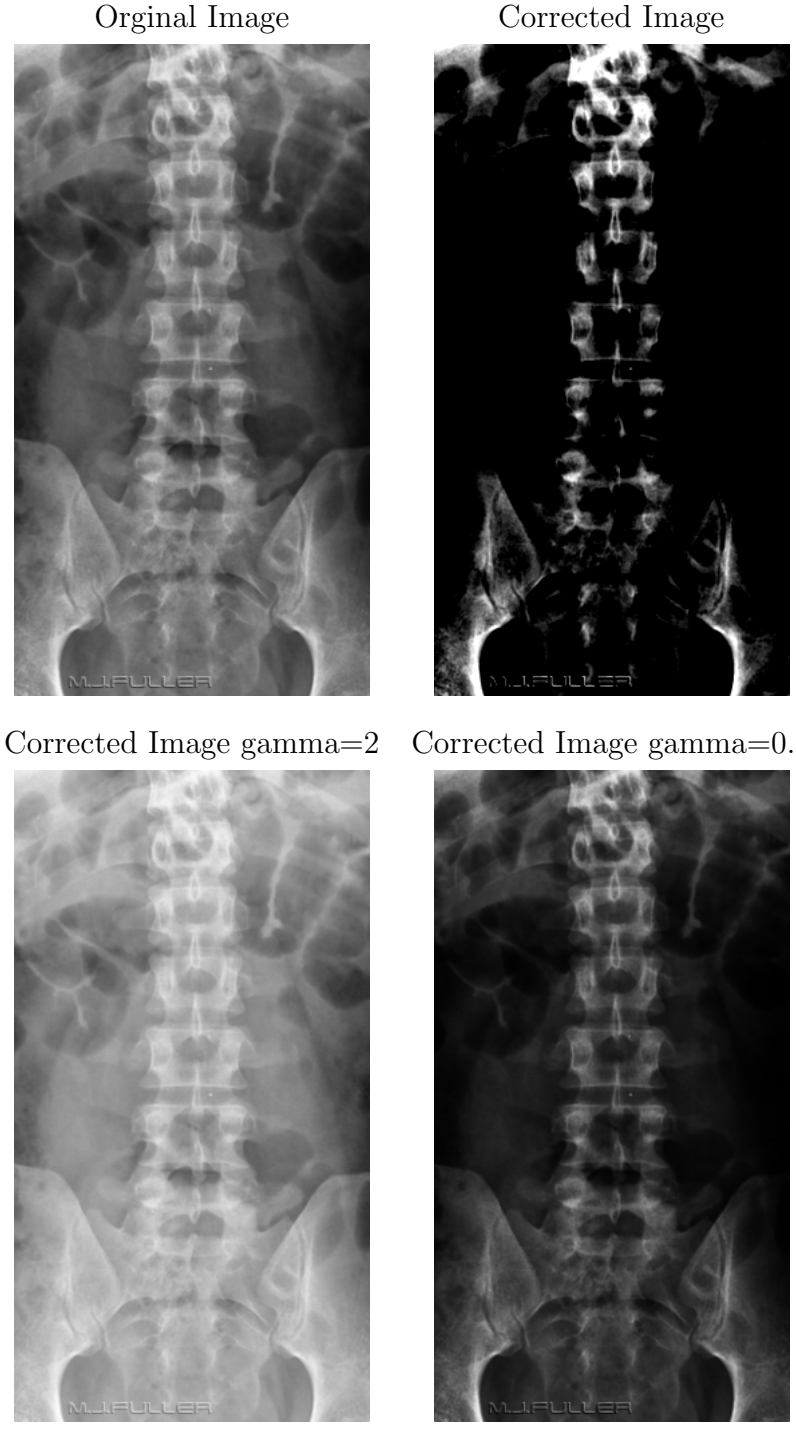
Obrazy można łatwo obrabiać zmieniając intensywność odcieni poszczególnych kolorów wykorzystując wbudowaną funkcję Jout = imadjust(In,[low_in; high_in], [low_out; high_out],gamma), która skaluje intensywności obraz In z zakresu [low_in; high_in] na przedział [low_out; high_out] przy użyciu funkcji gamma.

Przykład 14.

Wczytaj obraz rentgenowski GW266H488.jpeg[1] i przeprowadź korekcję stopnia szarości definiując różne funkcje gamma

```
clear all;
close all;
[lumbarspine]=imread('GW266H488.jpeg');
subplot(2,2,1);imshow(lumbarspine);
title('Orginal Image')
map=colormap('Gray');
g=imadjust(lumbarspine,[0.5 0.8], [0 1]); % pobieramy
% z obrazu odcienie z [0.5 0.8] i skalujemy je do
%pełnego zakresu [0 1]
subplot(2,2,2);imshow(g)
imwrite(g,map,'lumb-s.png');
title('Corrected Image')

g=imadjust(lumbarspine,[ ], [ ], 2);
subplot(2,2,3);imshow(g)
imwrite(g,map,'lumb-g2.png');
title('Corrected Image gamma=2')

g=imadjust(lumbarspine,[ ], [ ], 0.5);
subplot(2,2,4);imshow(g)
imwrite(g,map,'lumb-g0p5.png');
title('Corrected Image gamma=0.5')
```

[1] Obraz pochodzi ze strony http://www.wikiradiography.com/page/Lumbar+Spine+Radiography autorem obrazu jest M.J. Fuller.

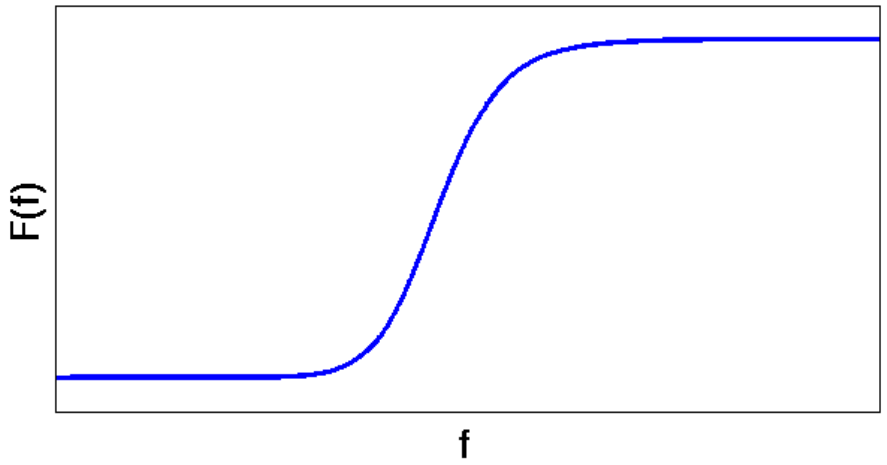


Rysunek 5.5. Obróbka obrazu za pomocą funkcji imadjust

Histogram obrazu pozwala szybko sprawdzić ilość pikseli posiadającą określoną intensywność. Z tego powodu jest ona wygodnym narzędziem za pomocą którego można wybierać progowe wartości dla różnego rodzaju filtrów. Przeprowadzimy teraz transformację która pozwoli nam podnieść kontrast wybranej struktury. Przyjmijmy następującą postać funkcji transformacyjnej

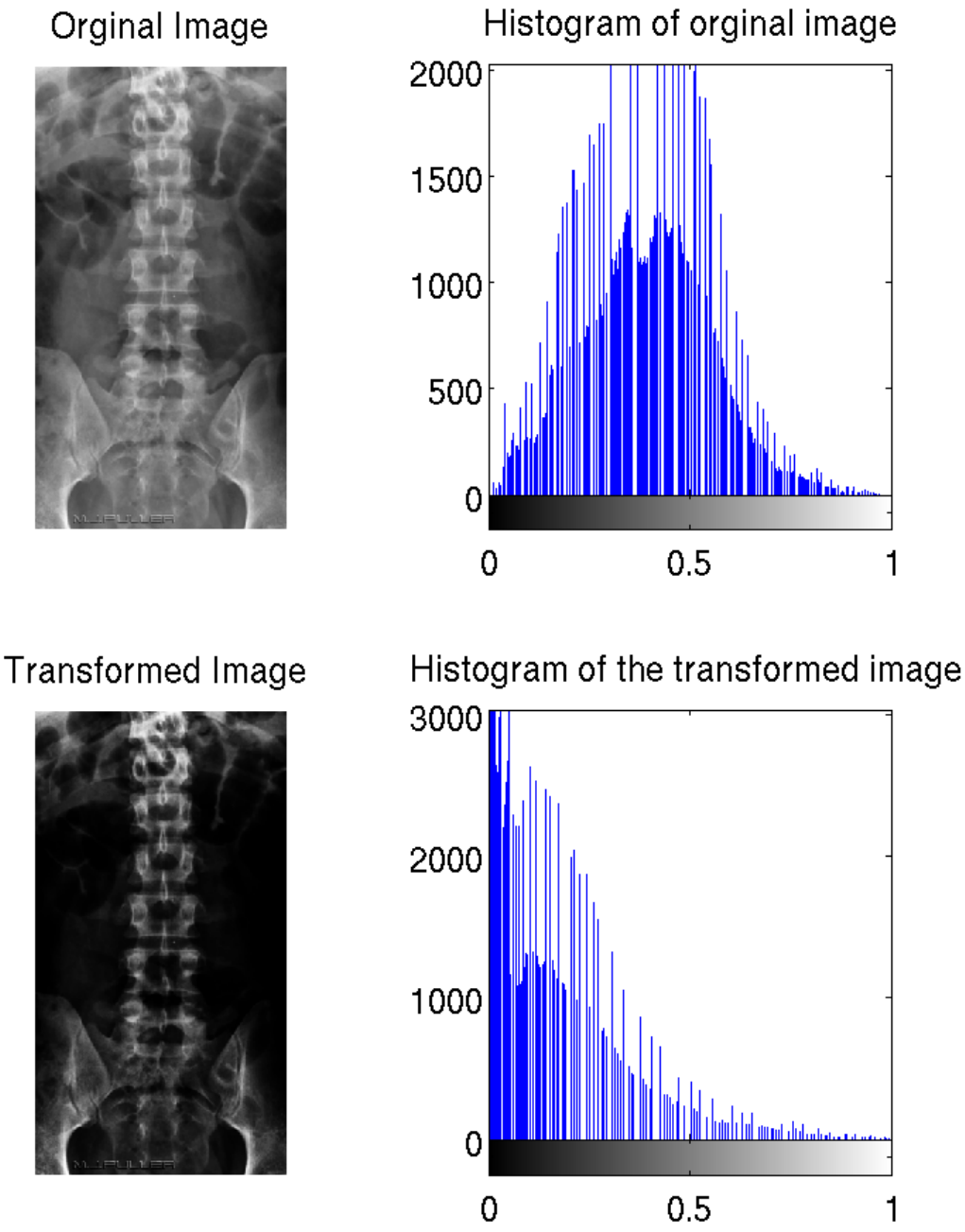
$$F(r) = \frac{1}{1 + (\eta/f)^E}, \tag{5.1}$$

gdzie η jest progiem transformacji, f jest transformowanym obrazem, E jest wykładnikiem większym od jeden.




```
clear all;
close all;
[lumb]=imread('GW266H488.jpeg');
% Konwertyjemy obraz z trzech kanałów RGB na stopnie szarości
lumbgray = .2989*lumb(:,:,1) +.5870*lumb(:,:,2) +.1140*lumb(:,:,3);
% skalujemy wartości do przedziału <0,1>
lumbi=im2double(lumbgray);

subplot(2,2,1);imshow(lumbi);
title('Orginal Image')

```

```
12 % tworzymy histogram
13 subplot(2,2,2);imhist(lumbi);
14 title('Histogram of orginal image')
15
16 % tworzymy transformację
17 % wybieramy punkt przegięcia
18 kappa=0.42;
19 %wybieramy wykładnik
20 E=4;
21 trans=1./(1+(kappa./(double(lumbi)+eps).^E));
22 %trans=log(1+1*double(lumbi));
23 transb=mat2gray(trans);
24 subplot(2,2,3);imshow(transb);
25 title('Transformed Image ')
26 subplot(2,2,4);imhist(transb);
27 title('Histogram of the transformed image ')
28 imwrite(transb,'imlog.png');
```



Rysunek 5.6. Histogram i transformacja

6. Nieliniowe metody analizy obrazów 2D.

7. Dodatki

Listing m-pliku generującego rysunek 2.1 z rozdziału 2.

```
t=0:0.01:4*pi;
y=sin(t);

figure
subplot(3,1,1);h=plot(t,y)
set(h,'LineWidth',2)
ylabel('signal [arb units]')
text(3.5*pi,0.5,'(a)')


t1=0:pi/8:4*pi;
y1=sin(t1);
subplot(3,1,2);h1=stem(t1,y1,'fill','--')
ylabel('signal [arb units]')
set(get(h1,'BaseLine'),'LineStyle',':')
text(3.5*pi,0.5,'(b)')

set(h1,'MarkerFaceColor','red')
subplot(3,1,3);h3=stairs(t1,y1)
set(h3,'LineWidth',2)
ylabel('signal [arb units]')
xlabel('t [s]')
text(3.5*pi,0.5,'(c)')
```

Listing m-pliku generującego rysunek 4.7 z rozdziału 4.1.

```
clear all;
close all;
fs=100
dt=1/fs; % odstęp między kolejnymi próbkami
t=0:dt:0.63;
y=1.5*sin(2*pi*2*(t)-1)+sin(2*pi*4*(t)-1);
figure(1);
hold on
plot(t,y,'b--');
 set(gca,'xtick',[],'ytick',[])
text(10.1,-2.8 ,['(a)']);
axis([-3*0.63 3*0.63 -3 2]); % skalowanie osi
ylabel('amplitude (arb. units)');
xlabel('time (arb. units.)');
t=0.63:dt:2*0.63;
plot(t,y,'k-');
t=2*0.63:dt:3*0.63;
plot(t,y,'b--');
t=-1*0.63:dt:0*0.63;
plot(t,y,'k-');
t=-2*0.63:dt:-1*0.63;
plot(t,y,'b--');
t=-3*0.63:dt:-2*0.63;
plot(t,y,'k-');
```

Bibliografia

[1] G. Blanchet, M. Charbit „Digital Signal and Image Processing using MATLAB" ISTE Ltd. (2006)

[2] R. C. Gonzalez, R. E. Woods, S. L. Eddins „Digital Image Processing Using Matlab" Pearson Education, Inc. (2004)

[3] T. Noeks „Lore of Running", Leisure Press, Champaign, Illinois, 3rd ed. (1991)

[4] J. L. Semmlow „Biosignal and Biomedical Image Processing" Marcel Dekker, INC (2004)

[5] Zahir M. Hussain., Amin Z. Sadik, Peter O'Shea „Digital Signal Processing", Springer-Verlag, Berlin Heidelberg (2011)

[6] R. Rangayyan „Biomedical signal analysus - A case study approach"

[7] A.V. Oppenheim, R. W. Schafer, „Digital Signal Processing", Prentice-Hall, Englewood Cliffs, NJ, (1975).

[8] C. Westbrook, C. Kaut Roth, J. Talbot „ MRI in Practice", Blackwell Science Ltd, Oxford (2000)