MPDI2

INFORMATYKA

Wykład 9

MATLAB cd

- **Macierze**
- **Iteracja (pętla)**
- **Operacje iteracyjne na macierzach**

TABLICE (MACIERZE)

Tworzenie tablicy

wektor wierszowy

M1=[1 2 3 4 5] lub M1=[1, 2, 3, 4, 5]

wektor kolumnowy

M2=[1; 2; 3; 4; 5]

tablica dwuwymiarowa

M3 = [1 2 3; 2 1 1; 1 0 0]

Metoda generowania tablicy o elementach ciągu arytmetycznego

x=0:2:10 %generowanie wektora od 0 do 10 co 2

% wart_pocz:krok:wart_koncowa

0	2	4	6	8	10

x= 0:0.1:2 %dozwolona wartość dziesiętna kroku

Można pominąć krok:

x=0:10 %generowanie wektora od 0 do 10 co 1

% wart_pocz:wart_koncowa

0	1	2	3	4	5	6	7	8	9	10

Podobnie w tablicach wielowymiarowych

M = [0:5 ; 10:15]

%wartość początkowa: wartość końcowa (krok=1)

0	1	2	3	4	5
10	11	12	13	14	15

ale UWAGA!

M = [0:5; 10:17]

błąd

arguments dimensions are not consistent

różne rozmiary wierszy (6 i 7)

Można wygenerować tablicę serią x i f(x):

```
x= [0 : 2: 180]      %wartość początkowa: krok: wartość końcowa
M=[x; sind(x)]
```

0	2	4	6	....						180
0	0.0349	0.0698	0.1045	....						0

```
x= [1 :10]
M=[x; log(x)]
```

1.0000	2.0000	3.0000	4.0000	5.0000	6.0000	7.0000	8.0000	9.0000	10.0000
0	0.6931	1.0986	1.3863	1.6094	1.7918	1.9459	2.0794	2.1972	2.3026

ale można też osobne dwa wektory:

```
x= [1 :10]
y = log(x)
```

Dostęp do elementu tablicy (macierzy)

NAZWA(*indeks lub indeksy*)

M1=[1 3 5 -11 7]

disp(M1(3))

wyświetlony zostanie trzeci element tablicy (indeks zaczyna się od 1)

M2 = [1 2 3; 2 1 1.5; 1 0 0]

disp(M2(2,3))

ale też M2(8) – odliczanie kolumnami

1 4 7

2 5 8

3 6 9

Można elementy zdefiniowanej tablicy wykorzystać w wyrażeniach:

y= M(2,2)^2

Operacje na tablicach (macierzach)

```
m4=[1 2 34 6;6 8/2 4 -2; 2 -5 56 6; 6 72 0.2 12]
m4t = m4'          %macierz sprzężona – transponowana
m4o = m4^-1        %macierz odwrotna (macierz kwadratowa!)
mo= inv(m)         % także obliczenie macierzy odwrotnej!
s=m4*m4o           %sprawdzenie - macierz jednostkowa
w=det(m4)          %wyznacznik,uwaga:macierz musi być kwadratowa!
```

Połączenie dwóch i więcej macierzy:

```
M=[M1, M2]  % M1 i M2 muszą mieć tyle samo wierszy

M=[M1; M2]  % M1 i M2 muszą mieć tyle samo kolumn
```

Operatory "kropkowe" dla tablic

jeśli A i B są tablicami

C=A*B

to iloczyn macierzowy – kiedy dozwolony? - gdy macierz A ma tyle kolumn ile macierz B wierszy

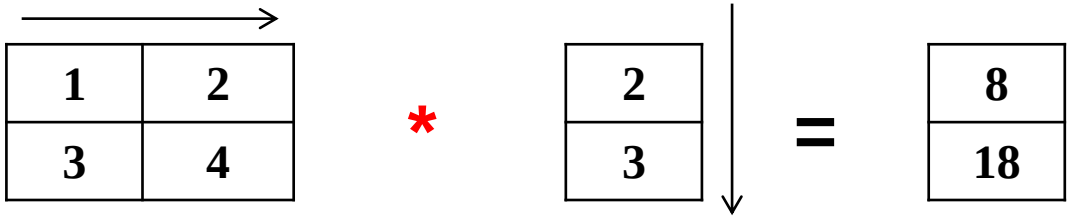
D=A.*B to iloczyn tablicowy – każdy element macierzy D powstaje z iloczynu "odpowiednich" elementów macierzy A i B – dozwolony gdy A i B mają te same wymiary i rozmiary

podobnie **./** **.^** **(dzielenie i potęgowanie tablicowe)**

A^2 % tożsame z A*A (uwaga:A musi być kwadratowa)

A.^2 % każdy element do potęgi – A dowolne

Proste przykłady operacji macierzowych:



1	2
3	4

*

2
3

=

8
18

sumy iloczynów!

1	3
4	1

.*

2	2
3	1

=

2	6
12	1

1	3
4	1

./

2	2
16	1

=

0.5	1.5
0.25	1

2	3
4	6

.^

2	2
3	1

=

4	9
64	6

Wybrane metody wykorzystania macierzy

Rozwiązywanie układu równań liniowych

$$
\begin{aligned}
2x + 3y - 4z &= 5 \\
x + y - z &= 3{,}5 \\
-2{,}5y - z &= 2
\end{aligned}
$$

Rozwiązanie w Matlabie (m-plik):

```
A = [2  3  -4 ; 1  1  -1  ; 0  -2.5  -1]
B = [ 5 ; 3.5  ;  2]
X= A^(-1)*B              %wektor rozwiązań (lub X=inv(A)*B)

A*X     % wynikiem powinien być wektor wyrazów wolnych B
```

…sprawdzenie rozwiązań:

s1=A(1,1)*X(1)+ A(1,2)*X(2)+A(1,3)*X(3)-B(1)

…. powinno dać wartość s1=0

podobnie:

s2=A(2,1)*X(1)+ A(2,2)*X(2)+A(2,3)*X(3)-B(2)

s3=A(3,1)*X(1)+ A(3,2)*X(2)+A(3,3)*X(3)-B(3)

Uwaga: rozwiązania istnieją jeśli równania układu nie są liniowo zależne

Wyznaczanie pierwiastków równania n-tego stopnia

funkcja roots(M)

- gdzie M jest wektorem współczynników przy kolejnych potęgach

np. $x^3 + 3x^2 - 4 = 0$

instrukcja:

R=roots ([1 3 0 -4])

wyznacza pierwiastki równania

R – będzie wektorem rozwiązań

- jeśli równanie rzędu N to mamy N rozwiązań
- rozwiązaniami mogą być liczby zespolone!

Użyteczne wbudowane funkcje tablicowe

rand(n) - losowo generowana tablica kwadratowa nxn

rand(n,m) - losowo generowana tablica prostokątna nxm

sum (A) - wektor sum elementów w kolumnach macierzy A

sum(sum(A)) - suma wszystkich elementów macierzy 2-wymiarowej

max(A) - wektor elementów maksymalnych w kolumnach

max(**max**(A)) - element największy w macierzy dwuwymiarowej

min(A) - wektor elementów minimalnych w kolumnach macierzy A

min(**min**(A)) - element najmniejszy w macierzy dwuwymiarowej

ndims(A) - ile wymiarów macierzy

numel(A) - liczba elementów macierzy

reshape(A,n,m) - rekonfiguracja macierzy

size(A) - rozmiary macierzy

length(A) - największy rozmiar

Macierze specjalne

ones(n,m) - macierz *nxm* wypełniona jedynkami

zeros(n,m) - macierz *nxm* wypełniona zerami

magic(n) - elementy 1..n^2, sumy kolumn i wierszy = const

```
x=magic(4)
y1=sum(x)
y2=sum(x')
```

```
x =
   16    2    3   13
    5   11   10    8
    9    7    6   12
    4   14   15    1
y1 =
   34   34   34   34
y2 =
   34   34   34   34
```

także przekątne mają sumę=34

Przykład

```
M=round(10*rand(3))
w=size(M)
M2=reshape(M,1, 9)
M2=reshape(M',1, 9)
```

```
M =
     2     2     5
     9     3     4
     3     6     8
w =
     3     3
M2=
     2     9     3     2     3     6     5     4     8
M3 =
     2     2     5     9     3     4     3     6     8
```

Sortowanie

sort (A, *i*, *typ*)

i: 1 - kolumny lub 2 - wiersze

t*yp*: 'ascend' – rosnąco

'descend' - malejąco

domyślne wartości: 1 i 'ascend'

```
clc, clear
m=round(10*rand(5))
disp('sortowanie kolumnami')
m1=sort(m,1)
disp('sortowanie wierszami')
m2=sort(m,2)
disp('sortowanie wierszami malejąco')
m2=sort(m,2,'descend')
```

```
m =
    3    5    8   10    8
    7   10    3    5    3
    7    3    5    1    8
    2    6    7    1    2
    1    2    9    3    9
sortowanie kolumn
m1 =
    1    2    3    1    2
    2    3    5    1    3
    3    5    7    3    8
    7    6    8    5    8
    7   10    9   10    9
sortowanie wierszy
m2 =
    3    5    8    8   10
    3    3    5    7   10
    1    3    5    7    8
    1    2    2    6    7
    1    2    3    9    9
sortowanie wierszy malejąco
m2 =
   10    8    8    5    3
   10    7    5    3    3
    8    7    5    3    1
    7    6    2    2    1
    9    9    3    2    1
```

Możliwe jest także tworzenie tzw. tablic komórkowych

A = {[1 8 2005], 'Jakiś tekst'; 2+4i, 1:2:7}

```
A =
        [1x3 double]    'Jakiś tekst'
   [2.0000 + 4.0000i]     [1x4 double]

s1= A{1,1}  % pierwsza składowa tablicy A
          s1 =
                    1     8     2005
```

Po co?

Umożliwiają umieszczenie różnych **typów** danych w komórkach (tablice heterogeniczne) – teksty, dane liczbowe, tablice

Fragmenty wektorów i macierzy dwuwymiarowych:

A(2:5)	fragment wektora (elementy od 2-go do 5-go)
A(2:end)	od 2-go do końca
A(1:2:end)	co drugi element począwszy od pierwszego
A(3,:)	cały trzeci wiersz macierzy A
A(3,2:5)	trzeci wiersz macierzy A o kolumnach od drugiej do piątej
A(:,2)	druga kolumna macierzy A
diag(A)	wektor głównej przekątnej macierzy A

Przykład

```
clear,clc
M=fix(rand(4)*10)
M2=M(2:3,2:3)
M3=M(2:end;1:end)
```

```
M =
     7     6     9     7
     7     1     3     2
     2     1     5     5
     6     4     2     6
M2 =
     1     3
     1     5
M3 =

     7     1     3     2
     2     1     5     5
     6     4     2     6
```

Instrukcja iteracyjna („pętla liczona”)

Schemat iteracji:

```
for zmienna = wartość_pocz : krok: wartość_końcowa
   instrukcja1
   instrukcja2
….itd.
end
```

Jeśli pominięty *krok* to krok=1

Przykłady prostych "pętli":

```
clc
for k= 1:10
        disp('jakiś tekst')
end;
```

```
clear
for k= 1:10
        a(k) = k^2 %generowanie kolejnych elementów wektora

        pause(1)                    %pauza 1 sek.
end;
```

Wykorzystanie licznika pętli do indeksowania tablicy i do obliczania wartości elementu

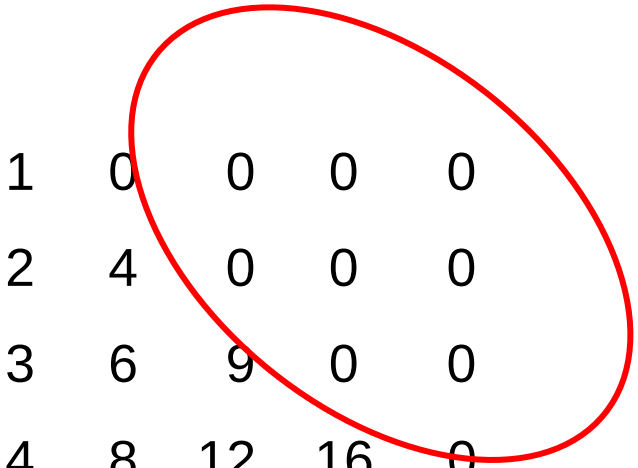
Przykład ("zagnieżdżanie" iteracji):

```
%generowanie kolejnych elementów wektora
for wiersz= 1:4
        for kolumna = 1:5
                a(wiersz , kolumna) = wiersz*kolumna
                pause %stop – ENTER kontynuuje
        end
end
```

Dla każdej wartości licznika pętli zewnętrznej wykonywana jest pętla wewnętrzna, czyli mamy 20 wykonań instrukcji wewnątrz pętli wewnętrznej.

Przykład (uzależnienie licznika "pętli" wewnętrznej):

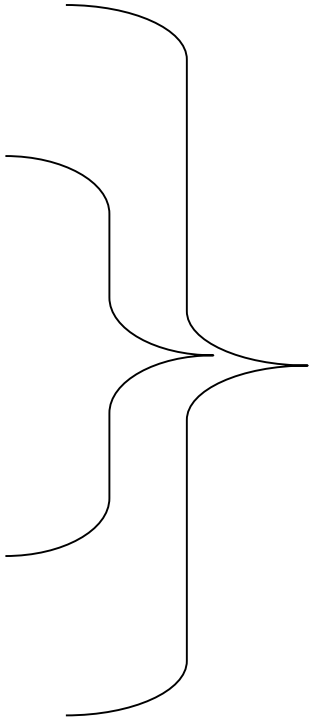
```
for w= 1:5
    for k = 1:w
        a(w , k) = w*k;
    end
end
disp(a)
```



1	0	0	0	0
2	4	0	0	0
3	6	9	0	0
4	8	12	16	0
5	10	15	20	25

Przykład (sumowanie elementów w tablicy dwuwymiarowej):

```
a=0;
s = 0; %konieczne wyzerowanie sumy
for w= 1:5,
 for k= 1:5,
   A(w , k) = 2*w - 4* k,
   pause, %zatrzymuje do naciśnięcia klawisza
   s=s+A(w , k);
 end
end
fprintf('Suma wynosi:%d\n', s)
```



Przykład (sumowanie i zliczanie warunkowe - elementów dodatnich w tablicy dwuwymiarowej)

```
clear; clc
A=rand(5,5)-0.5 %tablica 5x5, elementy z przedziału (-0.5, 0.5)
iledod= 0; sumadod=0;
s=0;
for w= 1:1:5
   for k = 1:1:5
    if  A(w , k)>0
        iledod=iledod+1;
        sumadod=sumadod+A(w,k);
    end
  end
end
fprintf ('Elementów dodatnich jest %d\n', iledod)
fprintf ('Ich suma to %f\n', sumadod)
```

Przy spełnionym warunku sumę zwiększamy o element a liczbę elementów o 1

Przykład

```
A=rand(5)
disp('Oto 3-ci wiersz')
for k= 1:5,
        disp(A(3, k))
end
disp('Oto przekątna')
for k= 1:5,
        disp(A(k, k))
end
```