WSTEP DO MATLABA

BYDGOSZCZ 2002

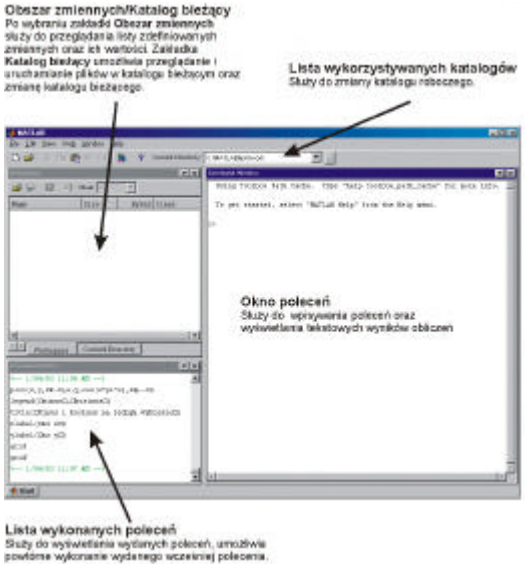
1. MATLAB

- Matlab jest interaktywnym systemem realizacji obliczen numerycznych. Nazwa Matlab pochodzi od slów Matrix Laboratory (Laboratorium macierzy).
- Cleve Moler napisal pierwsza wersje Matlaba w roku 1970 jako narzedzie upraszczajace korzystanie z bibliotek numerycznych LAPACK oraz BLAS. Obecnie jest to niezalezne narzedzie komercyjne.
- Istotnym rozszerzeniem funkcjonalnosci Matlaba sa niezalezne zbiory funkcji (toolboxy) sluzace do rozwiazywania wyspecjalizowanych problemów.
- Matlab uwalnia uzytkownika od rozwiazywania wielu zadan zwiazanych z metodami numerycznymi. Pozwala to skupic sie na rozwiazaniu problemu oraz umozliwia eksperymentowanie.
- Skomplikowane operacje moga byc przeprowadzone w niewielkiej liczbie kroków, przy znajomosci zaledwie kilku komend.
- Mozliwe jest tworzenie wlasnych zestawów funkcji oraz interfejsów uzytkownika dla potrzeb wlasnej aplikacji uruchamianej w srodowisku Matlaba.
- Dostepne sa doskonalej jakosci narzedzia wizualizacyjne, których efekty dzialania moga byc przenoszone do innych aplikacji.

2. URUCHOMIENIE MATLABA

Uruchomienie Matlab'a w systemie Microsoft Windows polega na dwukrotnym kliknieciu lewym klawiszem myszki (pojedynczym jesli jest aktywny Active Desktop) ikony umieszczonej na pulpicie.

Jesli uruchamiamy program z okna DOS, piszemy komende `matlab` po znaku zachety. Wpisujemy `matlab` znaku zachety równiez dla systemów z rodziny UNIX.
Po uruchomieniu Matlab wyswietla okno komend, które jest przeznaczone do wprowadzania komend oraz wyswietlania tekstowych wyników obliczen – rysunek 1.

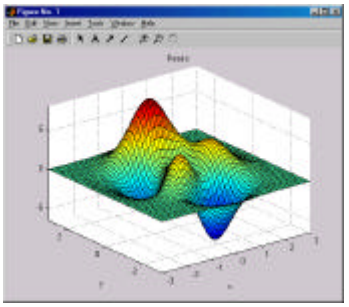


Rys.1. Okno programu Matlab

Polecenia wpisujemy za znakiem zachety:

```
>>      w przypadku wersji komercyjnej MATLABA
EDU>    w przypadku wersji edukacyjnej
```

Matlab odpowiada na wydawane polecenia natychmiast. Wyniki dzialania polecen sa wyswietlane w oknie polecen lub jesli wynikiem jest wykres w niezaleznym okienku graficznym – rysunek 2.



Rys.2. Okno graficzne programu Matlab

Przechodzenie pomiedzy oknami programu Matlab nastepuje przez kliknieciu lewym klawiszem myszki na pasku tytulowym.

3. MATLAB JAKO KALKULATOR

Podstawowymi operatorami matematycznymi w Matlabie sa: `+`, `-`, `*`, `/` oraz `^`. Wykorzystywane sa jednoczesnie z nawisami okraglymi: `( )`. Symbol `^` oznacza potegowanie. Polecenia wpisujemy po znaku zachety: **>>**.

```
>> 2+3/4*5
   ans =
       5.7500
```

Jakie to dzialanie `2+3/(4*5)`, a moze `2+(3/4)*5`? Operacje arytmetyczne wykonywane sa w nastepujacym porzadku:

1. wyrazenia w nawiasach,
2. potegowanie np.: `2+3^2` $\Rightarrow$ `2+9 = 11`
3. `*`, `/`, od lewej do prawej np.: `3*4/5=12/5`,
4. `+`, `-`, od lewej do prawej np.: `3+4-5=7-5`.

Dlatego, w pokazanym przykladzie obliczana jest wartosc wyrazenia `2+(3/4)*5`.

4. REPREZENTACJA LICZB W MATLABIE

Matlab dysponuje kilkoma róznymi rodzajami liczb

Rodzaj	Przyklady
Calkowite	`1362`, `-217897`
Rzeczywiste	`1.234`, `-10.76`
Zespolone	`3.21-4.3i` ($i=\sqrt{-1}$)
Nieskonczonosc	`Inf`, `-Inf` (np.: 1/0)
Nie liczba	`NnN` (np.: 0/0)

Do reprezentowania bardzo malych i bardzo duzych liczb wykorzystywana jest notacja naukowa z symbolem `e`:

`-1.3412e+03`=`-1.3412`$\times 10^{3}$=`-1341.2`
`-1.3412e-01`=`-1.3412`$\times 10^{-1}$=`-0.13412`

Wszystkie obliczenia realizowane sa z wykorzystaniem liczb o podwójnej precyzji, co oznacza okolo 15 cyfr znaczacych.
Sposób wyswietlania liczby jest kontrolowany poleceniem `format`. Zamieszona ponizej tabela pokazuje najczesciej wykorzystywane formaty.

Polecenie	Przyklad
`format short`	31.4162
`format short e`	3.1416e+01
`format long e`	3.141592653589793e+01
`format bank`	31.42

Powrót do podstawowego formatu wyswietlania liczb nastepuje po wydaniu polecenia `format` bez dodatkowych opcji. Polecenie:

```
>> format compact
```

Usuwa podczas wyswietlania wyników puste linie, co zwieksza ilosc pokazywanej na ekranie informacji.

5. ZMIENNE I STALE

```
>> x=3-2^4
   ans =
   -13
>> ans*5
   ans =
   -65
```

Wynikowi pierwszego dzialania przypisywana jest nazwa `ans`. Wartosc przypisana nazwie (zmiennej) `ans` jest wykorzystywana w drugim dzialaniu. Zmiennej `ans` przypisywany jest wynik drugiego dzialania.
Uzytkownik moze samodzielnie przypisywac nazwy wynikom obliczen (tworzyc zmienne) w operacji przypisania. W operacji tej wartosc liczbowa znajdujaca sie po prawej stronie znaku = jest wiazana z znajdujaca sie po lewej stronie znaku = nazwa zmiennej.

```
>> x=3-2^4
   x =
   -13
>> y=x*5
   y =
   -65
```

W tym przypadku zmiennej x przypisano wartosc -13, a y – -65. Zmienne `x` i `y` pojawily sie w obszarze zmiennych Matlaba (patrz rysunek 1) i moga byc wykorzystywane w dalszych obliczeniach.
Kazda zmienna musi miec przypisana wartosc liczbowa zanim bedzie mogla byc wykorzystana w obliczeniach.
Dozwolone sa nazwy zmiennych zawierajace dowolna kombinacje liter, cyfr oraz znaku „_”, rozpoczynajace sie litera. Powinny skladac sie z nie wiecej niz 31 znaków. W nazwach zmiennych rozpoznawana jest wielkosc liter tak wiec nazwy `mojasuma` i `mojaSuma` to dwie rózne zmienne. Przykladami prawidlowych nazw zmiennych sa:

```
NetCost, Left2Pay, x3, X3, z25_c5
```

Przykladami niedozwolonych nazw zmiennych sa:

```
Net-Cost, 2pay, %x, @sign
```

Ogólne zalecenie dotyczace nazw zmiennych mówi, ze powinny one dobrze opisywac zapamietywana w zmiennej wartosc. Ich nazwa nie powinna byc nazwa stalych Matlaba lub istniejacych funkcji, np.:

```
eps=2.2204e-16
pi = 3.14159...
```

Stale w Matlabie to:

Nazwa	Znaczenie
`realmax`	Najwieksza dostepna liczba rzeczywista
`realmin`	Najmniejsza dostepna liczba rzeczywista
`eps`	Blad maszynowy
`pi`	Wartosc liczby π
`Inf`	Nieskonczonosc
`NaN`	Nie liczba

Nie nalezy równiez wykorzystywac nazwy `ans` – poniewaz jest ona automatycznie przypisywana wynikom wszystkich dzialan, którym uzytkownik nie przypisal wlasnej nazwy.
Jesli w trakcie obliczen beda wykorzystywane liczby zespolone nie powinno sie jako nazw zmiennych uzywac liter `i` oraz `j`, które oznaczaja wartosc $\sqrt{-1}$.
Jesli liczby zespolone nie sa uzywane mozna korzystac z liter `i` oraz `j` jako nazw zmiennych.
W przykladzie ponizej najpierw wyswietlana jest wartosc stalych `i` oraz `j` a nastepnie nazwie `i` przypisuje sie nowa wartosc 3.

```
>> i,j, i=3
   ans =
   0 + 1.0000i
   0 + 1.0000i
   i =
   3
```

Nazwy stalych moga wystepowac wylacznie na prawej stronie wyrazen matematycznych

6. UKRYWANIE WYNIKÓW

Wyniki posrednich obliczen moga zostac ukryte. Jesli chcemy aby tak bylo polecenie konczymy srednikiem.

```
>> x = 5;
>> y = 59^(1/2);
>> z = y^0.75+x^0.25
z =
   6.1093
```

Dzialania przypisujacych wyniki dzialan do zmiennej x oraz y nie zostaly wyswietlony na oknie polecen (obie zmienne zostaly utworzone).

7. WIELE POLECEN W JEDNEJ LINII TEKSTU

W jednej linii tekstu mozna wprowadzic kilka kolejno wykonywanych polecen, oddzielonych od siebie przecinkiem lub srednikiem.

```
>> x=-13; y=5*x, z=x^2+y
   y =
   -65
   z =
   104
```

8. FUNKCJE MATLABA

Matlab zostal miedzy innymi wyposazony w nastepujace funkcje trygonometryczne: `sin()` – sinus, `cos()` – kosinus, `tan()` – tangens. Operuja one na argumentach podawanych w radianach (Matlab nie dysponuje mozliwoscia pracy na jednostkach katowych innych niz radiany).

Przyklad 8.1: *obliczyc wspólrzedne x i y punktu znajdujacego sie na okregu o promieniu 5 ze srodkiem w poczatku ukladu wspólrzednych, który wznosi sie nad poziom o kat 30° = p/6 radianów.*

```
>> x=pi*cos(pi/6), y=pi*sin(pi/6)
   x =
   4.3301
   y =
   2.500
```

Istnieja kofunkcje funkcji trygonometrycznych w postaci `asin()` = $\sin^{-1}$, `acos()` = $\cos^{-1}$, `atan()` = $\tan^{-1}$. Wyniki podawane sa w radianach.

```
>> acos(x/5), asin(x/5)
   ans =
   0.5236
   ans =
   0.5336
>> pi/6
   ans =
   0.5236
```

Matlab dysponuje olbrzymia liczba podstawowych funkcji matematycznych (skrócona lista z opisem na koncu rozdzialu)

```
>> x=9
>> sqrt(x),exp(x),log(sqrt(x)),log10(x^2+6)
   ans =
   3
   ans =
   8.1031e+03
   ans =
   1.0986
   ans =
   1.9395
```

Funkcja `sqrt()` oblicza pierwiastek kwadratowy podanego argumentu, `exp()` – wartosc funkcji e^x, funkcja do niej odwrotna jest `log()`.

```
>> format long e, exp(log(9)), log(exp(9))
   ans =
   9.000000000000002e+00
   ans =
   9
>> format short
```

Funkcja `log10()` oblicza logarytm o podstawie 10 dowolnego dopuszczalnego argumentu.
W Matlabie istnieje funkcji sluzacych do zaokraglania liczb rzeczywistych do calkowitych.

```
>> x1 = -1*pi, x2=pi
x1 =
  -3.1416
x2 =
   3.1416
>> round(x1), round(x2)
ans =
   -3
ans =
    3
>> fix(x1), fix(x2)
ans =
   -3
ans =
    3
>> floor(x1), floor(x2)
ans =
  -4
ans =
    3
>> ceil(x1), ceil(x2)
ans =
  -3
ans =
    4
```

Funkcja `round` zaokragla do najblizszej liczby calkowitej, `fix` – do najblizszej wiekszej liczby calkowitej dla wartosci mniejszych od 0 i do najblizszej mniejszej liczby calkowitej dla wartosci wiekszych od 0, `floor` – do najblizszej mniejszej liczby calkowitej, `ceil` – do najblizszej wiekszej liczby calkowitej. Ustalenie znaku liczby umozliwia funkcja `sign`.

```
>> sign(x1), sign(x2)
ans =
   -1
ans =
    1
```

Reszte z dzielenia liczby przez podana wartosc calkowita mozemy ustalic funkcja `rem`.

```
>> rem(x1,3)
ans =
   -0.1416
```

Natomiast funkcja `abs` umozliwia ustalenia wartosci bezwzglednej liczby.

```
>> abs(x1
ans =
   3.1416
```

9. EDYCJA POLECEN

Edycje, wywolanie, wykonanie wszystkich wydanych wczesniej polecen umozliwiaja klawisze kursora oraz kombinacje klawiszy (oraz okno z lista wydanych polecen – patrz rysunek 1). Wprowadzajac bledne polecenie:

```
>> (1 + sqt(5))/2
```

spowodujemy wyswietlenie w oknie polecen, komunikatu o bledzie informujacego nieznanej funkcji `sqt` w formie

```
Undefined function or variable 'sqt'.
```

Zamiast przepisywac cala linie wciskamy klawisz kursora ↑. Wprowadzone wczesniej wyrazenie pojawi sie za znakiem zachety. Korzystajac z klawisza ← przesuwamy kursor pomiedzy litery `q` oraz `t` i wpisujemy brakujace `r`.
Wciskajac wielokrotnie klawisz ↑ mozemy przechodzic do coraz wczesniej wydawanych komend (poczynajac od najpóźniej wydanych). Jesli chcemy odnaleziona komende powtórnie wykonac wystarczy wcisnac klawisz `Enter`.
Aby wywolac ostatnio wydana komende rozpoczynajaca sie od znaku `p`, wpisujemy `p` po znaku zachety `>>`, a nastepnie wciskamy ↑. Wpisanie dowolnego ciagu znaków po znaku zachety i wciskanie ↑ prowadzi do wyswietlania wydanych komend rozpoczynajacych sie od wpisanego ciagu.
Ponizsza tabela zestawia dostepne w oknie polecen skróty klawiaturowe wraz z krótkim komentarzem na temat ich dzialania.

Skrót	Znaczenie
-	Wywoluje ostatnie w ydane polecenie (dodatkowy opis w tekscie).
–	Wywoluje nastepne polecenie. Dziala tylko wtedy, kiedy uzyto przed nim klawisza ↑.
¬	Przesuwa kursor o jeden znak do tylu
®	Przesuwa kursor o jeden znak do przodu
Ctrl + ¬	Przesuwa kursor o jedno slowo do tylu
Ctrl + ®	Przesuwa kursor o jedno slowo do przodu
Home	Przesuwa kursor na poczatek linii polecen
End	Przesuwa kursor na koniec linii polecen
Ctrl + **Home**	Przesuwa kursor na poczatek okna polecen
Ctrl + **End**	Przesuwa kursor na koniec okna polecen
Esc	Kasuje aktualna linie polecen
Delete	Kasuje znak za kursorem
Backspace	Kasuje znak przed kursorem
Shift + ®	Zaznacza znak za kursorem
Shift + ¬	Zaznacza znak przed kursorem
Shift + **Home**	Zaznacza tekst od kursora do poczatku okna polecen
Shift + **End**	Zaznacza tekst do konca okna polecen
Ctrl + **C**	Kopiuje zaznaczony fragment okna polecen do schowka systemowego
Ctrl + **V**	Wkleja tekst znajdujacy sie w schowku systemowym do okna polecen za znakiem zachety

***Cwiczenie 9.1**: Napisz w oknie komend.*

```
>> x = -1:0.1:1;
>> plot(x,sin(pi*x),'w-')
>> hold on
>> plot(x,cos(pi*x),'r-')
```

Korzystajac z podanych wyzej informacji popraw wydane powyzej komendy zgodnie z ponizszym wzorem i wykonaj je jeszcze raz.

```
>> x = -1:0.05:1;
>> plot(x,sin(2*pi*x),'w-')
>> hold on
>> plot(x,cos(2*pi*x),'r-.'), hold off
```

10. OBSZAR ZMIENNYCH

Wszystkie zmienne utworzone w Matlabie podczas wykonywania polecen uzytkownika zapisywane sa w obszarze zmiennych (ang. Workspace). Po uruchomieniu obszar zmiennych Matlaba nie zawiera zadnych danych. W obszarze zmiennych mozemy: tworzyc, przypisywac oraz kasowac zmienne, zapisywac i odczytywac dane do plików i z plików.
Usuniecie wszystkich zmiennych z obszaru zmiennych umozliwia polecenie `clear`. Polecenie `who` wyswietli liste nazw zmiennych w obszarze zmiennych.

```
>> clear  %Kasuje wszystkie zmienne
>> who    %Brak odpowiedzi – zmienne skasowano
>> a = 5; b =2; c =1;
```

```
>> who
Your variables are:
a  b  c
```

Liste zmiennych, mozemy równiez wyswietlic wydajac komende:

```
>> whos
Name      Size        Bytes  Class
a         1x1             8  double array
b         1x1             8  double array
c         1x1             8  double array
Grand total is 3 elements using 24 bytes
```

Polecenie `whos` pokazuje uszczególowiona liste zmiennych zawierajaca oprócz nazwy jej rozmiar, wielkosc zajmowanej pamieci oraz klase. Klasy zmiennych dostepnych w Matlabie to `double`, `char`, `sparse`, `struct` oraz `cell`. Klasa zmiennej okresla rodzaj informacji, która jest w niej przechowywana. Niniejsze wprowadzenie do Matlaba opiera sie na danych liczbowych, których klasa jest `double` oraz ciagach znaków, których klasa jest `char`.
Zapamietanie istniejacych w obszarze zmiennych danych nastepuje po wydaniu polecenia:

```
>> save mojezmienne1
```

Zapisuje ono aktualne wartosci wszystkich zmiennych do pliku o nazwie `mojezmienne1.mat`. Ten plik nie moze byc edytowany poza Matlabem. Odmiana tego polecenia w postaci:

```
>> save mojezmienne2 x y z
```

zapisuje do pliku `mojezmienne2.mat` wartosci zmiennych o nazwach `x`, `y`, `z`. Zaladowanie danych z pliku o znanej nazwie do obszaru zmiennych nastepuje po wydaniu polecen o skladni i dzialaniu zblizonym do polecenia save:

```
>> load mojezmienne1
>> load mojezmienne2 x y z
```

W przypadku drugiego polecenia nastapi odczytanie jedynie zmiennych o nazwach x, y, z.

11. System pomocy i dokumentacja Matlaba

Pomoc na temat wybranej funkcji dostepna jest w Matlabie po wydaniu polecenia `help`, np.:

```
>> help log
LOG    Natural logarithm.
LOG(X) is the natural logarithm of the
elements of X. Complex results are
produced if X is not positive.
See also LOG2, LOG10, EXP, LOGM.
```

Polecenie w postaci:

```
>> help help
```

Wyswietla informacje na temat sposobu korzystania z polecenia `help`. Natomiast:

```
>> help
```

Wyswietla liste kategorii, na jakie zostaly pogrupowane wszystkie informacje dostepne w systemie pomocy Matlaba.
Przeszukanie systemu pomocy Matlaba w przypadku nieznajomosci dokladnej nazwy poszukiwanej funkcji umozliwia polecenie `lookfor`, po którym podaje sie poszukiwane slowo kluczowe (w jezyku angielskim). Na przyklad aby odnalezc liste funkcji Matlaba zwiazanych kosinusem wydajemy polecenie:

```
>> lookfor cosine
ACOS   Inverse cosine.
ACOSH  Inverse hyperbolic cosine.
COS    Cosine.
COSH   Hyperbolic cosine.
```

Obok systemu dostepnego pod postacia komendy `help` istnieje pomoc kontekstowa w postaci plików html dostepnych z poziomu dowolnej przegladarki internetowej. Matlab dostarczany jest z wlasna przegladarka, której uruchomienie wraz z wczytaniem glównej strony pomocy nastepuje po wydaniu polecenia:

```
>> helpwin
```

Z pomocy kontekstowej mozemy korzystac w sposób podobny jak w przypadku polecenia `help`, wywolujac na przegladarke od razu z wlasciwa strona dokumentacji, np.:

```
>> helpwin('sqrt')
```

12. Skrytpty demonstracyjne

Matlab jest instalowany wraz z przykladowymi skryptami prezentujacymi jego mozliwosci. Dostep do skryptów demonstracyjnych uzyskujemy po wydaniu polecenia.

```
>> demo
```

13. Typy danych w Matlabie

Podstawowy typem zmiennej w Matlabie jest macierz – wielowymiarowa tabela elementów. Elementami macierzy moga byc zarówno wielkosci liczbowe jak i znaki alfanumeryczne. Jesli elementami sa wielkosci numeryczne moga one byc zarówno rzeczywiste jak i zespolone (urojone).
W Matlabie istnieja bardziej zlozone typy danych bazujace na macierzach takie jak:, struktury, macierze komórkowe oraz obiekty. Niniejsze wprowadzenie w ykorzystuje jedynie dwu- lub jedno wymiarowe macierze (wektory) wielkosci liczbowych (rzeczywistych lub zespolonych) oraz macierze lancuchów znaków alfanumerycznych.

14. Macierze i wektory

Wszystkie zmienne w Matlabie sa macierzami. Macierz $m \times n$ jest tablica liczb (rzeczywistych lub zespolonych), nazywanych elementami macierzy, ulozonych w m wierszach i n kolumnach. W zapisie matematycznym macierz obejmujemy pionowymi liniami lub nawiasami kwadratowymi. W celu zachowania zgodnosci z konwencja Matlaba wykorzystane zostana nawiasy kwadratowe. Na przyklad, dla $m = 2$, $n = 3$ otrzymujemy macierz 2×3 w ponizszej formie.

$$A = \begin{bmatrix} 5 & 7 & 9 \\ 1 & -3 & -7 \end{bmatrix}$$

Wprowadzajac macierz do obszaru zmiennych Matlaba wpisujemy ja wiersz po wierszu oddzielajac kolejne elementy w wierszu spacjami lub przecinakami, a kolejne wiersze srednikami lub znakiem nowej linii.

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

W powyzszych przykladach A oraz B sa macierzami 2×3, C jest macierza 3×2.
W Matlabie zdefiniowano równiez pod osobnymi nazwami dwa szczególne przypadki macierzy jednowymiarowych (skladajacych sie z pojedynczego wiersza lub pojedynczej kolumny). Macierz jednowierszowa to w konwencji Matlaba wektor wierszowy (poziomy), natomiast macierz jednokolumnowa to wektor kolumnowy (pionowy).

Wektory wierszowe sa lista liczb oddzielonych od siebie przecinkami lub spacjami. Kazda z liczb skladajacych sie na wektor nazywamy elementem wektora.

```
>> v = [ 1 3 sqrt(5)]
   v =
   1.0000   3.0000   2.2361
```

Zarówno w przypadku macierzy jak i wektorów polozenie spacje rozdzielajacych elementy jest niezwykle istotne.

```
>> v2 = [3+ 4 5]
   v2 =
   7     5
>> v3 = [3 +4 5]
   v3 =
   3     4     5
```

W wektorach kolumnowych elementy oddzielone sa od siebie srednikiem lub znakiem nowej linii.

```
>> c = [1; 3; sqrt(5)]
   c =
   1.0000
   3.0000
   2.2361
>> c2 = [1
   4
   5]
   c2 =
   3
   4
   5
```

14.1 Dlugosci wektorów i rozmiary macierzy

Liczbe elementów wektora nazywamy jego dlugoscia Dlugosc wektora odczytujemy korzystajac z funkcji `length`.

```
>> length(v)
   ans =
   3
```

Liczbe wierszy i kolumn macierzy nazywamy jej rozmiarem. Rozmiar macierzy odczytujemy korzystajac z funkcji `size`.

```
>> size(A), size(v)
ans =
   2    3
ans =
   3    1
```

A jest macierza 2×3. Jesli zastosujemy funkcje `size` do wektora jeden z wyswietlonych parametrów przyjmie wartosc 1 (w powyzszym przykladzie v jest wektorem wierszowym o rozmiarach 1×3). Rozmiary macierzy moga byc zapisane do zmiennych i wykorzystywane do dalszych obliczen.

```
>> s = size(A)
s =
     2     3
```

14.2 Transpozycja wektorów

Wektor wierszowy mozemy zamienic na kolumnowy (i vice versa) w procesie nazywanym transpozycja oznaczanym w symbolem apostrofu `'`.

```
>> w, w', c, c'
w =
   1    -2     3
ans =
   1
  -2
   3
c =
   1.0000
   3.0000
   2.2361
ans =
   1.0000   3.0000   2.2361
>> t = w + 2*c'
t =
   3.0000   4.0000   7.4721
>> T = 5*w' – 2*c
T =
   3.0000
 -16.0000
  10.5279
```

Jesli `x` jest wektorem skladajacym sie z elementów bedacych liczbami zespolonymi wtedy `x'` jest transpozycja wektora `x`, w którym oryginalnie wystepujace liczby zespolone zostaly zamienione na odpowiadajace im sprzezone liczby zespolone.

```
>> x = [1+3i, 2-2i]
x =
   1.0000 + 3.0000i   2.0000 – 2.0000i
>> x'
ans =
   1.0000 – 3.0000i
   2.0000 + 2.0000i
```

Elementy wektora `x` (ich czesc urojona) zostaly zdefiniowane bez wykorzystania operatora mnozenia `*`. Zapis ten zawsze oznacza definicje liczby zespolonej i dziala nawet wtedy gdy zmienna `i` nadano wartosci inna niz pierwiastek z –1. Aby uzyskac zwykla transpozycje wektora liczby zespolonych (bez zamiany liczb na liczby sprzezone) korzystamy z operatora `.'`.

```
>> x.'
ans =
   1.0000 + 3.0000i
   2.0000 - 2.0000i
```

14.3 Transpozycja macierzy

Transpozycja wektora zmienia jego postac z wierszowej na kolumnowa i odwrotnie. Rozwiniecie tej idei na macierze polega na zamianie wiersza macierzy w odpowiadajaca mu pozycja kolumne, itd.

```
>> D, D'
D =
   1    2    3    4    5
   6    7    8    9   10
  11   13   15   17   19
ans =
   1    6   11
   2    7   13
   3    8   15
   4    9   17
   5   10   19
>> size(D), size(D')
ans =
   3    5
ans =
   5    3
```

14.4 Tworzenie wektorów - notacja z dwukropkiem

Jednym z wazniejszych elementów skladni Matlaba jest dwukropek. Jest on wykorzystywany zarówno jako operator tworzenia wektorów, jak i znak specjalny upraszczajacy dostep do elementów macierzy i wektorów, co zostanie opisane w kolejny podpunktach. Wykorzystanie dwukropka w trybie operatora prowadzi do uproszczonej metody tworzenia wektorów:

```
>> 1:4
   ans =
   1    2    3    4
>> 3:7
   ans =
   3    4    5    6    7
>> 1:-1
   ans =
```

```
    []
```

Zapis w postaci `a:b:c` tworzy wektor, w którym pierwszy element przyjmuje wartosc `a`, kolejne elementy oddalone sa od siebie o wartosc `b`, ostatni element jest nie wiekszy od `c` (dlatego w ostatnim przykladzie Matlab utworzyl wektora pusty `[]`). Jesli w powyzszym zapisie pominiemy wartosc `b` kolejne elementy wektora beda róznily sie od siebie o wartosc `1`.

```
>> 0.32:0.1:0.6
   ans =
   0.3200   0.4200   0.5200
>> 5:-5:-5
   ans =
   5.0   0.0   -5.0
```

Dwukropek mozna równiez wykorzystac do tworzenia macierzy.

```
>> D = [1:5; 6:10; 11:2:20]
D =
   1    2    3    4    5
   6    7    8    9   10
  11   13   15   17   19
```

14.5 Tworzenie wektorów - linspace

Funkcja `linspace` tworzy wektor wierszowy, w którym podana liczba elementów miesci sie w podanym zakresie. Kolejne elementy wektora róznia sie od siebie o ta sama wartosc, np.:

```
>> u1 = linspace(0.0,0.25,5)
u1 =
   0    0.0625   0.1250   0.1875   0.2500
```

Pierwszy parametr funkcji `linspace` okresla wartosc pierwszego elementu wektora, drugi – ostatniego, trzeci okresla liczbe elementów. Jesli w funkcji `linspace` nie zostanie podany ostatni parametr Matlab utworzy wektor o dlugosci 100 elementów

```
>> u2 = linspace(0,9,4)'
u2 =
   0
   3
   6
   9
```

Wektor kolumnowy tworzymy korzystajac z operacji transpozycji wykonanej na funkcji `linspace`.

14.6 Tworzenie wektorów - logspace

Funkcja `logspace` tworzy wektor wierszowy, w którym podana liczba elementów miesci sie w podanym zakresie na skali logarytmicznej. Kolejne elementy wektora sa od siebie równo oddalone na skali logarytmicznej, np.:

```
>> u3 = logspace(1,4,4)
u1 =
   10    100   1000  10000
```

Pierwszy parametr funkcji `linspace` okresla wartosc wykladnika potegi liczby 10 pierwszego elementu wektora (w przykladzie 10^1), drugi – wartosc wykladnika potegi liczby 10 ostatniego elementy wektora (w przykladzie 10^4), trzeci okresla liczbe elementów. Jesli w funkcji `logspace` nie zostanie podany ostatni parametr Matlab utworzy wektor o dlugosci 100 elementów

14.7 Macierze zer i jedynek

Matlab dysponuje funkcjami tworzacymi charakterystyczne macierze o dowolnym rozmiarze. Funkcja `ones(m,n)` tworzy macierz `m` $\times$ `n`, której wszystkie elementy maja wartosc 1.

```
>> P = ones(2,3)
P =
   1    1    1
```

```
   1    1    1
```

Funkcja `zeros(m,n)` tworzy macierz `m` $\times$ `n`, której wszystkie elementy maja wartosc 0.

```
>> Z = zeros(2,3), zeros(size(P'))
Z =
   0    0    0
   0    0    0
ans =
   0    0
   0    0
   0    0
```

Drugie polecenie pokazuje w jaki sposób mozna zbudowac macierz o rozmiarze identycznym jak inna istniejaca macierza.

Macierz `n` $\times$ `n` posiada taka sama liczbe kolumn i wierszy i jest nazywana macierza kwadratowa. Macierz nazywana jest symetryczna jesli jej postac przed i po transpozycji jest identyczna (nie zmienia sie na skutek zamiany wierszy i kolumn).

```
>> S = [2 –1 0; -1 2 –1; 0 –1 2]
S =
   2   -1    0
  -1    2   -1
   0   -1    2
>> St = S'
   2   -1    0
  -1    2   -1
   0   -1    2
>> S – St
ans =
   0    0    0
   0    0    0
   0    0    0
```

14.8 Macierz jednostkowa

Jednostkowa macierz `n` $\times$ `n` jest macierza której wszystkie elementy znajdujace sie na glównej przekatnej maja wartosc 1, pozostale elementy wartosc 0. Macierz tego typu oznaczana jest litera `I`, w Matlabie tworzona jest przy pomocy funkcji `eye(n)`.

```
>> I = eye(3), x = [8; -4; 1], I*x
I =
   1    0    0
   0    1    0
   0    0    1
x =
   8
  -4
   1
ans =
   8
  -4
   1
```

Iloczyn skalarny macierzy jednostkowej i wektora nie zmienia wartosci elementów wektora (mnozenie przez macierz jednostkowa jest odpowiednikiem mnozenia przez 1 wielkosci skalarnych).

14.9 Macierz diagonalna

Macierz diagonalna jest podobna do macierzy jednostkowej z ta róznica, ze elementy znajdujace sie na glównej przekatnej nie musza przyjmowac wartosci 1.

$$D = \begin{bmatrix} -3 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

`D` jest macierza diagonalna o rozmiarach `3` $\times$ `3`. Mozna utworzyc macierz tego typu korzystajac z zapisu.

```
>> D = [-3 0 0; 0 4 0; 0 0 2]
D =
```

```
   -3     0     0
    0     4     0
    0     0     2
```

Lecz zapis ten jest niepraktyczny jesli rozmiar macierzy jest duzy (np. 100×100). Mozna zamiast tego skorzystac z funkcji `diag`. W pierwszej kolejnosci definiowany jest wektor zawierajacy kolejne elementy przekatnej macierzy, nastepnie tworzona jest macierz.

```
>> d = [-3 4 2]; D = diag(d)
D =
   -3     0     0
    0     4     0
    0     0     2
```

Funkcja `diag` umozliwia takze odczytanie elementów z glównej przekatnej dowolnej macierzy.

```
>> F = [0 1 8 7; 3 –2 –4 2; 4 2 1 1]
F =
     0     1     8     7
     3    -2    -4     2
     4     2     1     1
>> diag(F)
ans =
    0
   -2
    1
```

Macierz nie musi byc kwadratowa. Dzialanie funkcji `diag` podobnie jak wielu innych funkcji w Matlabie uzaleznione jest od polozenie funkcji w wyrazeniu matematycznym oraz podanych argumentów.

14.10. Macierz liczb pseudolosowych

Funkcja `rand(m,n)` losuje liczby z zakresu od 0 do 1 i umieszcza je w macierzy o rozmiarach `m` $\times$ `n`. Funkcja `rand` wywolana bez parametrów losuje pojedyncza liczbe.

```
>> y = rand, Y = rand(2,3)
y =
    0.9501
Y =

    0.2311    0.4860    0.7621
    0.6068    0.8913    0.4565
```

Powtórne wywolanie polecen prowadzi do innego wyniku.

14.11 Skladanie wektorów i macierzy

Wektory mozna budowac z innych wektorów:

```
>> w = [1 2 3], z = [8 9]
>> cd = [2*z, -w]
   w =
   1     2     3
   w =
   8     9
   cd =
   16     18     -1     -2     -3
```

Mozna równiez budowac o wiekszych rozmiarach macierze z kombinacji macierzy o rozmiarach mniejszych.

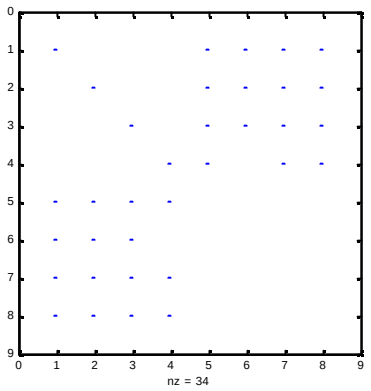
```
>> C = [0 1; 3 –2; 4 2], x=[8;-4;1]
>> G =[C x]
G =
    0     1     8
    3    -2    -4
    4     2     1
>> A, B, H = [A; B]
A =
    5     7     9
    1    -3    -7
B =
   -1     2     5
    9     0     5
H =
    5     7     9
    1    -3    -7
   -1     2     5
    9     0     5
```

W powyzszych przykladach dodano dodatkowa kolumne `x` do macierzy `C` tworzac macierz `G` oraz polaczono macierze `A` i `B` tworzac macierz `H`.

```
>> J=[1:4; 5:8; 9:12; 20 0 5 4]
J =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    20     0     5     4
>> K = [diag(1:4) J;J' zeros(4,4)]
K =
    1    0    0    0    1    2    3    4
    0    2    0    0    5    6    7    8
    0    0    3    0    9   10   11   12
    0    0    0    4   20    0    5    4
    1    5    9   20    0    0    0    0
    2    6   10    0    0    0    0    0
    3    7   11    5    0    0    0    0
    4    8   12    4    0    0    0    0
```

Polecenie `spy(K)` tworzy graficzny obraz lokalizacji niezerowych elementów macierzy `K` (wyswietla równiez liczbe niezerowych elementów macierzy `nz`).

```
>> spy(K)
```



Rys.3. Obraz lokalizacji niezerowych elementów macierzy.

14.12 Odwolania do elementów wektorów i macierzy

Kazdy element macierzy jest oznaczony indeksem, który zawiera numer wiersza i kolumny, w której znajduje sie element. Element znajdujacy sie w `i`-tym wierszu i `j`-tej kolumnie macierzy `A` jest oznaczany matematycznie $A_{i,j}$, a w Matlabie `A(i,j)`. Odwolanie wykorzystujace indeksy do dowolnego elementu macierzy `A` umieszczone po prawej stronie dowolnego wyrazenia matematycznego powoduje odczytanie wartosci tego elementu.

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> b = A(3,2)
b =
   8
>> c = A(1,1)
c =
   1
```

Jesli to samo odwolanie zostanie umieszczone po lewej stronie wyrazenia, bedzie oznaczalo przypisanie elementowi macierzy nowej wartosci.

```
>> A(1,1) = c/b
    0.2500    2.0000    3.0000
    4.0000    5.0000    6.0000
    7.0000    8.0000    9.0000
```

Odczytywanie wartosci elementów o indeksach wiekszych od rozmiarów macierzy jest niedozwolone i konczy sie komunikatem o bledzie.

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(1,4)
??? Index exceeds matrix dimensions.
```

Przypisywanie wartosci elementom o indeksach wiekszych od rozmiarów macierzy powoduje automatyczne powiekszenie macierzy.

```
>> A = [1 2 3; 4 5 6; 7 8 9]
A =
   1     2     3
   4     5     6
   7     8     9
>> A(4,4)=1
A =
   1     2     3     0
   4     5     6     0
   7     8     9     0
   0     0     0     1
```

Wykorzystanie opisywanego juz operatora dwukropka tym razem jako znaku specjalnego pozwala na odczytywanie calych kolumn lub wierszy macierzy

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(:,1)
ans =
   1
   4
   7
>> A(2,:)
ans =
   4     5     6
```

Dwukropek moze byc równiez wykorzystywany do wybierania wartosci wskazanego podzbioru kolumn i wierszy.

```
>> A(2:3,1)
ans =
   4
   7
>> A(1:2,2:3)
ans =
   2     3
   5     6
```

Ponizej przedstawiono odczytywanie wartosci elementów: z trzeciej kolumny, drugiej oraz trzeciej kolumne, czwartego wiersz oraz „srodkowych" wierszy i kolumny.

```
>> J=[1:4; 5:8; 9:12; 20 0 5 4]
J =
     1     2     3     4
     5     6     7     8
     9    10    11    12
    20     0     5     4
>> J(:,3)
ans =
     3
     7
    11
     5
>> J(:,2:3)
ans =
     2     3
     6     7
    10    11
     0     5
>> J(4,:)
ans =
     7     0     5     4
>> J(2:3,2:3)
ans =
     6     7
    10    11
```

Wszystkie opisane operacje mozna wykonac na wektorach, pamietajac, ze indeks dowolnego elementu wektora sklada sie tylko z jednej liczby (a wiec numeru kolumny w przypadku wektorów wierszowych lub wiersza w przypadku wektorów kolumnowych).

```
>> r5 = [1:2:6, -1:-2:-7]
   r5 =
   1    3    5    -1    -3    -5    -7
```

Odczytujemy elementy wektora `r5` od trzeciego do szóstego:

```
>> r5(3:6)
   ans =
   5    -1    -3    -5
```

Odczytujemy co drugi element wektora `r5` od pierwszego do ostatniego:

```
>> r5(1:2:7)
   ans =
   1    5    -3    -7
```

Dwukropek jest wykorzystywany do tworzenia wyrazen, których wynik bylby trudny do uzyskania innymi metodami.

```
>> A = ones(8,8);
>> A(3:6,3:6) = zeros(4,4)
A =
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   0   0   0   0   1   1
   1   1   0   0   0   0   1   1
   1   1   0   0   0   0   1   1
   1   1   0   0   0   0   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
```

Wreszcie dwukropek moze byc wykorzystany do zamiany dowolnego wektora lub macierzy w wektor kolumnowy.

```
>> x = 1:4
x =
   1     2     3     4
>> y=x(:)
y =
   1
   2
   3
   4
>> A = [1 2 3; 4 5 6; 7 8 9];
>> v = A(:)
v =
   1
   2
   3
   4
   5
   6
   7
   8
   9
```

14.13 Wyszukiwanie elementów wektorów i macierzy

Funkcja `find` tworzy liste indeksów elementów macierzy i wektorów spelniajacych podane jako argument funkcji warunki.:

```
>> x = -1:0.05:1;
>> y = sin(3*pi*x).*exp(-x.^2);
>> k = find(y > 0.2)
k =
  Columns 1 through 12
  9 10 11 12 13 22 23 24 25 26 27 36
  Columns 13 through 15
  37 38 39
>> km = find(x>0.5 & y<0)
km =
    32    33    34
>> A = [-2 3 4 4; 0 5 –1 6; 6 8 0 1]
A =
    -2     3     4     4
     0     5    -1     6
     6     8     0     1
>> k = find(A==0)
k =
```

```
     2
     9
```

W wyniku dzialania funkcji `find` uzyskujemy informacje, ze macierz A posiada element o wartosci 0 na pozycji 2 i 9. Trzeba jednoczesnie pamietac, ze funkcja `find` zmienia macierz A w wektor kolumnowy, co jest równoznaczne z ponumerowaniem jej elementów kolumnami, tak jak to zaprezentowano ponizej.

```
           1     4     7     10
           2     5     8     11
           3     6     9     12
>> n = find(A <= 0)
n =
     1
     2
     8
     9
>> A(n)
ans =
    -2
     0
    -1
     0
```

W przykladzie powyzej zmienna `n` zawiera liste pozycji elementów macierzy A, których wartosc jest mniejsza od 0. Zapis `A(n)` wyswietli wartosci elementów macierzy A znajdujacych sie na pozycjach zapamietanych w zmiennej `n`.

14.14 Podstawowe operacje arytmetyczne

Macierze o tych samych rozmiarach i wektory o tej samej dlugosci (zarówno kolumnowe jak i wierszowe) moga byc poddawane operacji dodawania i odejmowania oraz mnozenia przez stala. Dodawanie i odejmowanie realizowane jest element po elemencie na elementach o tych samych indeksach w obu skladnikach sumy lub róznicy. Mnozenie przez stala polega na kolejnym pomnozeniu kazdego z elementów wektora lub macierzy

a ile wynosza v i v3 i V2

```
>> v + v3
   ans =
   4.0000   7.0000   7.2361
>> v4 = 3*v3
   v4 =
   3.0000   9.0000   6.7082
>> v5 = 2*v-3*v3
   v5 =
   -7.0000  -6.0000  -10.7279
>> v + v2
   ??? Error using ==> +
   Matrix dimensions must agree.
```

Blad w ostatnim dzialaniu pojawia sie na skutek próby wykonania operacji na wektorach o róznej dlugosci.

14.15 Iloczyn skalarny wektorów (*)

Zakladajac, ze u oraz v sa wektorami o dlugosci n, u jest wektorem wierszowym, v – wektorem kolumnowym.

$$u = [u_1, u_2, ..., u_n],\ v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

Iloczyn skalarny jest wynikiem mnozenie odpowiadajacych sobie elementów obu wektorów oraz zsumowania wszystkich elementarnych iloczynów.

$$uv = \sum_{i=1}^{n} u_i v_i$$

Dla: $u = [10, -11, 12]$, $v = \begin{bmatrix} 20 \\ -21 \\ -22 \end{bmatrix}$ czyli n=3

$$uv = 10 \times 20 + (-11) \times (-21) + 12 \times (-22) = 167$$

Ten typ iloczynu realizujemy w Matlabie poleceniem:

```
>> u = [ 10, -11, 12],v = [20; -21; -22];
>> prod = u*v
```

Zamierzajac uzyskac iloczyn skalarny wektorów `u*w` oraz `v*z` definiujemy wektor wierszowyw oraz kolumnowy `z`.

```
>> w = [ 2, 1, 3], z = [7; 6; 5]
w =
   2     1     3
z =
   7
   6
   5
>> u*w
Error using ==> *
Inner matrix dimensions must agree
```

Blad dzialania operatora iloczynu skalarnego * jest spowodowany tym, ze oba wektory sa tego samego rodzaju (wierszowe). Uzyskanie iloczynu skalarnego dwóch wektorów kolumnowych lub wierszowych jest mozliwe po zastosowaniu do jednego z nich transpozycji.

```
>> u*w'
ans =
    45
>> u*u'
ans =
    365
>> v'*z
ans =
    -96
```

14.16 Iloczyn skalarny (*) macierz – wektor

Iloczyn skalarny macierzy z wektorem jest zdefiniowany wylacznie dla wektorów kolumnowych, które posiadaja liczbe elementów identyczna z liczba kolumn w macierzy. Jesli A jest macierza $m \times n$, a x jest wektorem kolumnowym o dlugosci n, wtedy istnieje iloczyn `A*x`. Wynikiem mnozenia macierzy $m \times n$ przez macierz $n \times 1$ jest macierz $m \times 1$.

Jesli macierz A przedstawimy jako m wektorów wierszowych polozonych jeden nad drugim. Iloczyn skalarny macierzy z wektorem bedzie mozna zdefiniowac jako m iloczynów skalarnych wektorów tworzacych poszczególne wiersze macierzy A z wektorem x. Wynikiem jest wektor kolumnowy o m elementach.

$$A \times x = \begin{bmatrix} [5 & 7 & 9] \\ [1 & -3 & -7] \end{bmatrix} \times \begin{bmatrix} 8 \\ -4 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \times 8 + 7 \times (-4) + 9 \times 1 \\ 1 \times 8 + (-3) \times (-4) + (-7) \times 1 \end{bmatrix} = \begin{bmatrix} 21 \\ 13 \end{bmatrix}$$

A w Matlabie.

```
>> A = [5 7 9; 1 –3 -7]
A =
   5     7     9
   1    -3    -7
>> x = [8; -4; 1]
x =
   8
  -4
   1
>> A*x
```

```
ans =
   21
   13
przy czym

>> x*A
??? Error using ==> *
Inner matrix dimensions must agree.
```

W odróżnieniu od mnozenia wielkosci skalarnych wyniki Iloczynów `A*x` oraz `x*A` sa różne.

14.17 Iloczyn skalarny (*) macierz – macierz

Przedstawmy macierz `A` jako `m` wektorów wierszowych o dlugosci `n` polozonych jeden nad drugim natomiast macierz `B` jako `p` wektorów kolumnowych o dlugosci `n` polozonych jeden za drugim.

$$A = m - wierszy\begin{Bmatrix}[\quad] \\ [\quad] \\ [\quad] \\ [\quad]\end{Bmatrix},\quad B = \underbrace{\begin{bmatrix} | & | & | & | \\ | & | & | & | \\ | & | & | & | \end{bmatrix}}_{p-kolumn}$$

Element macierzy bedacej iloczynem `A*B` polozony w `i`-tym wierszu i `j`-tej kolumnie jest iloczynem skalarnym `i`-tego wiersza macierzy `A` i `j`-tej kolumny macierzy `B`. Wynikiem iloczynu jest macierz o rozmiarze `m` × `p`.

(m ´ n) razy (n ´ p) Þ (m ´ p)

Ponizej zamieszczono przyklady iloczynów skalarnych macierzy zrealizowane w Matlabie.

```
>> A = [5 7 9; 1 –3 -7]
A =
     5     7     9
     1    -3    -7
>> B = [0 1;3 -2;4 2]
B =
     0     1
     3    -2
     4     2
>> C = A*B
C =
    57     9
   -37    -7
>> D = B*A
D =
     1    -3    -7
    13    27    41
    22    22    22
>> E = B'*A'
E =
    57   -37
     9    -7
```

Widzimy, ze E=C' co sugeruje, ze

*(A*B)'=B'*A'*

14.18 Iloczyn z kropka (.*)

Realizacja tego iloczynu wymaga wektorów tego samego rodzaju i dlugosci. Z definicji tego iloczynu, który bedziemy nazywali iloczynem z kropka, wynikiem jest wektor skladajacy sie z nastepujacych elementów:

$$u \cdot v = [u_1 v_1, u_2 v_2, \ldots, u_n v_n]$$

Wektor wynikowy jest tej samej dlugosci i tego samego rodzaju co `u` i `v`. W Matlabie iloczyn z kropka jest realizowany przy pomocy operatora `.*` co ilustruja ponizsze przyklady.

```
>> u.*w
ans =
    20   -11     36
>> u.*v'
ans =
   200   231   -264
>> v.*z, u'.*v
ans =
   140  -126   -110
ans =
   200   231   -264
```

Iloczyn z kropka dziala w przypadku macierzy identycznie jak w przypadku wektorów – element macierzy wynikowej powstaje przez mnozenie elementów o odpowiadajacych sobie pozycjach dwóch macierzy o tym samym rozmiarze.

```
>> A, B
A =
     5     7     9
     1    -3    -7
B =

    -1     2     5
     9     0     5
>> A.*B
ans =
    -5    14    45
     9     0   -35
>> A.*C
??? Error using ==> .*
Matrix dimensions must agree.
>> A.*C'
ans =

     0    21    36
     1     6   -14
```

14.19 Iloraz z kropka (./)

Nie istnieje matematyczna operacja dzielenia jednego wektora przez drugi. W Matlabie zdefiniowano operator `./`, który realizuje dzielenie elementów jednego wektora przez odpowiadajace im elementy drugiego wektora. Operator ilorazu z kropka jest dziala jedynie na wektorach tego samego rodzaju i dlugosci.

```
>> a = 1:5, b = 6:10, a./b
a =
   1     2     3     4     5
b =
   6     7     8     9    10
ans =
   0.1667  0.2857  0.3750  0.4444  0.5000
>> a./a
ans =
   1     1     1     1     1
>> c = -2:2, a./c
c =
   -2    -1     0     1     2
Warning: Divide by zero
ans =
   -0.5000 –2.0000  Inf  4.0000  2.5000
```

W ostatnim przykladzie wymagana jest realizacja dzielenia przez 0, którego wynikiem jest nieskonczonosc oznaczana w Matlabie symbolem `Inf`.

```
>> a.*b-24, ans./c
ans =
   -18    -10     0     12     26
Warning: Divide by zero
ans =
   9    10    NaN    12    13
```

W zadaniu wystepuje dzielenie 0/0, którego wynik jest nieoznaczony. Oznaczeniem dzialan o nieoznaczonym wyniku jest w Matlabie symbol `NaN`. Operator ilorazu z kropka dziala równiez w przypadku macierzy o tych samych rozmiarach.

```
>> A =[1 2 3 4; 5 6 7 8]
A =
   1     2     3     4
   5     6     7     8
```

```
>> B = [8 7 6 5;4 3 2 1]
B =
   8     7     6     5
   4     3     2     1
>> A./B
ans =
  0.1250    0.2857    0.5000    0.8000
  1.2500    2.0000    3.5000    8.0000
```

Operator ilorazu z kropka sluzy do przeprowadzenia operacji dzielenia wartosci skalarnej przez wektor.

```
>> 1/x
??? Error using ==> /
Matrix dimension must agree
>> 1./x
ans =
   10    100   1000  10000
```

Jak widac `1./x` mozna wykonac natomiast `1/x` nie funkcjonuje.

14.20 Potegowanie z kropka (.^)

Podniesienie do kwadratu elementów wektora moze byc w prosty sposób zapisane jako `u.*u`. Jednak bardziej elegancka metoda jest wykorzystanie operatora `.^`, który umozliwia stosowanie dowolnego wykladnika potegi.

```
>> u.^2
ans =
   100  121  144
>> u.*u
ans =
   100  121  144
>> u.^4
ans =
   10000   14641   20736
>> v.^2
ans =
   400
   441
   484
>> u.*w.^(-2)
ans =
   2.5000   -11.0000    1.3333
```

Przyklad dla macierzy

```
>> A =[1 2 3 4; 5 6 7 8]
A =
   1     2     3     4
   5     6     7     8
>> A.^2
ans =
   1     4     9    16
  25    36    49    64
```

Podnoszenie do potegi (z kropka w tym przypadku) jest wykonywane przed wszystkimi innymi operacjami arytmetycznymi.

15. Sumowanie elementów macierzy i wektorów

Funkcja `sum` zastosowana do wektora dodaje do siebie wszystkie jego elementy

```
>> sum(1:10)
ans =
    55
```

Zastosowana do macierzy dodaje do siebie wszystkie elementy znajdujaca sie w pojedynczej kolumnie dla kazdej kolumny osobno, co daje w wyniku wektor wierszowy. Zapis `sum(sum(A))` sumuje wszystkie elementy macierzy A.

```
>> A = [1:3; 4:6; 7:9]
A =
     1     2     3
     4     5     6
     7     8     9
>> s = sum(A)
s =
   12    15    18
>> ss = sum(sum(A))
ss =
   45
>> x = pi/4*(1:3)';
>> A = [sin(x),sin(2*x),sin(3*x)]/sqrt(2)
A =
    0.5000    0.7071    0.5000
    0.7071    0.0000   -0.7071
    0.5000   -0.7071    0.5000
>> s1 = sum(A.^2)
s1 =
    1.0000    1.0000    1.0000
>> s2 = sum(sum(A.^2))
s2 =
   3.0000
```

Suma kwadratów elementów w kazdej kolumnie macierzy A jest równa 0 a suma kwadratów wszystkich elementów macierzy A jest równa 3.

```
>> A*A'
ans =
    1.0000         0         0
         0    1.0000    0.0000
         0    0.0000    1.0000
>> A'*A
ans =
    1.0000         0         0
         0    1.0000    0.0000
         0    0.0000    1.0000
```

Co oznacza, ze wynikiem iloczynu `AA'` oraz `A'A` jest w obu przypadkach macierz jednostkowa. Macierz, która posiada powyzsza wlasnosc nazywamy ortogonalna.

16. Minimum i maksimum

Jesli `x` jest wektorem, wtedy `max(x)` zwraca element wektora o najwiekszej wartosci (`min(x)` – odpowiednio o wartosci najmniejszej).

```
>> x = [1.3 –2.4 0 2.3]
x =
    1.3000   -2.4000         0    2.3000
>> max(x)
ans =
    2.3000
>> max(abs(x))
ans =
    2.4000
>> [m,j]=max(x)
m =
    2.3000
j =
    4
```

Kiedy funkcja `max` (lub `min`) zostanie wywolana w postaci `[m,j]=max(x)` w zmiennej `m` znajdzie sie element o najwiekszej wartosci, a w `j` pozycja tego elementu.
Dla macierzy `A`, `max(A)` wyznaczy wektor wierszowy zawierajacy elementy o najwiekszych wartosciach w kazdej z kolumn macierzy. Aby znalezc element o wartosci maksymalnej w calej macierzy wymagany jest zapis `max(max(A))`.

17. Wektoryzacja obliczen

```
>> x = linspace(0,2*pi,6)';
>> y = sin(x);
>> z = cos(x);
>> [x y z]
ans =
         0         0    1.0000
    1.2566    0.9511    0.3090
    2.5133    0.5878   -0.8090
```

```
    3.7699   -0.5878   -0.8090
    5.0265   -0.9511    0.3090
    6.2832   -0.0000    1.0000
```

Wyrazenia `y=sin(x)` oraz `z=cos(x)` korzystaja z cechy charakterystycznej wiekszosci funkcji Matlaba – wektoryzacji. Jesli argumentem funkcji (w tym przypadku `sin()` oraz `cos()`) jest wektor lub macierz (w tym przypadku wektor `x`) wynikiem obliczen jest wektor lub macierz (w tym przypadku `y` oraz `z`) o tym samym rozmiarze. Wartosci elementów macierzy w wyniku obliczane sa przez zastosowanie wybranej funkcji na kazdym elemencie argumentu.

```
>> x = linspace(0,2*pi,6)';
>> [x 4*sin(3*x) 3*sin(4*x)]
ans =
         0         0         0
    1.2566   -2.3511   -2.8532
    2.5133    3.8042   -1.7634
    3.7699   -3.8042    1.7634
    5.0265    2.3511    2.8532
    6.2832   -0.0000   -0.0000
```

Wektoryzacja polega na wykorzystaniu pojedynczego wyrazenia, które operuje na wszystkich elementach macierzy bez odwolan do poszczególnych elementów na przyklad poprzez realizacje petli. Wektoryzacja umozliwia zwiezly zapis dzialan matematycznych.

```
>> x = 0:pi/4:pi;
x =
   0   0.7854   1.5708   2.3562   3.1416
y = cos(x)
>> y = cos(x)
y =
   1.0000  0.7071  0.0000 -0.7071 -1.0000
```

To samo dzialanie zapisane w jezyku programowania pozbawionym mozliwosci wektoryzacji (Fortran).

```
real x(5), y(5)
pi = 3.14159624
dx = pi/4.0
do 10 i=1,5
   x(i) = (i-1)*dx
   y(i) = cos(x(i))
10 continue
```

Ponizej przedstawiono inne przyklady zwektoryzowanych operacji matematycznych.

```
>> A = pi*[1 2; 3 4]
A =
    3.1416    6.2832
12.5664
>> S = sin(A)
S =
  1.0e-015 *
    0.1225   -0.2449
-0.4899
>> B =A/2
B =
    1.5708    3.1416
6.2832
>> T = sin(B)
>> T = sin(B)
T =
    1.0000    0.0000
   -1.0000   -0.0000
```

18. LICZBY ZESPOLONE

Matlab w identyczny sposób traktuje wartosci rzeczywiste i zespolone. Wszystkie operatory i funkcje dzialaja na obu typach wartosci (o ile wybrana operacja lub funkcja posiada dla danego typu wartosci sens matematyczny).

```
>> sqrt(-4)
ans=
   0 + 2.0000i
>> x = 1 +2i
x =
   1.0000 + 2.0000i
>> y = 1 – 2j
y =
   1.0000 – 2.0000i
>> z = x*y
z =
   5
```

Zmienne `i` oraz `j` posiadaja w Matlabie predefiniowana wartosc pierwiastka z –1.

```
>> i^2
ans =
   -1
```

Wartosc zmiennych `i` oraz `j` moze zostac zmieniona przez uzytkownika.

```
>> i =5;
>> t =8;
>> u = sqrt(i-t)
u =
   0 + 1.7321i
>> u*u
ans =
   -3.0000
>> A=[1 2;3 4];
>> i =2;
>>A(i,i) = 1
A =
   1     2
   3     1
```

Funkcja `abs` oblicza wartosc amplitudy liczby zespolonej jako pierwiastek z sumy kwadratów jej czesci rzeczywistej i zespolonej.

```
>> z = 2.500 + 4.3301i;
>> abs(z)
ans =
   5
```

Funkcja `imag` odczytuje z liczby zespolonej jej czesc urojona.

```
>> imag(z)
ans =
   4.3301i
```

Funkcja `real` odczytuje z liczby zespolonej jej czesc rzeczywista.

```
>> real(z)
ans =
   2.500
```

Funkcja `angle` oblicza kat liczby zespolonej w notacji Eulera.

```
>> angle(z)*180/pi
ans =
   60.0000
```

19. LANCUCHY ZNAKÓW

Lancuchy znaków w Matlabie sa macierzami. Do nazwy zmiennej przypisujemy lancuch znaków obejmujac go apostrofami. Poniewaz lancuch znaków jest macierza mozna odwolywac sie do jego elementów przy pomocy indeksów oraz notacji ze znakiem dwukropka.

```
>> imie = 'Jan'
>> nazwisko = 'Kowalski'
>> imie_nazwisko = [imie,' ',nazwisko]
imie_nazwisko =
   Jan Kowalski
>> lenght(imie_nazwisko)
ans =
   12
>> imie_nazwisko(9:12)
ans =
   lski
```

Funkcja `length` okresla liczbe znaków w lancuchu. Inna funkcja dzialajaca na lancuchach znaków jest `num2str`, zamieniajaca wartosc liczbowa na lancuch znaków.

```
>> komunikat1=['Jeden metr to ', ...
   num2str(100/2.54),' cali']
komunikat1 =
   Jeden metr to 39.3701 cali
>> komunikat2=['Jeden litr to ', ...
   num2str(1000/2.54^3),' cali szesciennych']
komunikat2 =
   Jeden litr to 61.0237 cali szesciennych
```

Porównanie dwóch lancuchów znaków realizuje funkcja `strcmp`.

```
>> strcmp(komunikat1,komunikat2)
ans =
   0
```

Poniewaz lancuchy sa rózne odpowiedzia jest 0, gdyby byly identyczne odpowiedzia byloby 1.Funkcja zblizona do `strcmp` jest `strncmp` porównujaca ze soba n pierwszych znaków dwóch lancuchów.

```
>> strncmp(komunikat1,komunikat2,6)
ans =
   1
```

Pierwszych szesc znaków w obu lancuchach jest identyczne. Funkcja `findstr` odszukuje pozycje pierwszego lancucha znaków w drugim lancuchu.

```
>> findstr('cali',komunikat1)
ans =
   23
```

20. WIELOMIANY

Wielomiany sa reprezentowane w Matlabie przez wektory współczynników. Kolejne elementy wektora sa współczynnikami wielomianu stojacymi przy argumencie x o coraz nizszych potegach.

$$P_n(x) = c_1 x^n + c_2 x^{n-1} + \ldots + c_n x + c_{n+1}$$

Wielomian o zapisie matematycznym

$$P_n(x) = x^3 - 2x + 12 \Rightarrow x^3 + 0x^2 - 2x + 12$$

Zapisujemy w Matlabie jako wektor.

```
>> p = [1 -2 0 12];
```

Nalezy zwrócic uwage ze elementem tego wektora sa również nieistniejace potegi x w wielomianie – stad 0 przy x^2 oraz wyraz wolny 12. Istnieje grupa funkcji, których argumentami sa wektory zawierajace współczynniki wielomianu.

```
>> x=0:0.5:2
x =
         0    0.5000    1.0000    1.5000
2.0000
>> polyval(p,x)
ans =
   12.0000   11.6250   11.0000   10.8750
12.0000
>> roots(p)
ans =
   1.8907 + 1.7781i
   1.8907 - 1.7781i
  -1.7814
```

Funkcja `polyval()` oblicza wartosc wielomianu dla podanych wartosci argumentu `x`, `roots()` oblicza wartosc pierwiastków wielomianu. Mnozenie oraz dzielenie wielomianu przez wielomian realizuja funkcje `conv()` oraz `deconv()`.

```
>> p1=[1 0]
p1 =
     1     0
>> p2=conv(p,p1)
p2 =
     1    -2     0    12     0
>> deconv(p2,p1)
ans =
     1    -2     0    12
```

21. WYKRESY FUNKCJI 2-D

Zadanie polega na wykresleniu wykresu funkcji `y=sin3πx` dla `x∈<0,1>` poprzez wyliczenie wartosci funkcji dla pewnej liczby argumentów `x` i polaczeniu współrzednych `(x,y)` odcinkami linii prostej.

```
>> x=linspace(0,1,10);
```

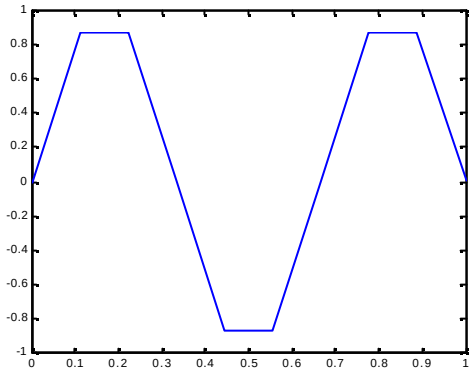
Po wykonaniu powyzszych polecen uzyskamy zbiór wartosci `x=0,1/10,2/10,...,9/10,1`. Mozemy obliczyc wartosc funkcji y przez wydanie polecenia:

```
>> y=sin(3*pi*x);
```

Tworzymy wykres bazujac na wyliczonych współrzednych:

```
>> plot(x,y)
```

Uzyskany wykres (rysunek 4) wskazuje na zbyt mala liczbe współrzednych przyblizajacych przebieg funkcji.

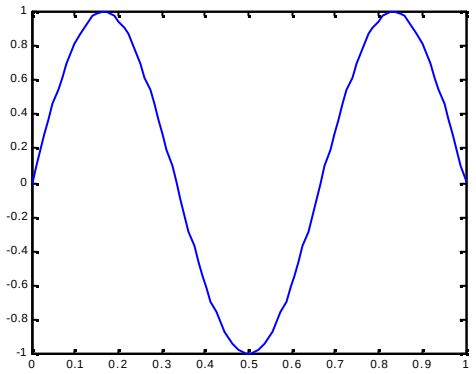


Rys.4. Wykres funkcji y=sin3?x dla x Î<0,1>, krok h=0.1.

Zmiana liczby punktów na `100` i powtórne wykonanie.

```
>> x=linspace(0,1,100);
>> y=sin(3*pi*x);
>> plot(x,y)
```

prowadzi do wykresu prezentowanego na rysunku 5.



Rys.5. Wykres funkcji y=sin3px dla x Î<0,1>, krok h=0.01.

21.1 Tytuly wykresów i opis osi

Do umieszczenia na wykresie tytulu i opisu osi wykorzystywane sa polecenia:

```
>> title('Wykres funkcji y=sin(3*pi*x)')
>> xlabel('os x')
>> ylabel('os y')
```

Opis zawarty w apostrofach moze byc dowolny.

21.2 Siatka na osiach wykresu

Siatka nakladana na uklad wspólrzednych wywolywana jest poleceniem `grid`.

```
>> grid
```

Moze byc usunieta przez powtórne wykonanie polecenia `grid` lub wywolanie `grid off`.

21.3 Wybór rodzaju i koloru linii wykresu

Domyslnie wszystkie wykresy wykonywane sa linia ciagla. Wykonanie wykresu ciagla czerwona linie realizuje polecenie:

```
>> plot(x,y,'r-')
```

Trzecim argumentem funkcji `plot` jest lancuch znaków zawarty w apostrofach, w którym pierwszy znak (opcjonalny) okresla kolor, a drugi rodzaj linii. Przykladowe znaki okreslajace kolory i rodzaje linii zebrano w tabeli

Kolory		Rodzaje linii	
y	Zólty	**.**	Punkt
m	Fioletowy	**o**	Kólko
c	Blekitny	**x**	Znak 'x'
r	Czerwony	**+**	Krzyzyk
b	Niebieski	**-**	Linia ciagla
g	Zielony	*****	Gwiazdka
w	Bialy	**s**	Prostokat
k	Czarny	**d**	Diament
		v	Trójkat (w dól)
		^	Trójkat (w góre)
		>	Trójkat (w prawo)
		<	Trójkat (w lewo)
		p	Gwiazda piecioramienna
		h	Gwiazda szescioramienna
		-	Linia ciagla
		:	Linia kropkowana
		-.	Kreska – kropka
		--	Linia przerywana

21.4. Wiele wykresów w jednym oknie graficznym

W pojedynczym ukladzie wspólrzednych mozna umiescic kilka wykresów, np.:

```
>> plot(x,y,'r-',x,cos(2*pi*x),'g--')
```

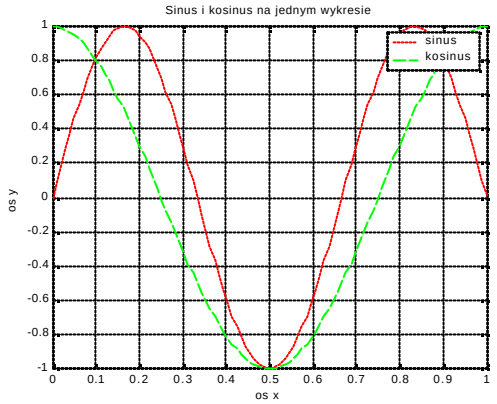
Mozna równiez dodac opisujaca wykresy legende:

```
>> legend('sinus','kosinus')
```

wyswietlajaca liste wykorzystanych w komendzie `plot` rodzajów linii z umieszczonym przy nich opisem. Matlab automatycznie okresla polozenie legendy w oknie graficznym. Na rysunku 6 pokazano rezultat wykonania polecen:

```
>> plot(x,y,'r-',x,y,cos(2*pi*x),'g--')
>> legend('sinus','kosinus')
>> title('Sinus i kosinus na jednym wykresie')
>> xlabel('os x')
>> ylabel('os y')
>> grid
```

Legenda moze zostac przesunieta przy pomocy myszki.



Rys.6. Wykres funkcji y=sin3px oraz y=cos2px dla x Î <0,1>, krok h=0.01.

21.5 Zamrazanie wykresów

Kazde wywolanie funkcji `plot` usuwa zawartosc okno graficznego przed wykonaniem kolejnego wykresu. Nie jest to pozadane jesli dodajemy kolejne wykresy do juz istniejacych. Okno graficzne przed wyczyszczeniem zabezpiecza polecenie `hold on`.

```
>> plot(x,y,'r-')
>> hold on
>> plot(x,y,'gx')
>> hold off
```

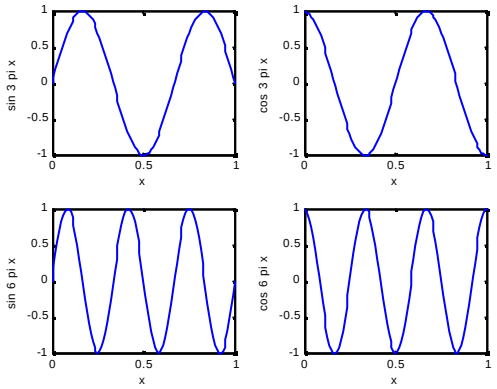
Polecenie `hold off` usuwa to zabezpieczenie (lecz nie usuwa zawartosci okna która to operacja moze byc wykonana komenda `clg`).

21.6 Podzial okna graficznego

Okno graficzne moze zostac podzielone na prostokatne obszary ulozone w tablice o `m` wierszach i `n` kolumnach. W kazdym z obszarów mozna umiescic niezalezny uklad wspólrzednych. Obszary te sa ponumerowane wierszami od `1` do `m*n` rozpoczynajac od umieszczonego w górnym lewym rogu okna graficznego. Funkcje `hold` i `grid` dzialaja zawsze w wybranym obszarze.

```
>> subplot(221),plot(x,y)
>> xlabel('x'), ylabel('sin 3 pi x')
>> subplot(222),plot(x,cos(3*pi*x))
>> xlabel('x'), ylabel('cos 3 pi x')
>> subplot(223),plot(x,sin(6*pi*x))
>> xlabel('x'), ylabel('sin 6 pi x')
>> subplot(224),plot(x,cos(6*pi*x))
>> xlabel('x'), ylabel('cos 6 pi x')
```

`subplot(221)` – lub `subplot (2,2,1)` oznacza, ze okno graficzne jest podzielone na obszary umieszczone w tablicy o dwóch wierszach i dwóch kolumnach. Ostatnia liczba wskazuje w którym obszarze tablicy bedzie tworzony wykres (rysunek 7).



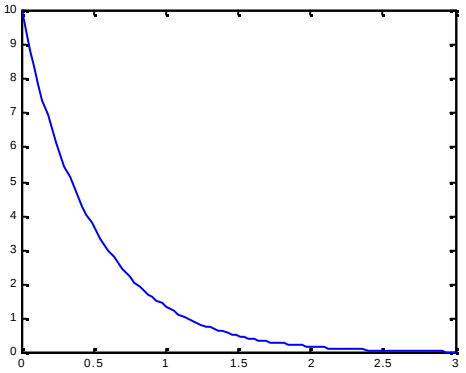
Rys.7. Podzial okna graficznego na obszary.

21.7 Modyfikacja osi wykresu

Podstawowym ukladem wspólrzednych, w którym Matlab umieszcza wykresy jest prostokatny uklad wspólrzednych. Wykresy moga byc równiez umieszczane w ukladach logarytmicznym lub semilogarytmicznym (jedna os liniowa, druga logarytmiczna). Polecenia sluzace do tworzenia wykresów tego typu zebrano w tabeli. Ich skladnia jest identyczna jak dla funkcji plot.

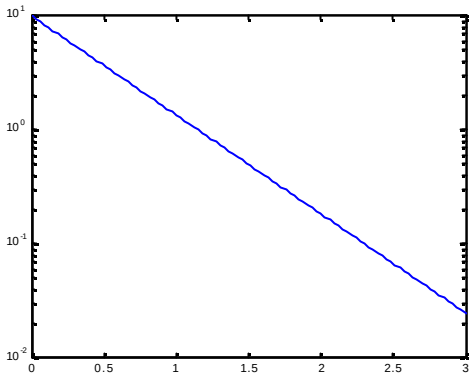
Polecenie	Typ ukladu wspólrzednych
`loglog()`	Os pozioma i pionowa logarytmiczne ($\log_{10}$)
`plot()`	Os pozioma i pionowa liniowe - rysunek 8
`semilogx()`	Os pozioma logarytmiczna ($\log_{10}$), os pionowa liniowa
`semilogy()`	Os pozioma liniowa, os pionowa logarytmiczna ($\log_{10}$) – rysunek 9

```
>> x = linspace(0,3);
>> y = 10*exp(-2*x);
>> plot(x,y)
```



*Rys.8. Wykres funkcji y=10*exp(-2*x) w prostokatnym ukladzie wspólrzednych.*

```
>> semilogy(x,y)
```

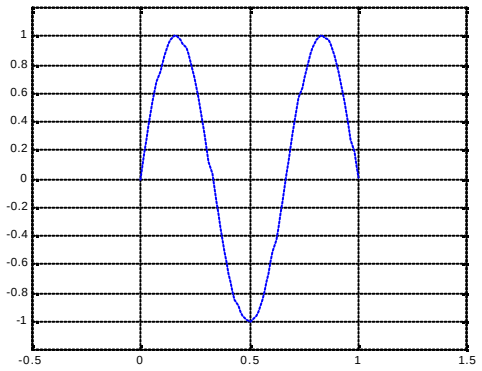


*Rys.9. Wykres funkcji y=10*exp(-2*x) w prostokatnym ukladzie wspólrzednych.*

Korzystajac z logarytmicznych ukladów wspólrzednych nalezy sprawdzic, czy wspólrzedne kreslonej funkcji nie przyjmuja wartosci ujemnych poniewaz Matlab wyswietli tylko ta czesc wykresu, która bedzie posiadala dodatnie wspólrzedne.
Niezaleznie od rodzaju ukladu wspólrzednych czesto zachodzi potrzeba zmiany zakresów osi pionowej i poziomej widocznych na rysunku.

```
>> clg, N=100; h=1/N; x=0:h:1
>> y=sin(3*pi*x); plot(x,y)
>> axis([-0.5 1.5 –1.2 1.2]), grid
```

Funkcja `axis` posiada cztery parametry, pierwsze dwa okreslaja minimalna i maksymalna wartosc pokazywana na osi poziomej, ostatnie dwa wyznaczaja minimalna i maksymalna wartosc pokazywana na osi pionowej. Parametry funkcji `axis` sa ujete w nawiasy kwadratowe. Rezultat wykorzystania funkcji `axis` przedstawia rysunek 10.

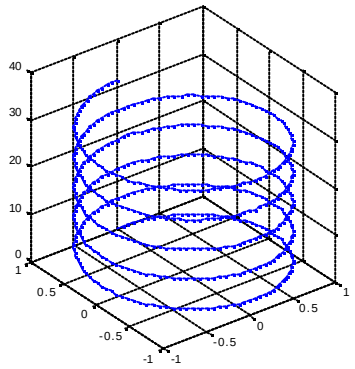


Rys.10. Efekt zmiany zakresów osi na wykresie

22. WYKRESY 3-D

Identyczna w skladni jak `plot` dla wykresów dwuwymiarowych posiada funkcja `plot3` sluzaca do wizualizacji funkcji trójwymiarowych. Jesli `x`, `y` oraz `z` sa trzema wektorami o tej samej dlugosci `plot3(x,y,z)` utworzy w trójwymiarowym prostokatnym ukladzie wspólrzednych linie, której wspólrzedne sa kolejnymi elementami wektorów `x`, `y` i `z`. Ponizszy przyklad pokazuje sposób tworzenia helisy (rysunek).

```
>> t = 0:pi/50:10*pi;
>> plot3(sin(t),cos(t),t)
>> axis square; grid on
```

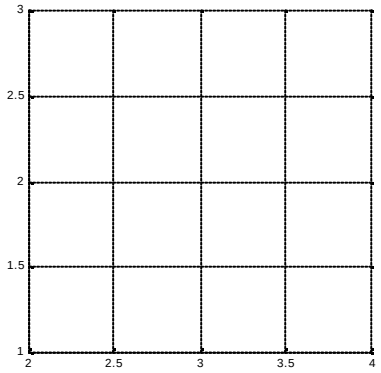


Rys.11. Wizualizacja powierzchni

W nieco inny sposób podchodzi sie w Matlabie do wizualizacji powierzchni. Powierzchnie sa funkcja f(x,y), w której w odniesieniu do kazdej pary wspólrzednych (x,y) obliczana jest jej wartosc:

$$z=f(x,y)$$

W celu wizualizacji powierzchni nalezy wybrac zakres zmiennosci x oraz y np.: $2 \le x \le 4$ oraz $1 \le y \le 3$. Te dwa przedzialy wyznaczaja kwadrat na plaszczyznie (x,y). Nastepnie na tak okreslona dziedzine funkcji nakladamy siatke, której wezly wyznacza nam punkty, dla których wyznaczymy wartosc funkcji f(x,y) (rysunek 12).



Rys.12. Przyklad siatki 2D

Wybierajac siatke o odleglosciach miedzy liniami równymi 0.5 w kazdym kierunku uzyskamy w spólrzedne jej wezlów:

```
x = 2:0.5:4; y = 1:0.5:3;
```

Powyzsza siatke budujemy funkcja meshgrid:

```
>> [X,Y] = meshgrid(2:.5:4, 1:.5:3);
>> X
X =
    2.0000   2.5000   3.0000   3.5000   4.0000
    2.0000   2.5000   3.0000   3.5000   4.0000
    2.0000   2.5000   3.0000   3.5000   4.0000
    2.0000   2.5000   3.0000   3.5000   4.0000
    2.0000   2.5000   3.0000   3.5000   4.0000
>> Y
Y =
    1.0000   1.0000   1.0000   1.0000   1.0000
    1.5000   1.5000   1.5000   1.5000   1.5000
    2.0000   2.0000   2.0000   2.0000   2.0000
    2.5000   2.5000   2.5000   2.5000   2.5000
    3.0000   3.0000   3.0000   3.0000   3.0000
```

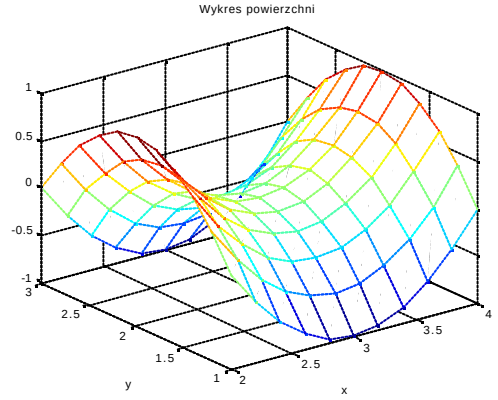
Jesli zalozymy, ze i-ty wezel liczac od lewej oraz j-ty wezel liczac od dolu siatki jest elementem macierzy X oraz Y znajdujacym sie w i-tej kolumnie oraz j-tym wierszu wtedy elementy X(i,j),Y(i,j) beda wspólrzednymi tego wezla siatki. Wyznaczenie wartosc funkcji f polega na wstawieniu w miejsce x oraz y macierzy X oraz Y.

Cwiczenie 22.1: *Wykresl powierzchnie zdefiniowana przez funkcje*

$$f(x,y)=(x-3)^2-(y-2)^2$$

dla: 2£x£4; 1£y£3

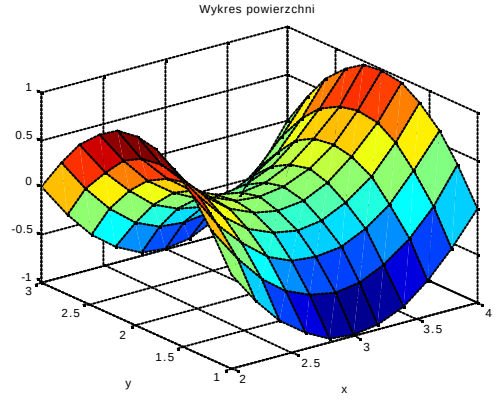
```
>> [X,Y] = meshgrid(2:.2:4, 1:.2:3);
>> Z = (X-3).^2-(Y-2).^2;
>> mesh(X,Y,Z);
>> title('Wykres powierzchni')
>> xlabel('x'), ylabel('y')
```



Rys.13. Wizualizacja powierzchni

W miejsce funkcji mesh mozemy zastosowac funkcje surf o tej samej skladni

```
>> surf(X,Y,Z);
```



Rys.14. Wizualizacja powierzchni

Cwiczenie 22.2: *Wykresl powierzchnie zdefiniowana przez funkcje*

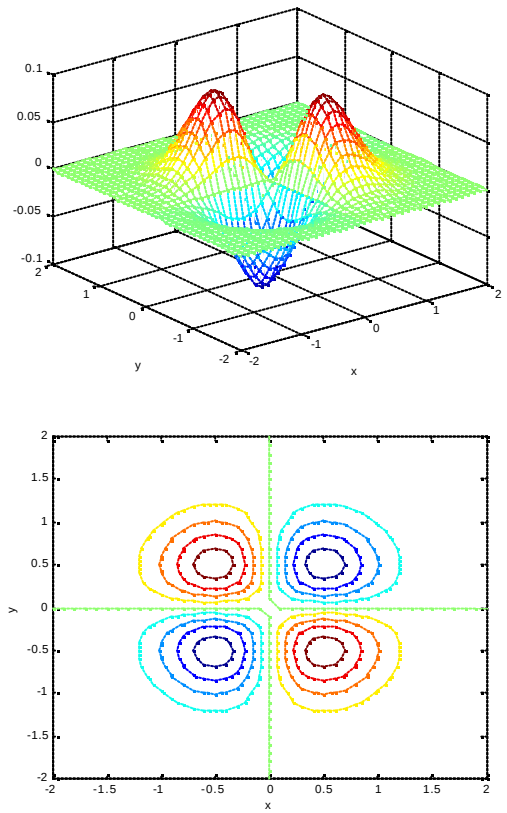
$$f(x,y)=xye^{-2\left(x^2+y^2\right)}$$

dla: -2£x£2; -2£y£2. Znajdz wartosci i polozenie maksimów i minimów funkcji.

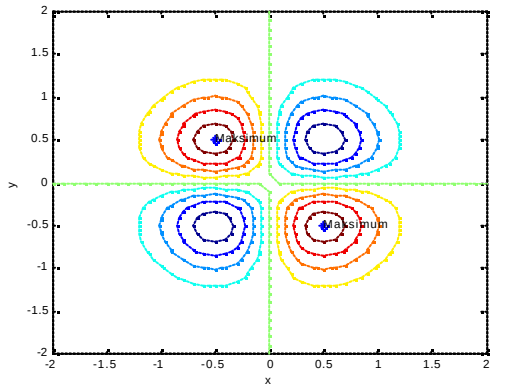
```
>> [X,Y] = meshgrid(-2:.1:2, -2:.1:2);
>> f = -X.*Y.*exp(-2*(X.^2+Y.^2));
>> mesh(X,Y,f);
>> xlabel('x'), ylabel('y'), grid
>> contour(X,Y,f)
>> xlabel('x'), ylabel('y'), grid, hold on
```

Aby odnalezc minima I maksima funkcji na siatce:

```
>> fmax = max(max(f));
fmax =
    0.0920
>> kmax = find(f==fmax)
kmax =
     641
    1041
>> plot(X(kmax), Y(kmax),'*')
>> text(X(kmax), Y(kmax),'Maksimum')
```



Rys.15. Wykresy powierzchniowy „mesh” i konturowy „contour”



Rys.16. Wykres konturowy z zaznaczonymi maksymami funkcji

23. ELEMENTY PROGRAMOWANIA

Wykorzystanie Matlaba wylacznie w trybie interaktywnym nie oddaje w pelni jego mozliwosci. Matlab to równiez jezyk programowania. Podstawowym elementem tego jezyka jest plik tekstowy o dowolnej dozwolonej w systemie operacyjnym nazwie zawierajacy blok polecen Matlaba. Nazwy plików konczy sie rozszerzeniem `.m` (np.: `wykres.m`) i z tego powodu sa one znane pod nazwa `m-plików`. Aby polecenia zawarte w pliku mogly byc wykonane musi on byc zapisany na dysku w katalogu znajdujacym sie na liscie katalogów Matlaba. Jesli plik istnieje, nie zawiera bledów lecz jest umieszczony w katalogu spoza listy nie zostanie uruchomiony poniewaz nie zostanie odnaleziony. Liste katalogów Matlaba uzyskuje sie wydajac polecenie.

```
>> path
```

Aby wyswietlic liste m-plików znajdujacych sie w biezacym katalogu nalezy wydac polecenie:

```
>> what
```

23.1 Skrypty

Skrypty sa podstawowa forma programowania w Matlabie. Sluza glównie do automatyzacji prostych zadan. Poniewaz uzytkownik uruchamiajac skrypt nie moze podawac swoich parametrów jak równiez wynik dzialania skryptu nie moze zostac przypisany zmiennej ich funkcjonalnosc jest ograniczona (np. wyliczaja wartosc funkcji wylacznie dla danego z góry I zapisanego w skrypcie zbioru argumentów). Mozliwosci jezyka programowania Matlaba wykorzystuja dopiero funkcje, których opis znajduje sie w dalszej czesci tego wprowadzenia.
Skrypty sa uruchamiane przez podanie nazwy pliku. Wywolujac skrypt nie wpisuje sie rozszerzenia `.m`. W trakcie wykonywania skryptu w oknie komend lub w oknie graficznym pokazywane sa jedynie wyniku dzialania komend zawartych w skrypcie (pod warunkiem, ze nie jest uzywany znak srednika), natomiast same komendy nie sa wyswietlane.

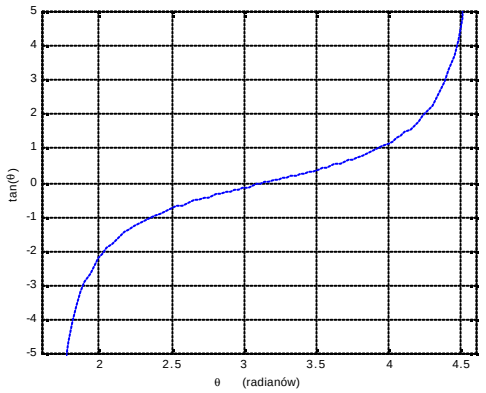
Przyklad 23.1: *Utwórzyc skrypt tworzacy wykres z opisanymi osiami funkcji y=tan(q) dla 1.6£q£4.6 (wartosci q w radianach.*

Tworzenie skryptu rozpoczyna uruchomienie edytora m-plików w Matlabie komenda **New->M-File** z menu **File** okna komend. W oknie edytora nalezy wpisac polecenia Matlaba, które realizuja zadanie zdefiniowane w przykladzie.

```
theta = linspace(1.6,4.6);
tandata = tan(theta);
plot(theta,tandata)
xlabel('\theta     (radianów)')
ylabel('tan(\theta)')
grid
axis([min(theta) max(theta) -5 5]);
```

Polecenia zapisujemy na dysku pod nazwa `wykres.m` poleceniem **Save As** z menu **File**. Uruchomienie skryptu nastepuje przez wpisanie w oknie komend polecenia.

```
>> wykres
```

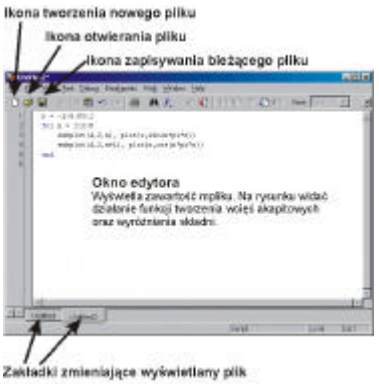


Rys.17. Wynik dzialania skryptu wykres.m

Jakakolwiek zmiana w powyzszym wykresie wymaga wprowadzenia zmian do skryptu i jego ponownego uruchomienia.
Wszystkie zmienne wykorzystywane przez skrypt sa zapisywane do obszaru zmiennych Matlaba (lub z niego odczytywane). Moze to powodowac powstawanie bledów w dzialaniu skryptu Bledy te to na przyklad, zmianie przez skrypt wartosci zmiennych istniejacych w obszarze zmiennych Matlaba, a potrzebnych do innych niz realizowane przez skrypt obliczen. Moze to byc takze zmiana w trakcie obliczen realizowanych przed uruchomieniem skryptu wartosci zmiennych potrzebnych do jego poprawnego dzialania.

23.2 Edytor m-plików

Dostepny w Matlabie edytor stanowi graficzne narzedzie umozliwiajace tworzenie oraz uruchamiania m-plików.



Rys.18. Okno edytora m-plików

Utworzenie nowego m-pliku w Matlabie polega wybraniu komendy **File® New® M-file** z menu w oknie komend (lub wybraniu odpowiedniej ikony, jak na rysunku 18). W oknie edytora zostanie utworzony pusty plik o nazwie `Untitled`. W edytorze mozna tworzyc lub otwierac jednoczesnie w iele m-plików. Aby utworzyc kolejny m-plik nalezy wybrac komende **File® New® M-file** tym razem z menu edytora. Aby otworzyc istniejacy plik nalezy wybrac komende **File® Open** z menu okna edytora (lub wskazac odpowiednia ikone, jak na rysunku 18). Kolejne, tworzone lub otwierane, m-pliki pojawiaja sie jako kolejna zakladka na pasku statusu okna edytora. Wybór m-pliku do edycji nastepuje po wskazaniu zakladki z jego nazwa. Wprowadzenie poprawek do m-pliku sygnalizuje pojawienie sie znaku gwiazdki * obok jego nazwy na listwie tytulowej edytora. Aby zapisac m-plik nalezy skorzystac z jednej z komend **Save** w menu **File. Save** – zapisuje plik pod nadana mu wczesniej nazwa. Dla nowo utworzonego pliku zostanie wyswietlone okno dialogowe pozwalajace nadac mu nazwe oraz okreslic katalog, w którym zostanie zapisany. **Save As** – zapisuje plik pod podana w oknie dialogowym nazwa i we wskazanym katalogu. **Save All** – zapisuje wszystkie otwarte m-pliki pod nadanymi wczesniej nazwami. Dla kazdego nowo utworzonego pliku otwiera okno w którym nastepuje podanie nazwy oraz wskazanie katalogu docelowego.
Podstawowa edycje tekstu ulatwiaja skróty klawiaturowe dostepne w edytorze. Ponizej zamieszczono tabele zawierajaca ich skrócona liste

Skrót	Znaczenie
-	Przesuwa kursor o jedna linie do góry
¯	Przesuwa kursor o jedna linie w dól
¬	Przesuwa kursor o jeden znak do tylu
®	Przesuwa kursor o jeden znak do przodu
Ctrl + ¬	Przesuwa kursor o jedno slowo do tylu
Ctrl + ®	Przesuwa kursor o jedno slowo do przodu
Home	Przesuwa kursor na poczatek linii
End	Przesuwa kursor na koniec linii
Ctrl + **Home**	Przesuwa kursor na poczatek pliku
Ctrl + **End**	Przesuwa kursor na koniec pliku
Del	Kasuje znak za kursorem
Ctrl + Del	Kasuje znaki od kursora do konca linii
Backspace	Kasuje znak przed kursorem
Shift + ®	Zaznacza znak za kursorem
Shift + ¬	Zaznacza znak przed kursorem
Shift + **Home**	Zaznacza tekst od kursora do poczatku linii
Shift + End	Zaznacza tekst do konca linii

Zaawansowane funkcje edytora ulatwiaja tworzenie m-plików zawierajacych polecenia Matlaba. Do funkcji tych naleza: wyróznianie skladni – automatyczne oznaczanie kolorem fragmentów kodu np.: funkcji (kolor niebieski), lancuchów znaków (kolor czerwony), komentarzy (kolor zielony), automatyczne wciecia akapitowe – tworzace wciecia akapitowych ulatwiajace analize takich konstrukcji skladniowych Matlaba jak petle lub instrukcje warunkowe, pomoc kontekstowa – umozliwiajaca uzyskanie pomocy na temat zaznaczonego fragmentu kodu po wybraniu komendy **Help on Selection** z menu kontekstowego otwieranego prawym klawiszem myszki.

23.3 Wyrazenia logiczne

Tworzenie wyrazen logicznych w Matlabie umozliwiaja dwa rodzaje operatorów: operatory relacji i operatory logiczne.
Operatory relacji umozliwiaja porównanie dwóch wartosci (lub zmiennych). Wynikiem operacji porównania wartosci jest wartosc logiczna prawda (ang. true) lub falsz (ang. false). Matlab przedstawia wartosci logiczne jako liczby calkowite 1 i 0.

```
true = 1, false = 0
```

Jesli zmiennej `x` podczas w dowolnej chwili zostanie przypisana wartosc, mozna ja porównywac z inna wartoscia przez zastosowanie operatorów relacji, np.:

- `x == 2` – Czy x jest równe 2 ?
- `x ~= 2` – Czy x jest rózne od 2?
- `x > 2` – Czy x jest wieksze od 2?
- `x < 2` – Czy x jest mniejsze od 2?
- `x >= 2` – Czy x jest wieksze lub równe 2?
- `x <= 2` – Czy x jest mniejsze lub równe 2?

Szczególna uwaga powinna byc poswiecona testowi równosci, który korzysta z dwóch znaków „==”. Pojedynczy znak oznaczalby przypisanie zmiennej x wartosci 2.

```
>> x = pi
x =
   3.1416
>> x ~= 3, x ~= pi
ans =
   1
ans =
   0
```

Jesli `x` jest wektorem lub macierza powyzsze testy wykonywane sa element po elemencie.

```
>> x = [-2 pi 5; -1 0 1]
x =
   -2.0000    3.1416    5.0000
   -1.0000         0    1.0000
>> x == 0
ans =
   0     0     0
   0     1     0
x > 1, x >= 1
ans =
   0     1     1
   0     0     0
ans =
   0     1     1
   1     1     1
>> y = x>=1, x > y
y =
   0     1     1
   1     1     1
ans =
   0     1     1
   0     0     0
```

Wartosci logiczne bedace wynikiem operacji porównania mozna laczyc ze soba w wyrazenia logiczne korzystajac z operatorów logicznych.

```
>> x = [-2 pi 5; -1 0 1]
```

```
x =
   -2.0000    3.1416    5.0000
   -1.0000         0    1.0000
>> x > 3 & x < 4
ans =
   0     1     0
   0     0     0
>> x > 3 | x == -3
ans =
   0     1     1
   0     1     0
```

Operator & oznacza iloczyn logiczny (funkcja „i") natomiast | oznacza sume logiczna (funkcja „lub") operator ~ oznacza zaprzeczenie (funkcja „nie") stad konstrukcja operatora ~= oznaczajacego „nie równy". Z zaprzeczenia mozna równiez korzystac w odniesieniu do calych wyrazen np.: `~(x>0)`, etc.

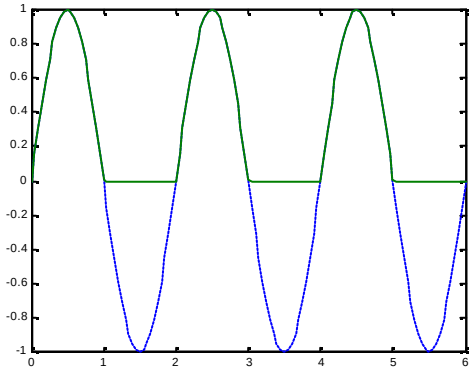
```
>> x > 3 | x == -3 | x <= -5
ans =
   0     1     1
   1     1     0
```

Jednym z zastosowan wyrazen logicznych jest maskowanie (ukrywanie) niektórych z elementów macierzy.

```
>> x, L = (x>=0)
x =
   -2.0000    3.1416    5.0000
   -1.0000         0    1.0000
L =
   0     1     1
   0     1     1
>> pos = x.*L
pos =
   0    3.1416    5.0000
   0         0         0
```

Macierz `pos` zawiera dodatnie elementy macierzy `x`. Inny przyklad maskowania elementów.

```
>> x = 0:0.05:6; y = sin(pi*x); Y = (y>=0).*y;
>> plot(x,y,':',x,Y,'-')
```



*Rys.19. Wykres funkcji sin(pi*x) oraz sin(pi*x) dla dodatnich wartosci funkcji wykonany metoda maskowania elementów wektorów i macierzy*

23.4 Petle

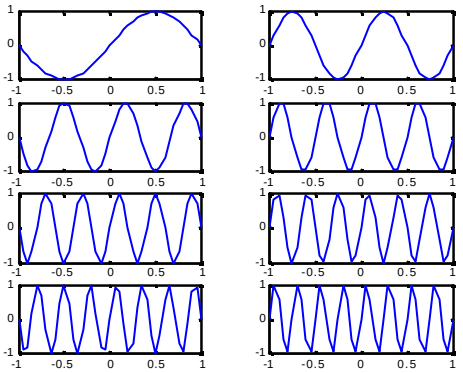
Istnieja sytuacje kiedy zachodzi potrzeba wielokrotnego powtarzania polecenia lub grupy polecen w Matlabie. Ze wzgledu na wektoryzacje funkcji oraz operator dwukropka sytuacja taka wystepuje rzadziej niz w innych jezykach programowania.

Przyklad 23.2: *Narysowac funkcje* $y=\sin(npx)$ *w przedziale* $-1 \pounds x \pounds 1$ *dla* $n = 1,2,...,8$.

Mozna to wykonac kreslac funkcja `plot` osiem niezaleznych wykresów lecz latwiej jest skorzystac z petli.

```
>> x =-1:0.05:1;
>> for n =1:8
     subplot(4,2,n), plot(x,sin(n*pi*x))
   end
```

Wszystkie polecenia pomiedzy liniami zaczynajacymi sie od `for` oraz `end` sa powtarzane dla kolejnych wartosci `n` najpierw 1 potem 2, i tak dalej az do `n = 8`. Funkcja `subplot` buduje tabele 4 x 2 obszarów w oknie graficznym oraz w `n-tym` przejsciu petli umieszcza wykres w `n-tym` obszarze okna graficznego.

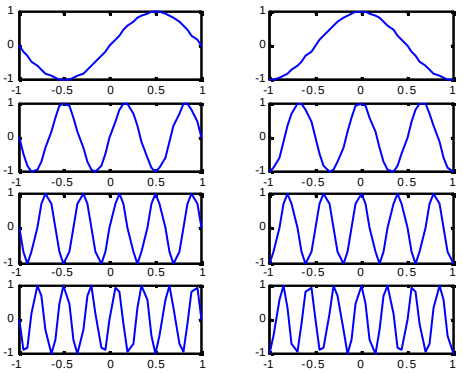


*Rys.20. Wykres funkcji sin(n*p*x) dla n = 1,2,...,8.*

Podobne zadanie realizuje ponizsza grupa polecen umieszczona w petli.

```
>> x = -1:0.05:1
>> for n = 1:2:8
    subplot(4,2,n), plot(x,sin(n*pi*x))
    subplot(4,2,n+1), plot(x,cos(n*pi*x))
   end
```

która umieszcza obok siebie wykresy funkcji $\sin(npx)$ oraz $\cos(npx)$ dla $n = 1, 3, 5, 7$.



Rys.21. Wykres funkcji sin(npx) oraz cos(npx) dla n = 1,3,5,7.

Jako „licznik petli" (n w powyzszych przykladach) moze byc wykorzystana zmienna o dowolnej nazwie. W odróznieniu od innych jezyków programowania, w których licznik petli podaza od jednej wartosci do drugiej z podanym krokiem w Matlabie kolejnymi wartosciami licznika sa kolejne elementy wektora zdefiniowanego w zmiennej po poleceniu `for`. Oprócz odtwarzania dzialania petli `for` z innych jezyków programowania (tak jak w przykladach powyzej) mozliwa jest równiez konstrukcja.

```
>> for licznik = [23 11 19 5.4 6]
     ...
   end
```

Polecenia umieszczone pomiedzy `for` i `end` powtarzane sa tyle razy ile elementów zawiera wektor `licznik` dla kolejnych wartosci licznika przyjmujacego najpierw wartosc 23 potem 11 potem 19 itd.

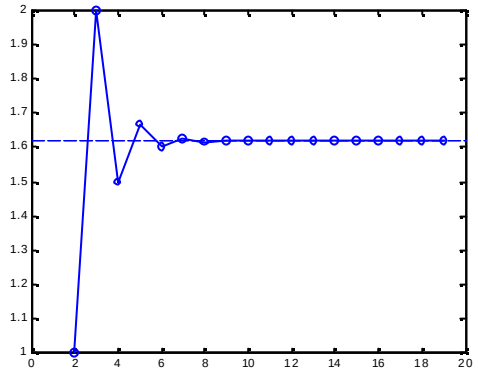
Przyklad 23.3: *Ciag Fibbonaciego rozpoczyna sie od liczb 0 oraz 1, a kolejne wyrazy tego ciagu sa suma dwóch poprzednich.*

$$f_1 = 0$$
$$f_2 = 1$$
$$f_n = f_{n-1} + f_{n-2} \; dla \; n = 3,4,5,...$$

Nalezy udowodnic twierdzenie, ze stosunek f_n/f_{n+1} dwóch kolejnych elementów ciagu zmierza do zlotego podzialu

$$\frac{\sqrt{5}+1}{2} = 1.6180..$$

```
>> f(1) = 0; f(2) = 1;
>> for i = 3:20
     f(i) = f(i-1) + f(i–2);
   end
>> plot(1:19, f(2:20)./f(1:19),'o')
>> hold on
>> plot(1:19, f(2:20)./f(1:19),'-')
>> plot([0 20], (sqrt(5)+1)/2*[1 1],'--')
```



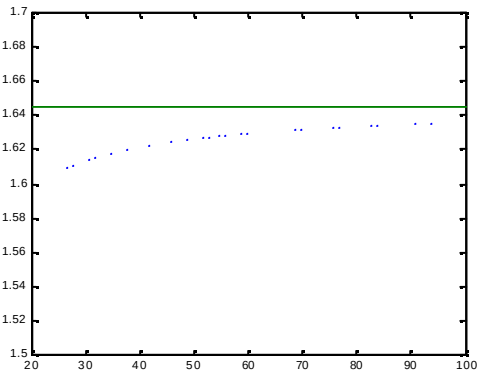
Rys.22. Wykres kolejnych wyrazów ciagu Fibonnaciego.

Przyklad 23.4: *Utworzyc liste wyrazów ciagu:*

$$S_{20} = 1 + 1/22 + 1/32 + ... + 1/202$$
$$S_{21} = 1 + 1/22 + 1/32 + ... + 1/202 + 1/212$$
$$S_{100} = 1 + 1/22 + 1/32 + ... + 1/202 + 1/212 + ... + 1/1002$$

Dla powyzej zdefiniowanego ciagu istnieje 81 wyrazów bedacych sumami. Pierwszy z wyrazów moze byc obliczony jako `sum(1./(1:20).^2)`. Kolejne wyrazy mozna wyznaczyc przy pomocy ponizszego kodu.

```
>> S = zeros(100,1);
>> S(20)=sum(1./(1:20).^2);
>> for n = 21:100
     S(n) = S(n-1) + 1/n^2;
   end
>> clf;
>> plot(1:100,S,'.',[20 100],[1 1]*pi^2/6,'-')
>> axis([20 100 1.5 1.7])
>> [(98:100)' S(98:100)]
ans =
   98.0000    1.6348
   99.0000    1.6349
  100.0000    1.6350
```



Rys.23. Wykres wyrazów ciagu S.

Wektor S zawiera poszczególne wyrazy ciagu. Pierwszy wyraz jest obliczany bezposrednio funkcja `sum`, kolejne wyrazy obliczane sa przez dodanie do poprzedniego skladnika $1/n^2$. Ostatnie polecenia wyswietla wartosci trzech ostatnich wyrazów, aby pokazac, ze stopniowo osiagana jest granica tego ciagu ($p^2/6 = 1.64493...$).

Oprócz petli `for...end` realizujacej zawarte w niej polecenia scisle okreslona liczbe razy wystepuja sytuacje, w których konieczne jest powtarzanie polecen w petli, az do spelnienia okreslonych warunków zapisanych w postaci wyrazenia logicznego (bez znajomosci koniecznej liczby powtórzen jak ma to miejsce w petli `for`...). Petle tego rodzaju realizuje konstrukcja `while...end`.

Przyklad 23.5: *Jaka jest najwieksza wartosc n, która mozna przyjac w sumie*

$$1^2 + 2^2 + ... + n^2$$

aby jej wartosc byla mniejsza od 100?

```
>> S = 1; n = 1;
>> while S + (n+1)^2 < 100
     n = n + 1; S = S + n^2;
   end
>> [n S]
ans =
     6    91
```

Przyklad 23.6: *Odnalezc przyblizona wartosc pierwiastka równania $x = cos(x)$. Mozna to zrealizowac przyjmujac na przyklad $x_1=p/4$, a nastepnie obliczajac ciag wartosci:*

$$x_n = cos(x_{n-1}) \; dla \; n = 2,3,4,...$$

Do momentu, w którym róznica pomiedzy dwoma kolejnymi elementami ciagu $|x_n - x_{n-1}|$ bedzie wystarczajaca mala.

Metoda 1

```
>> x = zeros(1,20); x(1) = pi/4;
>> n = 1; d = 1;
>> while d > 0.001
     n = n+1; x(n) = cos(x(n-1));
     d = abs(x(n) - x(n-1));
   end
>> n,x
n =
    14
x =
Columns 1 through 7
0.7854 0.7071 0.7602 0.7247 0.7487 0.7326 0.7435
Columns 8 through 14
0.7361 0.7411 0.7377 0.7400 0.7385 0.7395 0.7388
Columns 15 through 20
0      0      0      0      0      0
```

Wektor `x` przechowuje wszystkie uzyskane rezultaty. Poniewaz nie jest znana liczby powtórzen petli nie mozna okreslic dlugosci wektora `x`. Ponadto rzadko kiedy interesujace sa wynik posrednie, korzysta sie raczej z wyniku koncowego. Problemem jest równiez to, ze moze wystapic sytuacja, w której warunek `d<=0.001` nigdy nie bedzie spelniony, petla nie zostanie zakonczona, a tym samym nie zostana zakonczone obliczenia. Konieczne jest zatem zastosowanie warunku ograniczajacego liczbe realizowanych powtórzen w petli `while`.

Metoda 2

```
>> starex = pi/4; n = 1; d = 1;
>> while d > 0.001 & n < 20
     n = n + 1; nowex = cos(starex);
     d = abs(nowex - starex);
     starex = nowex;
   end
>> [n, nowex, d]
ans =
   14.0000    0.7388    0.0007
```

Petla `while` jest wykonywana dopóki `d>0.001` i `n<20`. Petla `while` posiada nastepujaca budowe:

```
while wyrazenie_logiczny
   Polecenia, które sa realizowane dopóki
   wyrazenie_logiczne jest spelnione.
end
```

23.5 Instrukcje warunkowe

Instrukcje warunkowe decyduja o wykonaniu lub ominieciu bloku polecen na podstawie wartosci wyrazenia logicznego. Jedna z nich jest `if`. Aby sprawdzic, czy π^e jest mniejsze lub równe e^p stosujemy polecenie.

```
>> a = pi^exp(1); c = exp(pi);
>> if a>=c
     b = sqrt(a^2 - c^2)
   end
```

Zmiennej `b` przypisywana jest wartosc `sqrt(a^2-c^2)` tylko wtedy gdy $p^e >= e^p$. Jesli ten warunek nie jest spelniony zmienna `b` nie jest tworzona. W bardziej ogólnym przypadku zmienna `b` jest zawsze tworzona.

```
>> if a >= c
     b = sqrt(a^2 – c^2)
   else
     b = 0
   end
b =
   0
```

W poleceniu tym zmiennej `b` przypisywana jest wartosc `sqrt(a^2–c^2)` jesli spelniony jest warunek `a>=c`. W przypadku niespelnienia warunku `b=0`. Najbardziej rozbudowana postac polecenia `if` prezentuje przyklad.

```
>> if a >= c
     b = sqrt(a^2 – c^2)
   elseif a^c > c^a
     b = c^a/a^c
   else
     b = a^c/c^a
   end
b =
    0.2347
```

Instrukcje warunkowa if charakteryzuje ponizszy schemat dzialania.

```
If wtrazenie_logiczne_1
   Polecenia wykonywane gdy
   wyrazenie_logiczne_1 jest spelnione
elseif wyrazenie_logiczne_2
   Polecenia wykonywane gdy
   wyrazenie_logiczne_2 jest spelnione oraz
   wyrazenie_logiczne_1 jest niespelnione
.
.
.
elseif wyrazenie_logiczne_n
   Polecenia wykonywane gdy
   wyrazenie_logiczne_n jest spelnione oraz
   wyrazenia od 1 do n-1 sa niespelnione.
else
   Polecenia wykonywane, gdy wszystkie powyzsze
   wyrazenia logiczne sa niespelnione
end
```

23.6 Funkcje

Elementem jezyka programowania w Matlabie sa funkcje. Nazwe swoja zaczerpnely od funkcji matematycznych. Ich zadaniem jest powiazanie zaleznoscia wielkosci danych i szukanych. Funkcje wykonuja dzialania na wielkosciach przekazywanych poprzez nazwy zmiennych wejsciowych i wyjsciowych. Funkcja Matlaba podobnie jak skrypt ma postac m-pliku, lecz o szczególnej budowie.

***Przyklad 26.1**: Napisac funkcje obliczajaca wartosc sumy kwadratów oraz szescianów dwóch liczb.*

Korzystajac z edytora m-plików wprowadzono

```
function [k, s]=sum23(x, y)
% sum23  Oblicza wartosc sumy kwadratów k
%        i szescianów s liczb x I y.
k = x.^2 + y.^2;
s = x.^3 + y.^3;
```

Elementem odrózniajacym funkcje od skryptów jest definicja funkcji. Zawiera ja pierwsza linia m-pliku.

```
function [k,s]=sum23(x,y)
```

Definicje funkcji rozpoczyna slowo `function` po którym wystepuja kolejno:

- lista nazw zmiennych wyjsciowych (szukanych), której elementy sa oddzielone od siebie przecinkami i objete nawiasami kwadratowymi – `[k,s]`;
- nazwa funkcji umieszczona po znaku równosci – `sum23` – która jest równiez nazwa m-pliku funkcji;
- lista nazw zmiennych wejsciowych (danych), której elementy sa oddzielone od siebie przecinkami i objete nawiasami okraglymi `(x,y)`.

Utworzona w edytorze funkcje nalezy zapisac na dysku pod nazwa `sum23.m`. Nazwy zmiennych wejsciowych `x`, `y` oraz wyjsciowych `k`, `s` sa tzw. parametrami formalnymi funkcji `sum23`. Obliczenia wykonywane wewnatrz funkcji realizowane sa na parametrach formalnych. Uruchomienie funkcji moze nastapic z poziomu okna komend Matlaba

```
>> a = [1 2 3]';
>> b = [3 2 1]';
>> [kwadrat, szescian] = sum23(a,b)
kwadrat =
    10
     8
    10
szescian =
    28
    16
    28
```

Wywolanie to pokazuje istote parametrów formalnych. Uzytkownik definiuje zmienne o nazwach `a` i `b` zawierajace wartosci dla których ma zostac obliczona wartosc funkcji `sum23`. Wywolanie `sum23(a,b)` informuje, ze parametrom formalnym `x` i `y` wystepujacym w definicji funkcji zostaly przypisane wartosci zmiennych `a` i `b`. Zapis `[kwadrat, suma]` przed znakiem równosci informuje, ze wynik dzialania funkcji, zapisany w parametrach formalnych `k` oraz `s` ma zostac przepisany do zmiennych `kwadrat` i `szescian`

Wewnatrz m-pliku funkcji mozna równiez korzystac ze zmiennych pomocniczych nie bedacych parametrami formalnymi.

***Przyklad 26.2:** Pole trójkata A o bokach o dlugosciach a, b oraz c jest dane wzorem.*

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

gdzie $s=(a+b+c)/2$. Nalezy utworzyc funkcje Matlaba, która bedzie przyjmowala na wejsciu dlugosci boków a, b, c, obliczala pole jego powierzchni A.

Rozwiazaniem tego zadania jest m-plik o nazwie `powierzchnia.m` zawierajacy funkcje `powierzchnia` oraz zmienna pomocnicza `s`..

```
function [A] = powierzchnia(a,b,c)
% Oblicza pole powierzchni trójkata A,
% którego boki posiadaja dlugosc a,b,c
s = (a + b + c)/2;
A = sqrt(s*(s-a)*(s-b)*(s-c));
```

Cecha parametrów formalnych oraz zmiennych definiowanych wewnatrz m-plików funkcji jest to, ze powstaja w trakcie wywolania funkcji i sa usuwane z pamieci z chwila zakonczenia jej dzialania. Co wiecej jezeli w obszarze zmiennych Matlaba istnieja zmienne o nazwach identycznych jak nazwy zmiennych

wykorzystywanych wewnatrz m-pliku funkcji to moga one koegzystowac nie wplywajac wzajemnie na swoja wartosc

```
>> s = 2+15i
s =
   2.0000 + 15.0000i
>> pole_tr=powierzchnia(10,15,20)
pole_tr =
   72.6184
>> s
s =
   2.0000 +15.0000i
```

W powyzszym zapisie zdefiniowano w obszarze zmiennych Matlaba zmienna `s`. W funkcji `powierzchnia` wykorzystywana jest równiez zmienna pomocnicza `s`. Po wywolaniu funkcji `powierzchnia` wartosc zmiennej `s` w obszarze zmiennych nie zmienila sie.
Parametry formalne pozwalaja na zastosowanie identycznego ciagu dzialan dla róznych danych. Obliczenie pola trójkata dla innych dlugosci boków wymaga wylacznie zmiany wartosci wejsciowych w wywolywanej funkcji, a nie napisania nowego m-pliku jak to mialo miejsce w przypadku skryptów.

```
>> pole_tr=powierzchnia(2,3,4)
pole_tr =
    2.9047
```

Istotnym lecz niedocenianym elementem m-pliku funkcji sa komentarze znajdujace ponizej linii z definicja funkcji. Korzysta z nich polecenie `help` wyswietlajace informacje na temat podanego jako parametr polecenia.

```
>> help powierzchnia
```

Spowoduje wyswietlenie zawartego w naglówku funkcji komentarza

```
Oblicza pole powierzchni trójkata A,
którego boki posiadaja dlugosc a,b,c
```

Czesto stosowana metoda rozwiazywania problemów jest dzielenie ich na zadanie reprezentowane w postaci m-plików funkcji. Pelne wykorzystanie tego podejscia daje inna cecha Matlaba jaka jest mozliwosc wywolywania funkcji z wnetrza innych funkcji. Na przyklad, funkcja `sum23`, moze zostac zastapiona trzema m-plikami

sumaks.m

```
function [k, s]=sumaks(x, y)
% sum23  Oblicza wartosc sumy kwadratów k
%        i szescianów s liczb x I y.
k = suma2(x,y);
s = suma3(x,y);
```

suma2.m

```
function [k]=suma2(x, y)
% sum23  Oblicza wartosc sumy
%        kwadratów k liczb x I y.
k = x.^2 + y.^2;
```

suma3.m

```
function [s]=suma3(x, y)
% sum23  Oblicza wartosc sumy
%        szescianów s liczb x I y.
s = x.^3 + y.^3;
```

Funkcja `sumaks` realizuje to samo dzialanie co funkcja `sum23`, lecz w tym celu wywoluje dwie dodatkowe funkcje `suma2` – obliczajaca sume kwadratów oraz `suma3` – obliczajaca sume szescianów. Taka definicja umozliwia niezalezne korzystanie z funkcji `suma b2` i `suma3`.

```
[kwadrat,szescian]=sumaks(a,b)
kwadrat =
    10
     8
    10
szescian =
    28
    16
    28
>> kwadrat1=suma2(a,a)
kwadrat1 =
     2
     8
    18
>> szescian1=suma3(b,b)
szescian1 =
    54
    16
     2
```

W Matlabie istnieje szczególny sposób wywolywania funkcji z wnetrza innych funkcji. Realizowany jest on przy pomocy polecenia `feval`. Polecenie to umozliwia tworzenie funkcji Matlaba, których zmiennymi wejsciowymi sa funkcje dane w postaci m-plików. Dzialanie polecenia `feval` wyjasnione zostanie na bazie dwóch m-plików. Pierwszy z nich o nazwie `sincos.m` oblicza wartosc funkcji *y=sin(x)*cos(x)*

```
function y=sincos(x)
% sincos   oblicza wartosc sin(x)*cos(x)
%          dla dowolnego x
y = sin(x).*cos(x);
```

Drugi o nazwie `fsum.m` oblicza sume wartosci dowolnej funkcji `fun` w `n` równoodleglych punktach w przedziale od `a` do `b`.

```
function s=fsum(fun,a,b,n)
% fsum   Oblicza sume wartosci dowolnej
%        funkcji fun w n równoodleglych
%        punktach przedzialu od a do b
x = linspace(a,b,n);
y = feval(fun,x);
s = sum(y);
```

Chcac obliczyc wartosc sumy wartosci funkcji *y=sin(x)*cos(x)* dla 5 punktów w przedziale od 0 do π nalezy zastosowac wywolanie:

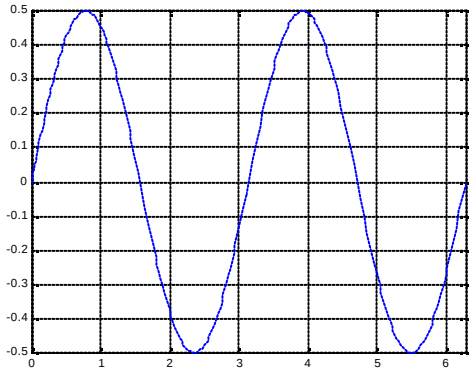
```
>> fsum('sincos',0,pi,5)
ans =
 -1.1442e-017
```

Nazwe m-pliku funkcji, dla której obliczmy sume podajemy w apostrofach. Tak zdefiniowana funkcja `fsum` dziala równiez na wbudowanych funkcjach matematycznych Matlaba np.: *y=sin(x)*, czy tez *y=cos(x)*:

```
>> fsum('sin',0,pi,5)
ans =
    2.4142
>> fsum('cos',0,pi,5)
ans =
     0
```

Matlab dysponuje wbudowanymi poleceniami, które jako argument wejsciowy przyjmuja nazwe funkcji. Jednym z najczesciej wykorzystywanych jest `fplot`, które tworzy wykres danej funkcji w zadanym przedziale.

```
>> fplot('sincos',[0, 2*pi])
>> grid on
```



*Rys.24. Wykres funkcji y=sin(x)*cos(x) w przedziale <0,2p>.*

Poczynajac od wersji 5.0 w Matlabie pojaw ilo sie polecenie `inline`, które eliminuje koniecznosc tworzenia osobnego m-pliku dla prostych funkcji matematycznych takich jak np.: *y=sin(x)*cos(x)*. Zamiast tworzyc m-plik tworzona jest zmienna zawierajaca definicje funkcji:

```
>> sincos=inline('sin(x).*cos(x)');
```

Mozna oczywiscie zastosowac utworzona wczesniej funkcje `fsum`. Wywolujac ja trzeba pamietac, ze nazwa funkcji jest teraz zmienna, co powoduje koniecznosc pominiecia apostrofów.

```
>> fsum(sincos,0,pi,5)
ans =
 -1.1442e-017
```

23.7 Uwagi o programowaniu

Styl programowania to cos co nadaje tworzonej aplikacji latwy do odczytania I zrozumienia wyglad, co ulatwia pojmowanie istoty jej funkcjonowania. Na styl programowania skladaja sie:

- sposób formatowania kodu programu;
- przyjeta konwencja nazw zmiennych I funkcji;
- dokumentacja oraz komentarze.

Formatowanie kodu to w glównej mierze wciecia akapitowe w blokach petli I instrukcji warunkowych oraz puste linie pomiedzy glównymi blokami kodu, np.:

```
if a >= c
     b = sqrt(a^2 – c^2)
elseif a^c > c^a
     b = c^a/a^c
else
     b = a^c/c^a
end
```

Konwencja nazw zmiennych i funkcji. To stosowanie znaczacych nazw, które opisuja znaczenie danej zmiennej lub funkcji w okreslonym fragmencie kodu, np.:

zamiast
```
d = 5;
g = 0.02
r = d/2;
r2 = r +t;
```
```
d_wew = 5;
grubosc = 0.02;
r_wew = d_wew/2;
r_zew = r_wew+grubosc;
```

Nie nalezy jednak przesadzac, konsekwencja w stosowaniu nazw jest wazniejsza od konwencji.

Tworzac dokumentacje kodu nalezy pamietac, ze:

- komentarze pisze sie w chwili tworzenia kodu, a nie pózniej.
- prolog umieszczony we wlasnych funkcjach umozliwia korzystanie z polecenia `help`.
- komentarze powinny byc krótkie i nie powinny powtarzac tego co zawiera kod programu.
- komentarz sam z siebie nie tworzy dobrego programu, nie mozna poprawic kodu programu poprawiajac komentarz

Tworzac wlasne funkcje nalezy pamietac, ze:

- powinny ona byc przeznaczona do rozwiazywania scisle okreslonego pojedynczego zadania,
- komentarze powinny opisywac parametry wejsciowe I wyjsciowe,
- nie nalezy zakladac, ze uzytkownik poda prawidlowe wartosci parametrów wejsciowych,
- w przypadku pojawienia sie bledu funkcja winna wyswietlic odpowiedni komunikat, tak aby uzytkownik mógl zrozumiec powstaly problem,
- celem tworzenia m-plików nie jest maksymalizacja ich liczby lecz uproszczenie rozwiazania problemu.
- polecenie `error` wyswietla w oknie komend komunikat o bledzie i zatrzymuje wykonywania programu,
- polecenie `warning` wyswietla komunikat o bledzie lecz nie przerywa dzialania programu,
- polecenia `pause` oraz `keyboard` mozna uzyc do chwilowego zatrzymania dzialania programu.

24. LITERATURA

[1] Cheney W., Kincaid D.: Numerical Mathematics and Computing, Brooks/Cole Publishing Company, Fourth Edition, 1999.

[2] Dajda J.: Wprowadzenie do Matlab, http://student.uci.agh.edu.pl/~jdajda/mownit/referat, Kraków 2001.

[3] Griffits D.F: An Introduction to Matlab. The University of Dundee, Departament of Mathematics, Dundee 1997.

[4] Recktenwald G.W.: Numerical Methods with Matlab: Implementations and Aplications, Perentice-Hall, New York 2000.

[5] Zalewski A., Cegla R.: Matlab – obliczenia numeryczne i ich zastosowania, Wydawnictwo Nakom, Poznan 1997.

24. SPIS TRESCI