Scilab

Data ostaniej modyfikacji: **2 grudnia 2008**

Piotr Fulmański

Piotr Fulmański[1]
Wydział Matematyki, Uniwersytet Łódzki
Banacha 22, 90-232, Łódź
Polska
1 email:fulmanp@imul.math.uni.lodz.pl

Spis treści

Wstęp

Wstęp. Najpierw wstęp, bo każda dobra książka musi mieć wstęp. Wstęp nadaje książce charakter, sprawia, że jest ona więcej niż książką – jest ważną, a może i nawet bardzo ważną książką. Im wstęp jest dłuższy i nudniejszy tym książka, zdaniem autora, jest ważniejsza.

Ta książka będzie miała krótki wstęp. Moja przygoda z Scilabem rozpoczęła się w gdzieś około roku 2003. Wówczas środowisko dopiero raczkowało, ale już wtedy było sensownym konkurentem dla Matlaba. Jeszcze ustępowało mu w wielu aspektach, ale pierwsze ziarno zostało zasiane. Na przestrzeni tych kilku lat Scilab istotnie się zmienił i dziś, choć nadal ustępuje Matlabowi, jest bardzo wygodną, użyteczną i co istotne tanią alternatywą dla tego typu aplikacji komercyjnych.

Różnorakie zajęcia i obowiązki jakimi zostałem obdarzony przez pierwsze lata mojej pracy, nie pozwoliły mi poświęcić Scilabowi odpowiedniej ilości czasu. I choć wiedziałem, że warto, to także pomysł poprowadzenia zajęć z tego środowiska musiał zostać odłożony „na potem". Przygotowanie nowego przedmiotu owszem wymaga dużego nakładu pracy ale przede wszystkim czasu. Zwykle w takich przypadkach pomaga „zewnętrzny motywator", który w pewnym sensie zmusza do znalezieni tego czasu. W moim przypadku takim motywatorem stali się uczniowie z I Liceum Ogólnokształcącego im. H. Sienkiewicza w Kędzierzynie Koźlu kierowani przez Tomasza Czyżewskiego. Niezwykły entuzjazm jakim emanował przy okazji każdego kontaktu był wynagrodzeniem za, było nie było, poświęcenie mojego prywatnego czasu. Przecież mógłbym w tym czasie oglądać telewizje (której nota bene nie mam).

Tak czy inaczej, współpraca nasza rozpoczęła się we wrześniu 2008 i mam nadzieje, że z czasem będzie rozwijała się co raz bardziej. Dokument ten stanowi zaś przymiarkę do podręcznika pomocnego w prowadzeniu takich zajęć. Będąc człowiekiem twardo stąpającym po ziemi, doskonale zdaje sobie sprawę, że jest to bardzo zgrubna przymiarka. Horyzont czasowy jaki sobie wyznaczyliśmy na przygotowanie w miarę spójnej wersji to 3 lata. Dopiero po takim czasie można ocenić na ile układ materiału i sposób jego przekazywania jest czytelny.

Ponieważ miało być krótko, więc na zakończenie zachęcam wszystkich do przesyłania wszelkich komentarzy i uwag na temat „podręcznika". W miarę możliwości czasowych będę starał się ustosunkować do wszelkich wniosków.

Piotr Fulmański

Łódź, 10 października, 2008

Rozdział 1

Wektory i macierze

Właściwie to tytuł tego rozdziału powinieni brzmieć: Macierze. Scilab bowiem traktuje wszystkie obiekty jak macierz. Nawet gdy mamy do czynienia z pojedyńczą wartością liczbową to sprawdzając jej rozmiar otrzymujemy odpowiedź, że jej wymiar to 1×1.

```
-->b=3
 b  =

    3.

-->size(b)
 ans  =

    1.    1.
```

1.1 Wprowadzanie macierzy

Najprostszym sposobem definiowania macierzy (a wektora lub skalara w szczególności) w środowisku Scilab jest wprowadzenie z klawiatury listy jej elementów, stosując następującą konwencję:

- elementy tego samego wiersza oddzielone są spacją lub przecinkiem;
- lista elementów musi być ujęta w nawias kwadratowy `[ ]`;
- w przypadku wprowadzania skalarów nawias kwadratowy można pominąć;
- każdy wiersz, z wyjątkiem ostatniego, musi być zakończony średnikiem.

Dla przykładu polecenie:

```
-->A=[1 2 3;4 5 6;7 8 9]
```

powoduje utworzenie macierzy A

```
 A  =
    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

W przypadku, gdy instrukcja zostanie zakończona średnikiem, wynik jej działania nie pojawi się na ekranie[1]. Wpisując

```
-->b=[1 2 3 4];
```

[1] Dotyczy to wszystkich instrukcji, nie tylko tych związanych z macierzami.

tworzymy wierszowy (poziomy) wektor b; nie zostanie on jednak wypisany na ekranie. Aby zobaczyć jego współrzędne, wystarczy napisać

```
-->b
```

a odpowiedzią będzie

```
 b =
   2.  10.  44.  190.
```

W praktyce wygodną cechą jest możliwość wprowadzania jednej macierzy w kilku wierszach. Należy tylko, przechodząc do następnej linii, linię poprzednią zakończyć trzema kropkami, jak w poniższym przykładzie

```
-->T = [1 2 3;...
-->     4 5 6;...
-->     7 8 9]
```

co w efekcie prowadzi do zdefiniowania macierzy T

```
 T  =
    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

Wektor kolumnowy (pionowy) można wprowadzić na dwa sposoby.

Sposób pierwszy: w oparciu o podane zasady zdefiniować jednokolumnową macierz, np.

```
-->t = [1;2;3]
```

co w efekcie prowadzi do zdefiniowania wektora (macierzy) t

```
 t  =
    1.
    2.
    3.
```

Sposób drugi: w oparciu o podane zasady zdefiniować jednowierszową macierz, np.

```
-->t = [1 2 3];
```

a następnie użyć operacji transpozycji (znak ')

```
-->t=t';
```

lub wszystko za jednym razem

```
-->t = [1 2 3]';
```

1.2 Typowe wektory i macierze

Istnieją funkcję do konstrukcji typowych macierzy i wektorów.

Macierz jednostkowa

Aby otrzymać macierz jednostkową o wymiarach 4 na 4 stosujemy polecenie:

```
-->I=eye(4,4)
 I  =
    1.    0.    0.    0.
    0.    1.    0.    0.
    0.    0.    1.    0.
    0.    0.    0.    1.
```

Argumentami funkcji `eye(n,m)` jest liczba wierszy (`n`) oraz kolumn (`m`).

Macierze zerowa i jedynkowa

Funkcje `zeros` i `ones` pozwalają stworzyć, odpowiednio, macierze zerowe i macierze składające się z jedynkek. Podobnie jak dla funkcji `eye` ich argumentami są liczba wierszy i liczba kolumn.

```
-->C = ones(3,4)
 C  =
    1.    1.    1.    1.
    1.    1.    1.    1.
    1.    1.    1.    1.
```

Można także jako argument użyć nazwę macierzy już zdefiniowanej w środowisku. W efekcie otrzymujemy macierz o wymiarach równych wymiarom macierzy będącej argumentem

```
-->Z = zeros(C)
 Z  =
    0.    0.    0.    0.
    0.    0.    0.    0.
    0.    0.    0.    0.
```

Macierz diagonalna, diagonalna macierzy

Aby otrzymać macierz diagonalną D, w której elementy na głównej przekątnej pochodzą z wcześniej zdefiniowanego wektora $d = [1234]$ wpisujemy

```
-->D=diag(d)
 D  =
    1.    0.    0.    0.
    0.    2.    0.    0.
    0.    0.    3.    0.
    0.    0.    0.    4.
```

Funkcja `diag` pozwala uzyskać główną przekątną macierzy jako wektor kolumnowy

```
-->b_2=diag(D)
 b_2  =
    1.
    2.
    3.
    4.
```

Macierz trójkątna

Funcje `triu` i `tril` pozwalają otrzymać macierz trójkątną górną i dolną:

```
-->C = ones(3,4);
-->U = triu(C)
 U  =
    1.    1.    1.    1.
    0.    1.    1.    1.
    0.    0.    1.    1.
-->U = tril(C)
 U  =
    1.    0.    0.    0.
    1.    1.    0.    0.
    1.    1.    1.    0.
```

Dodatkowy argument pozwala na wybór diagonalnej. Przyjrzyjmy się przykładom

```
-->C = ones(5,5);
-->U = triu(C)
 U  =
    1.    0.    0.    0.    0.
    1.    1.    0.    0.    0.
    1.    1.    1.    0.    0.
    1.    1.    1.    1.    0.
    1.    1.    1.    1.    1.
-->U = tril(C,1)
 U  =
    1.    1.    0.    0.    0.
    1.    1.    1.    0.    0.
    1.    1.    1.    1.    0.
    1.    1.    1.    1.    1.
    1.    1.    1.    1.    1.
-->U = triu(C,-2)
 U  =
    0.    0.    0.    0.    0.
    0.    0.    0.    0.    0.
    1.    0.    0.    0.    0.
    1.    1.    0.    0.    0.
    1.    1.    1.    0.    0.
```

Macierze o elementach losowych

Funkcja `rand` pozwala utworzyć macierz o elementach pseudolosowych pochodzących z przedziału (0,1):

```
-->M = rand(2, 5)
 M  =
    0.2113249    0.6653811    0.8497452    0.8782165    0.5608486
    0.7560439    0.3303271    0.6283918    0.0683740    0.6623569
```

n-elementowy wektor o stałej różnicy między elementami

Aby wprowadzić n-elementowy wektor (wierszowy) x o składowych równomiernie rozmieszczonych pomiędzy x_1 i x_n, tzn. aby zachodził związek

$$x_{i+1} - x_i = \frac{x_n - x_1}{n-1},$$

używamy funkcji `linspace`.

```
-->x = linspace(0,1,11)
 x  =
    0.    0.1    0.2    0.3    0.4    0.5    0.6    0.7    0.8    0.9    1.
```

Analogiczną instrukcją która pozwala utworzyć wektor począwszy od zadanej wartości, z ustalonym krokiem (różnicą pomiędzy dwiema kolejnymi składowymi)i ostatniej współrzędnej nie większej niż zadana wartość jest

```
-->y = 0:0.3:1
 y  =
    0.    0.3    0.6    0.9
```

Składnia jest następująca

```
y = wartośćPoczątkowa:przyrost:wartośćGraniczna
```

Gdy przyrost wynosi 1, można go wówczas pominąć:

```
-->i = 1:5
 i   =
    1.    2.    3.    4.    5.
```

1.3 Odwoływanie się do elementów macierzy

Mając daną macierz możemy „wydobyć" z niej pojedyńczy element, cały wiersz lub kolumnę. Zdefiniujmy macierz A postaci

```
 A  =
    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

Wówczas uzyskanie elementu z 2 wiersza i 3 kolumny możliwe jest dzięki poleceniu

```
-->A(2,3)
 ans  =
    6.
```

Polecenie

```
-->A(2,:)
 ans  =
    4.    5.    6.
```

„wydobywa" drugi wiersz z macierzy. W ogólności symbol : oznacza wszystkie elementy (indeksy) z danego wymiaru.

Także odwołanie się do podmacierzy macierzy nie nastręcza żadnych problemów.

```
-->A(1:2,2:3)
 ans  =

    2.    3.
    5.    6.
```

Podmacierz możemy też określić w inny sposób. Zamiast podawać zakres zmienności indeksów w postaci `od:do`, można wymienić indeksy, które nas interesują:

```
-->A([1 3],[1 3])
 ans  =

    1.    3.
    7.    9.
```

Określając zakres zmienności można też podać krok z jakim będzie zmieniał się indeks stosując formę `od:krok:do`

```
-->B=[0 1 2 3 4 5 6 7 8 9;10 11 12 13 14 15 16 17 18 19]
 B  =

    0.    1.    2.    3.    4.    5.    6.    7.    8.    9.
    10.   11.   12.   13.   14.   15.   16.   17.   18.   19.

-->B(:,1:3:10)
 ans  =

    0.    3.    6.    9.
    10.   13.   16.   19.
```

1.4 Usuwanie i dodawanie wierszy oraz kolumn

Do określonej macierzy zawsze możemy dodać lub usunąć wiersze lub kolumny. Do zaznaczenia faktu, że dany element (wiersz, kolumna) ma zostać usunięty wykorzystujemy nawiasy kwadratowe

```
-->A(2,:)=[]
 A  =

    1.    2.    3.
    7.    8.    9.
```

Możemy też macierze łączyć ze sobą. Jeśli mamy dany wektor $v2$

```
-->v2
 v2  =

    4.    5.    6.
```

oraz macierz C postaci

```
-->C=ones(3,3)
 C  =

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.
```

wówczas możemy rozszerzyć macierz C w taki sposób, że wektor $v2$ stanie się jego kolumną

```
-->C(:,4)=v2'
 C  =

    1.    1.    1.    4.
    1.    1.    1.    5.
    1.    1.    1.    6.
```

Co więcej, dodawane macierze nie muszą się ze sobą „stykać"

```
-->C(:,6)=v2'
 C  =

    1.    1.    1.    0.    0.    4.
    1.    1.    1.    0.    0.    5.
    1.    1.    1.    0.    0.    6.
```

Poniżej rozważamy bardziej złożony przykład łączenia macierzy w jedną macierz (czyli przeprowadzania tzw. *konkatenacji* macierzy. Rozważmy następującą macierz podzieloną na

bloki:

$$A = \left(\begin{array}{c|ccc} 1 & 2 & 3 & 4 \\ \hline 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \end{array}\right) = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}.$$

Należy zatem zdefiniować podmacierze $A_{11}, A_{12}, A_{21}, A_{22}$:

```
-->A11=1;

-->A12=[2 3 4];

-->A21=[1;1;1];

-->A22=[4 9 16;8 27 64;16 81 256];
```

ostatecznie otrzymujemy macierz A powstałą z połączenia 4 bloków:

```
-->A=[A11 A12; A21 A22]
A =
!  1.  2.  3.   4.   !
!  1.  4.  9.   16.  !
!  1.  8.  27.  64.  !
!  1.  16. 81.  256. !
```

Z punktu widzenia syntaktyki, macierze blokowe traktowane są jak zwykłe skalary (należy przy tym oczywiście pamiętać o zgodności wymiarów odpowidnich macierzy blokowych).

1.5 Operacje na macierzach i wektorach

Operacje na macierzach i wektorach są tak naturalne jak tylko jest to możliwe i przypominają tradycyjną notację matematyczna. Przyjrzyjmy się przykładowej sesji.

Utwórzymy najpierw macierz na której będziemy działać i nazwijmy ją A

```
-->A = [1 2 3;4 5 6;7 8 9]
 A  =

    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

Na podstawie macierzy A (jej wymiarów) tworzymy zgodną z nią (pod względem wymiarów) macierz jedynkową M i dodajemy je do siebie

```
-->M = A + ones(A)
 M  =

    2.    3.    4.
    5.    6.    7.
    8.    9.    10.
```

W celu dokonania transpozycji używamy operatora '

```
-->At=A'
 At  =

    1.    4.    7.
    2.    5.    8.
    3.    6.    9.
```

Pokażemy teraz jak wykonywać mnożenie macierzy i wektorów. Najpierw jednak trochę teorii, gdyż w tym przypadku należy ściśle przestrzegać odpowiednich wymiarów mnożonych obiektów.

Najczęstszym sposobem mnożenia macierzy jest tzw. *mnożenie Cauchy'ego*. Działanie to zdefiniowane jest wyłącznie dla macierzy, z których pierwsza ma tyle kolumn, co druga wierszy. Jeżeli A jest macierzą $n \times m$, a B to macierz typu $m \times p$, to ich iloczyn, oznaczany AB, czasem też $A \cdot B$, jest macierzą typu $n \times p$. Jeżeli $C = AB$, a $c_{i,j}$ oznacza element macierzy C znajdujący się w i-tym wierszu i j-tej kolumnie, to

$$c_{i,j} = \sum_{t=1}^{m} a_{i,t} b_{t,j}, \quad \text{dla} \quad i = 1, \ldots, n \quad j = 1, \ldots, p.$$

A więc możemy wykonać takie mnożenie:

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \cdot \begin{bmatrix} g & h & i & j \\ k & l & m & n \end{bmatrix} = \begin{bmatrix} ag+bk & ah+bl & ai+bm & aj+bn \\ cg+dk & ch+dl & ci+dm & cj+dn \\ eg+fk & eh+fl & ei+fm & ej+fn \end{bmatrix}$$

ale takiego już nie

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \cdot \begin{bmatrix} g & h & i & j \\ k & l & m & n \\ o & p & q & r \end{bmatrix} = ?$$

Z tej definicji wynika także, że (traktując wektor jak macierz której jeden z wymiarów równy jest 1) można mnożyć macierz przez wektor:

$$\begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \cdot \begin{bmatrix} g \\ h \end{bmatrix} = \begin{bmatrix} ag+bh \\ cg+dh \\ eg+fh \end{bmatrix}$$

i wektor przez macierz:

$$\begin{bmatrix} g & h & i \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} = \begin{bmatrix} ga+hc+ie & gb+hd+if \end{bmatrix}$$

Po takiej dawce teorii możemy przejść do praktyki. Iloczyn macierzy zapisujemy w naturalny sposób

```
-->M = A * A
 M  =

    30.     36.     42.
    66.     81.     96.
    102.    126.    150.
```

Mając dane dwa wektory: $v1$ oraz $v2$

```
-->v1=[1 2 3]
 v1  =

    1.    2.    3.

-->v2=[4 5 6]
 v2  =

    4.    5.    6.
```

możemy spróbować je pomnożyć. Cały czas pamiętać musimy o zgodności wymiarów w przeciwnym razie otrzymamy komunikat podobny to poniższego

```
-->v=v1*v2
       !--error 10
Inconsistent multiplication.
```

Po wykonaniu transpozycji wektora $v2$ operacja mnożenia wektorów jest już możliwa do przeprowadzenia. W zależności od tego który wektor jest wierszowy a który kolumnowy, efektem mnożenia będzie liczba

```
-->v=v1*v2'
 v  =

    32.
```

lub macierz

```
-->v=v1'*v2
 v  =

    4.     5.     6.
    8.     10.    12.
    12.    15.    18.
```

Podobnie z mnożeniem macierzy przez wektor – także i w tym przypadku musimy przestrzegać zgodności zmiennych:

```
-->M=[1 2;3 4;5 6]
 M =

   1.     2.
   3.     4.
   5.     6.

-->a=[7 8]
 a =

   7.     8.

-->M*a
    !--error 10
Inconsistent multiplication.

-->M*a'
 ans  =

    23.
    53.
    83.

-->b=[7 8 9]
 b  =

    7.    8.    9.

-->b'*M
     !--error 10
Inconsistent multiplication.

-->b*M
 ans  =

    76.    100.
```

Jeśli chcemy przeprowadzić operację na odpowiadających sobie elementach macierzy, wówczas operator należy poprzedzić znakiem . (kropka).

```
-->v=v1.*v2
 v  =

    4.    10.    18.

-->A./A
 ans  =

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.
```

Pomnożenie macierzy przez skalar (liczbę) daje łatwy do przewidzenia efekt w postaci pomnożenia każdego elementu tej macierzy przez skalar

```
-->M = A*2
 M  =

    2.     4.     6.
    8.     10.    12.
    14.    16.    18.
```

Warto też wiedzieć o możliwości odwołania się do ostatniego wyniku, jeśli nie nastąpiło jawne przypisanie do zmiennej. Odwołanie to następuje wówczas przez automatycznie tworzoną zmienną o nazwie `ans`.

```
-->A*2
 ans  =

    2.     4.     6.
    8.     10.    12.
    14.    16.    18.

-->N = ans * (-1)
 N  =

  - 2.   - 4.   - 6.
  - 8.   - 10.  - 12.
  - 14.  - 16.  - 18.
```

Podobnie jak mnożenie macierzy przez skalar skutkowało wykonaniem tej operacji (to jest mnożenia przez skalar) na każdym elemencie macierzy, tak wykonanie funkcji na argumencie, którym jest macierz, skutkuje tym, że funkcja wykonana jest na każdym elemencie macierzy

```
-->sqrt(A)
 ans  =

    1.           1.4142136    1.7320508
    2.           2.236068     2.4494897
    2.6457513    2.8284271    3.
```

1.6 Macierze wielowymiarowe

Macierze wielowmiarowe są uogólnieniem tradycyjnie pojmowanej macierzy pozwalającym na stosowanie dowolnej ilości wymiarów. Przykładową trójwymiarową macierz jedynkową tworzymy w następujący sposób

```
-->M=ones(3,3,3)
 M  =

(:,:,1)

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.
(:,:,2)

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.
(:,:,3)

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.
```

Chcąc zmienić jakiś element macierzy podajemy jego indeks w dokładnie taki sam sposób jak dla macierzy dwuwymiarowych

```
-->M(2,2,2)=2
 M  =

(:,:,1)

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.
(:,:,2)

    1.    1.    1.
    1.    2.    1.
    1.    1.    1.
(:,:,3)

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.

-->M(3,3,3)=3
 M  =

(:,:,1)

    1.    1.    1.
    1.    1.    1.
    1.    1.    1.
(:,:,2)

    1.    1.    1.
    1.    2.    1.
    1.    1.    1.
(:,:,3)

    1.    1.    1.
    1.    1.    1.
    1.    1.    3.
```

Zmianę możemy wprowadzić, w większej ilośći elementów używając np. znaku :

```
-->M(2,2,:)=3
 M  =

(:,:,1)

    1.    1.    1.
    1.    3.    1.
    1.    1.    1.
(:,:,2)

    1.    1.    1.
    1.    3.    1.
    1.    1.    1.
(:,:,3)

    1.    1.    1.
    1.    3.    1.
    1.    1.    3.

-->M(2,:,:)
 ans  =

(:,:,1)

    1.    1.    1.
(:,:,2)

    1.    2.    1.
(:,:,3)

    1.    1.    1.
```