

2012

Politechnika Łódzka

Instytut Informatyki
Stosowanej

Opracował:
Dr hab. inż. Jacek
Kucharski

MATLAB

WIADOMOŚCI OGÓLNE

MATLAB jest **językiem programowania** należącym do grupy **interpreterów**. Kiedy uruchamiamy MATLABa zgłasza się *Matlab Command Window* i poprzez ten mechanizm możemy komunikować się z interpreterem. Pisząc **demo** zapoznamy się krótko z możliwościami MATLABa.

Po wprowadzeniu każdego polecenia MATLAB sprawdza kolejno:

1. Czy jest to zmienna
2. Jeżeli nie to czy jest to funkcja wbudowana MATLABa (built-in function)
3. Jeżeli nie to czy jest taka funkcja (typu M-pliku, MEX-pliku albo DLL) w bieżącym katalogu
4. Jeżeli nie to czy jest taka funkcja w ścieżce dostępu MATLABa (MATLAB search path)

Ścieżka dostępu MATLABa zapisana jest w pliku *matalabrc.m*, który jest wykonywany automatycznie jako pierwsze polecenie po uruchomieniu programu. Ścieżkę tę można sprawdzać i modyfikować poleceniem `path`.

```
>>path - sprawdza ścieżkę  
>>path('C:\.....', path) - dodaje nową ścieżkę do listy.
```

Można wrócić do poprzednio wykonywanych poleceń używając klawiszy $\uparrow\downarrow$. Jeżeli wcześniej zostaną wpisane pierwsze litery poszukiwanego polecenia to zostanie odnalezione najbliższe na liście takie polecenie. Polecenie takie można edytować i ponownie wykonać.

MATLAB rozróżnia małe i wielkie litery. Wszystkie funkcje muszą być wprowadzane małymi literami.

Można z poziomu MATLABa wykonywać polecenia DOSa i WINDOWS w następujący sposób:

```
>>!dir - otwarcie nowego okna, wykonanie polecenia i oczekiwanie na jego  
zakończenie  
>>!dir& - otwarcie nowego okna i wykonanie polecenia w tle bez oczekiwania na jego  
zakończenie  
>>!dir| - wykonuje polecenie w tle
```

Zakończenie pracy w MATLABie następuje po podaniu polecenia `quit` lub `exit`. Jest to równoznaczne z wymazaniem wszystkich zmiennych w przestrzeni roboczej. Całość lub część przestrzeni roboczej można zapisać w pliku dyskowym przed zakończeniem pracy w MATLABie:

```
>>save - zapisuje wszystkie zmienne w pliku matlab.mat  
>>save temp - zapisuje wszystkie zmienne w pliku temp.mat  
>>save temp X Y - zapisuje zmienne X i Y w pliku temp.mat
```

Przywrócenie przestrzeni roboczej po ponownym uruchomieniu programu uzyskuje się poleceniem `load`, np. `load temp`.

help – wyświetla zestawienie katalogów MATLAB-a wraz z ich opisem

help nazwa polecenia – podaje odpowiedź na temat żadanego polecenia, np. help ver

lookfor szukana_nazwa – przeszukuje wszystkie teksty pomocy w celu odnalezienia szukanej nazwy, lub jej fragmentów,

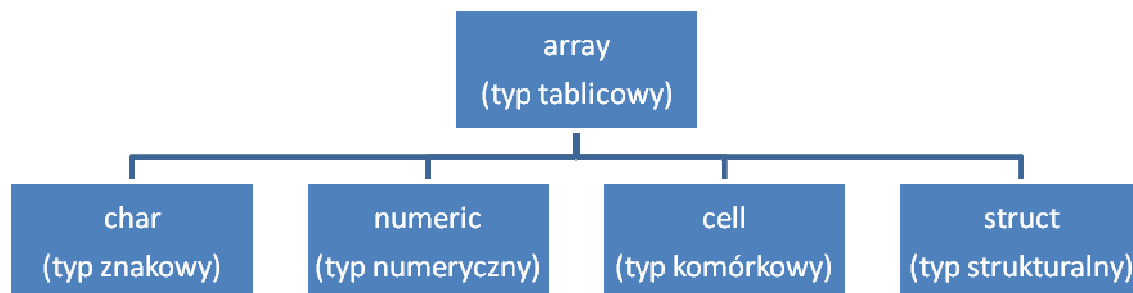
which nazwa_m-pliku – powoduje wyświetlenie ścieżki dostępu do m-pliku lub do m-funkcji
matlabpath – podaje przeszukiwane katalogi programu MATLAB

LICZBY, ZMIENNE I WYRAŻENIA

Zapis liczb jest typowy dla notacji matematycznej (dopuszczalne są cyfry, znaki +-, kropka dziesiętna oraz jednostka urojona i lub j dla liczb zespolonych). Należy pamiętać, aby przy pisaniu elementów macierzy unikać spacji w liczbach zespolonych i przed potęgą e ($1 + 3i$, $1.34 e-5$). Jeżeli w danej przestrzeni roboczej używamy zmiennej i lub j do innych celów niż jednostka urojona to trzeba zdefiniować nową jednostkę urojoną, np. `ii=sqrt(-1)` ponieważ następuje przysłanianie zmiennych.

Typy danych

Obecnie w MATLAB-ie zostało zdefiniowanych sześć typów (klas) danych, z których każdy może być tablicą wielowymiarową. Na szczycie tej hierarchii znajduje się tablica (array).



numeric- typ numeryczny podwójnej precyzji. Jest to podstawowy typ danych dla zmiennych MATLAB-a (wszystkie obliczenia dla zmiennych numerycznych są prowadzone w trybie podwójnej precyzji).

char – typ tablicowy znakowy lub łańcuchowy (każdy znak zapisywany jest 16-bitowo).

cell – typ komórkowy. Elementy tablic komórkowych mogą zawierać inne tablice,

struct – typ strukturalny. Odwołują się do nazw pól, które mogą zawierać inne tablice.

Zatem w środowisku MATLAB dostępne są:

- macierze,
- tablice wielowymiarowe,
- tablice komórkowe,
- tablice strukturalne,

Podstawowym typem zmiennych w MATLAB-ie jest macierz. W ogólnym przypadku jej elementami mogą być liczby zespolone. Jeżeli macierz ma jedną kolumnę lub jeden wiersz to jest to **wektor**, a jeśli wymiar macierzy jest 1x1 to mamy do czynienia ze **skalarem**.

Dwie zmienne mają specjalny charakter (są permanentne tzn. nie można ich wymazać podczas pracy). Są to **ans** - standardowa zmienna wynikowa i **eps** - zmienna tolerancji.

Ponadto występują dwie zmienne dla wyrażeń nieokreślonych tj $\text{Inf} = \infty$ jako wynik operacji dzielenia przez zero oraz **NaN** (*Not a Number*) dla wyrażeń typu $0/0$ lub ∞/∞ .

Nazwy zmiennych lub funkcji nie mogą przekraczać 19 znaków (tyle zapamięta MATLAB), mogą składać się z liter i cyfr, ale muszą się rozpoczynać od litery.

Polecenie **who** podaje **aktualną listę zmiennych** w przestrzeni roboczej (**whos** dodatkowo podaje ich rozmiary).

Można usunąć zmienną z przestrzeni roboczej poleceniem **clear**, np. **clear X**.

Wyrażenie w MATLABie ma typową formę postaci:

variable=expression

lub po prostu

expression

Wyrażenie może składać się z operatorów, nazw funkcji i zmiennych.

Wynikiem wyrażenia jest macierz.

Jeżeli wyrażenie podane jest bez operacji przypisania (drugi z podanych sposobów) to następuje **automatyczne przypisanie** do zmiennej **ans**.

Jeżeli wyrażenie zakończone jest **średnikiem** (;) to wynik wyrażenia nie jest wyprowadzany na ekran (ale przypisanie wciąż ma miejsce).

Jeżeli wyrażenie jest tak długie, że **nie mieści się w jednej linii** to można je przerwać trzema kropkami (...) i kontynuować w następnej linii (już bez kropek).

W wyrażeniach wykorzystuje się **typowe operatory arytmetyczne**: +, -, *, ^, /, \. Dwa ostatnie operatory to dzielenie prawo i lewostronne rozróżniane w przypadku macierzy, ale dla skalarów jest to to samo ($4/2=4\backslash 2$). Kropka dziesiętna poprzedzająca znak operatora (np. .*, ./) oznacza **operację tablicową** tj. element po elemencie.

Format wyświetlanych wyników może być określony przez użytkownika przez polecenie **format** z jednym z parametrów: short, short e, long, long e, bank, hex, +, compact.

OPERACJE NA MACIERZACH

Macierze jedno- i dwuwymiarowe.

Macierz może być zdefiniowana na kilka sposobów:

- **przez wyliczenie jej elementów**

W nawiasach kwadratowych podaje się elementy macierzy oddzielając wiersze średnikiem. Wyrazy w wierszu mogą być oddzielone spacją lub przecinkiem:

$$A=[1\ 2\ 3; 4,5,6]$$

Można też wyliczyć elementy układając je „graficznie” wewnątrz macierzy, gdyż sekwencja oznaczająca koniec wiersza (ENTER) jest traktowana jak średnik rozdzielający wiersze budowanej macierzy:

$$A=[1\ 2\ 3\ \text{(tu jest ENTER)} \\ 4\ 5\ 6]$$

Możliwe jest też wykorzystanie trzech kropek dziesiętnych w celu kontynuacji wiersza w nowej linii, tak więc polecenie

$$A=[1\ 2\ 3\ \dots \\ 4\ 5\ 6]$$

jest równoważne poleceniu

$$A=[1\ 2\ 3\ 4\ 5\ 6]$$

- **przez wygenerowanie elementów**

Wykorzystujemy wyrażenie w ogólnej postaci:

$$\text{min:krok:max}$$

które generuje wektor wierszowy o następującej budowie:

$$[\text{min}, \text{min}+\text{krok}, \text{min}+2*\text{krok}, \dots, \text{max}]$$

Jeżeli parametr krok jest pominięty to jest on domyślnie przyjmowany jako 1.

Macierze są też generowane jako efekt wyrażen lub funkcji.

- **przez zbudowanie z innych elementów**

Mając zdefiniowane macierze A, B, C można zbudować macierz D np. jako:

$$D=[A\ B; C]$$

co daje układ

$$D = \begin{bmatrix} A & B \\ C \end{bmatrix}$$

- **przez mieszanie omówionych wyżej technik**

Przykład: jeżeli $A=[1\ 2\ 3; 4\ 5\ 6]$ to przez podstawienie

$$D=[A, [1;2]; 1:4]$$

uzyskamy:

$$D = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 2 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Dostęp do elementów macierzy uzyskuje się przez podanie współrzędnych podmacierzy w nawiasach okrągłych za nazwą macierzy:

- $A(2,4)$ - czwarty element w drugim wierszu,
- $A(1:2,1:2)$ - podmacierz czterech lewych górnych elementów,
- $A(2,1:6)$ - drugi wiersz elementy od 1 do 6,
- $A(2,:)$ - wszystkie elementy drugiego wiersza,
- $A([1\ 3], 1:2:5)$ - elementy na przecięciu 1 i 3 wiersza z 1, 3 i 5 kolumną.

Można też wybierać **elementy spełniające pewne warunki**:

- $A(A>3)$ - wybiera wszystkie elementy macierzy większe od 3,
- $A(:,A(3,:)>2)$ - wybiera te kolumny, których trzeci wiersz ma element większy niż 2.

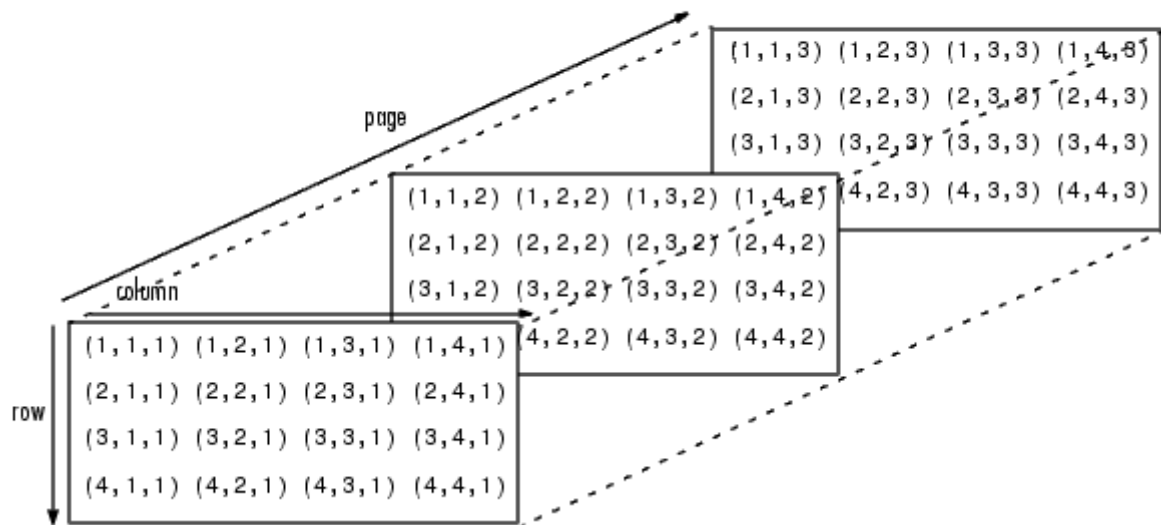
Zawartość macierzy A można wysłać do okna poleceń funkcją $\text{disp}(A)$.

Aktualny **rozmiar macierzy** można sprawdzić poleceniem size

- $[n,m]=\text{size}(A)$ - zwraca wierszy n i kolumn m macierzy A ,
- $n=\text{size}(A,1)$ - zwraca liczbę wierszy,
- $m=\text{size}(A,2)$ - zwraca liczbę kolumn.
- $k=\text{size}(A,3)$ - zwraca liczbę stron.

Tablice wielowymiarowe

Tablice wielowymiarowe w MATLAB-ie są rozszerzeniem tablic dwuwymiarowych. Przykład macierzy trójwymiarowej przedstawia rysunek 1



Rys. 1 Graficzna interpretacja macierzy trójwymiarowej

Tablice wielowymiarowe można tworzyć następująco:

1. Poprzez indeksowanie

$$A = [5 \ 7 \ 8; 0 \ 1 \ 9; 4 \ 3 \ 6];$$

$$A(:, :, 2) = [1 \ 0 \ 4; 3 \ 5 \ 6; 9 \ 8 \ 7];$$

Wynikiem powyższych operacji będzie macierz:

$$A(:, :, 1) =$$

$$\begin{matrix} 5 & 7 & 8 \\ 0 & 1 & 9 \\ 4 & 3 & 6 \end{matrix}$$

$$A(:, :, 2) =$$

$$\begin{matrix} 1 & 0 & 4 \\ 3 & 5 & 6 \\ 9 & 8 & 7 \end{matrix}$$

2. Poprzez zastosowanie do tworzenia tablicy funkcji ones, zeros, randn, repmat:

$$A = \text{ones}(3, 2, 3);$$

$$A(:, :, 1) =$$

$$\begin{matrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{matrix}$$

$A(:,:,2) =$

```
1 1
1 1
1 1
```

$A(:,:,3) =$

```
1 1
1 1
1 1
```

$B = \text{randn}(2,3,2);$

$B(:,:,1) =$

```
-0.4326  0.1253 -1.1465
-1.6656  0.2877  1.1909
```

$B(:,:,2) =$

```
1.1892  0.3273 -0.1867
-0.0376  0.1746  0.7258
```

$B = \text{repmat}(5,[3 4 2]);$

$B(:,:,1) =$

```
5 5 5 5
5 5 5 5
5 5 5 5
```

$B(:,:,2) =$

```
5 5 5 5
5 5 5 5
5 5 5 5
```

3. Za pomocą konkatencji (scalania) tablic:

$B = \text{cat}(\text{dim}, A1, A2\dots)$, gdzie dim jest rozmiarem macierzy wynikowej.

$B = \text{cat}(3, [2 8; 0 5], [1 3; 7 9])$

$B(:,:,1) =$

```
2 8
0 5
```

$B(:,:,2) =$

```
1 3
7 9
```

Funkcja cat dodaje automatycznie indeks równy 1, jeśli zajdzie taka potrzeba, np.:

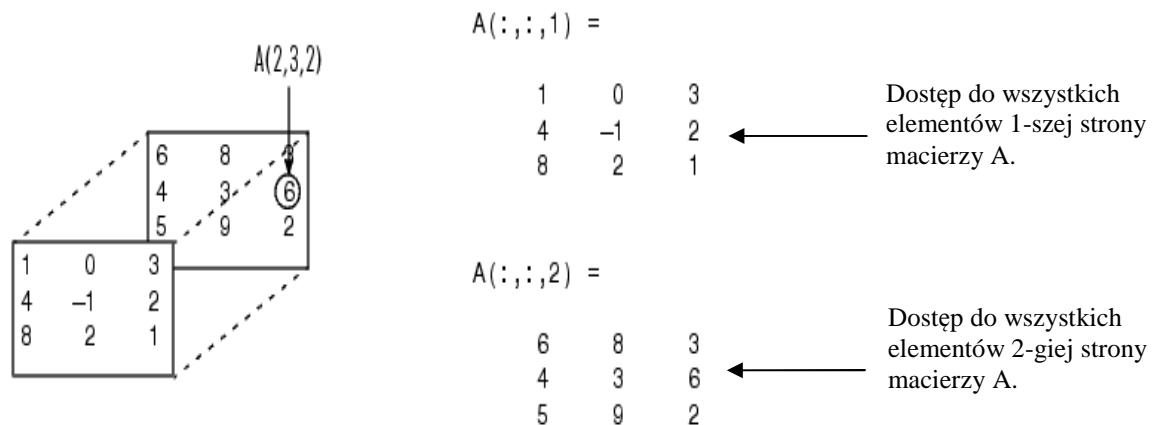
$C = \text{cat}(4, [1 2; 4 5], [7 8; 3 2]);$

Wynikiem będzie macierz C o rozmiarach:
 $C(1,2, 1,2)$

Liczbę indeksów tablicy wielowymiarowej pobieramy za pomocą funkcji **ndims**.
Indeks równy 1

Działania na tablicach wielowymiarowych

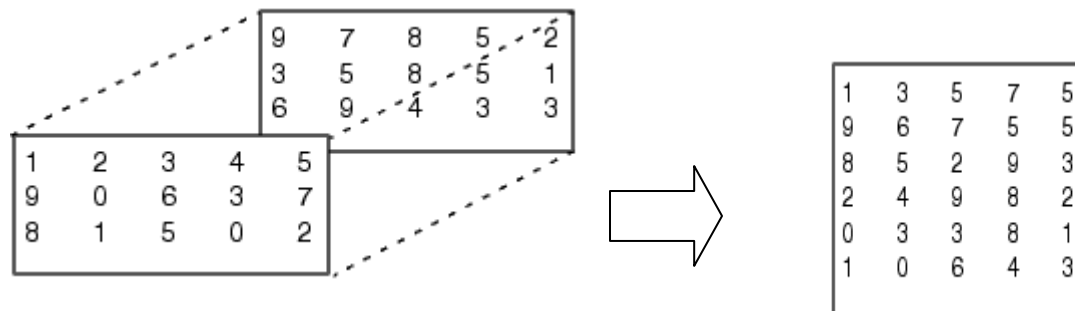
1. Podstawowe sposoby znajdowania elementów macierzy A przedstawia rys.2



Rys. 2 Znajdywanie elementów macierzy A

Zmiana kształtu tablicy za pomocą funkcji reshape:

reshape(x,m,n) – zwraca macierz o nowym wymiarze $m \times n$, której elementy są elementami macierzy x



Rys. 3. Działanie funkcji **reshape**

Przykład:

Niech $B = \text{cat}(3, [2 \ 8; 0 \ 5], [1 \ 3; 7 \ 9])$,

Wtedy:

`prod(size(B))`

```
ans =
```

```
8
```

```
>> C=reshape(B,[2 4])
```

```
C =
```

```
2 8 1 3  
0 5 7 9
```

Dostęp do podtablicy:

Stosując dwukropek : można wycinać fragmenty tablicy lub tablic, np.:

Niech

```
A=[1 9 2  
4 7 3];
```

```
B=[5 9 6  
0 8 10];
```

```
C=cat(3,A,B)
```

Zatem:

```
C(:,:,1) =  
1 9 2  
4 7 3
```

```
C(:,:,2) =  
5 9 6  
0 8 10
```

Liczba elementów w macierzy C wynosi:
prod(size(C))

```
ans =
```

```
12
```

Wybieramy 3 wiersze, 2 kolumny i 2 strony

```
reshape(C,[3 2 2])
```

```
ans(:,:,1) =  
1 7  
4 2  
9 3
```

```
ans(:,:,2) =  
5 8  
0 6  
9 10
```

Tablice komórkowe

Tablice komórkowe są specjalną klasą tablic w MATLAB-ie. Elementami tablic komórkowych są komórki (cells), które mogą z kolei zawierać inne tablice.

cell 1,1 <table border="1"><tr><td>1</td><td>4</td><td>3</td></tr><tr><td>0</td><td>5</td><td>8</td></tr><tr><td>7</td><td>2</td><td>9</td></tr></table>	1	4	3	0	5	8	7	2	9	cell 1,2 'Anne Smith'	cell 1,3 []
1	4	3									
0	5	8									
7	2	9									
cell 2,1 3+7i	cell 2,2 [-3.14...3.14]	cell 2,3 []									
cell 3,1 []	cell 3,2 []	cell 3,3 5									

Rys. 5. Tablica komórkowa

Konstrukтором tablicy komórkowej są nawiasy { }. Działają one podobnie jak nawiasy []. Nawiasy { } mogą być zagnieżdżane (mogą tworzyć komórkę w komórce). Jako separator elementów w nawiasach { } jest stosowany przecinek (,) lub spacja, która oznacza zmianę kolumny, średnik zaś (;) zmianę wiersza.

Tworzenie tablicy komórkowej:

1. Przez indeksowanie

$A(1,1) = \{[1\ 4\ 3; 0\ 5\ 8; 7\ 2\ 9]\};$

$A(1,2) = \{'Anne\ Smith'\};$

$A(2,1) = \{3+7i\};$

$A(2,2) = \{-pi:pi/10:pi\};$

celldisp(A)

$A\{1,1\} =$
1 4 3
0 5 8
7 2 9

$A\{2,1\} =$
3.0000 + 7.0000i

$A\{1,2\} =$
Anne Smith

$A\{2,2\} =$

Columns 1 through 8

-3.1416 -2.8274 -2.5133 -2.1991 -1.8850 -1.5708 -1.2566 -0.9425

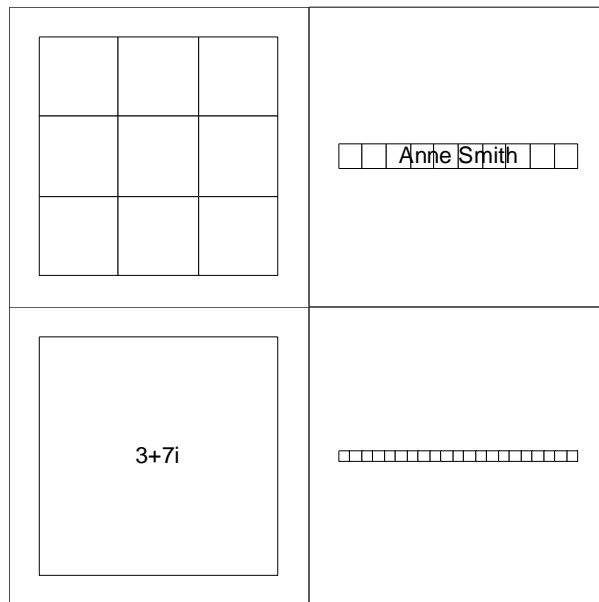
Columns 9 through 16

-0.6283 -0.3142 0 0.3142 0.6283 0.9425 1.2566 1.5708

Columns 17 through 21

1.8850 2.1991 2.5133 2.8274 3.1416

Tablicę komórkową można także przedstawić graficznie za pomocą polecenia **cellplot(A)**.



Rys.4. Graficzne prezentacja tablicy komórkowej

2. **Przez przypisanie danych do komórki, których indeks przekracza rozmiar tablicy komórkowej**

W takim przypadku zachodzi rozszerzenie bieżącej tablicy komórkowej do wymaganego wymiaru.

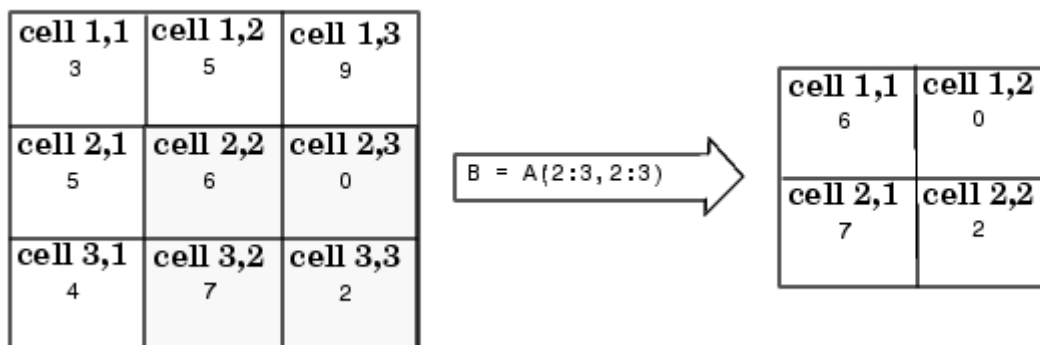
Przykład: do poprzednio wygenerowanej macierzy A dodajemy:

$A(3,3) = \{5\}$;

W wyniku otrzymamy tablicę komórkową postaci:

<table border="1"><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>										<input type="text" value="Arnie Smith"/>	
<table border="1"><tr><td>$3+7i$</td></tr></table>	$3+7i$	<table border="1"><tr><td></td></tr></table>									
$3+7i$											
		<table border="1"><tr><td>5</td></tr></table>	5								
5											

Tworzenie nowej macierzy B poprzez przypisanie jej elementów podmacierzy A, np.:



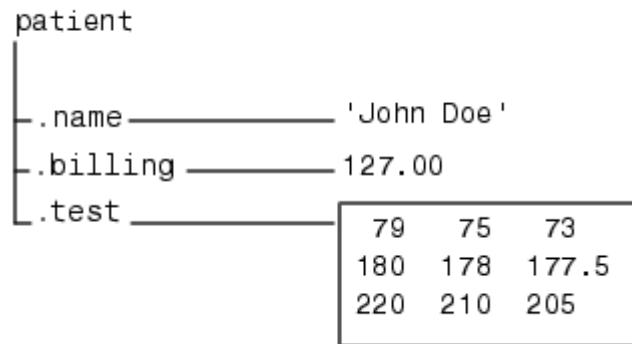
Kasowanie komórek

Odbywa się poprzez przypisanie komórce tablicy pustej, np.:

$A\{3,3\} = [];$

Tablice strukturalne

tablice strukturalne są odmianą tablic MATLAB-a, w których dostęp do danych jest możliwy przez podanie nazwy pola. Pola tablicy strukturalnej mogą zawierać różne typy danych, np.:



Sposoby tworzenia tablic strukturalnych

1. Używając instrukcji przypisania, np.:

```
patient.name = 'John Doe';
patient.billing = 127.00;
patient.test = [79 75 73; 180 178 177.5; 220 210 205];
```

ELEMENTY JĘZYKA MATLAB

MATLAB jest językiem programowania o składni zapożyczony z języka C. Zawiera on instrukcję warunkową, dwa rodzaje instrukcji iteracyjnych oraz instrukcje przerywania wykonywanych instrukcji. Dodatkowo polecenie `error` pozwala na tworzenie własnej diagnostyki błędów. Pełne zestawienie elementów języka można uzyskać poprzez polecenie `help lang`.

INSTRUKCJE

- **warunkowa**

```
if wyrażenie
    polecenia
elseif wyrażenie
    polecenia
else
    polecenia
end
```

Polecenia po `if` są wykonywane jeśli macierz będąca wynikiem wyrażenia jest prawdą logiczną tzn. jej wszystkie elementy są niezerowe. Instrukcja ta może pełnić rolę instrukcji wyboru, ale `elseif` i `else` są opcjonalne.

- **iteracyjne**

1. z nieokreśloną liczbą obiegu pętli

```
while wyrażenie
    polecenia
end
```

Wykonuje polecenia dopóki wartość wyrażenia jest prawdą (patrz warunkowa)

2. z określoną liczbą obiegu pętli

```
for zmienna_iterowana=macierz_wartości
    polecenia
end;
```

Wykonuje polecenia dla kolejnych wartości *zmiennej iterowanej*. Wartościami tymi są kolejne wektory kolumnowe pobrane z *macierzy wartości*, jeżeli więc ta macierz jest wektorem wierszowym to polecenia wykonywane są dla kolejnych elementów tego wektora. O uszeregowaniu wartości decyduje programista umieszczając je w odpowiedniej kolejności w macierzy wartości. Nie musi być tam ciągów rosnących czy malejących.

- **instrukcja break**

Powoduje przerwanie wykonywania pętli przy czym opuszczany jest tylko jeden poziom zagłębienia pętli.

- **instrukcja return**

Powoduje bezwarunkowe opuszczenie danej funkcji lub skryptu i powrót do miejsca jej wywołania.

M-pliki

Programy napisane w języku MATLABa umieszczane są w plikach dyskowych z rozszerzeniem .m (tzw. M-pliki) i mogą być tworzone przy użyciu dowolnego edytora tekstowego. Mogą one zawierać oprócz sekwencji poleceń i funkcji MATLABa, wywołania innych M-plików, jak również wywołania samych siebie.

Rozróżnia się dwa rodzaje M-plików: skryptowe i funkcyjne.

M-pliki skryptowe zawierają ciągi poleceń operujących na zmiennych globalnych zawartych w przestrzeni roboczej. Wykorzystuje się je w celu grupowania poleceń użytkownika ułatwiając ich wykonywanie np. w przypadku powtarzających się operacji. Skrypt nie posiada mechanizmu hermetyzacji tzn. dane utworzone w skrypcie zostaną dołączone do danych w przestrzeni roboczej.

Komentarze w skrypcie poprzedza się znakiem %. Pierwszy blok komentarzy wysyłany jest na ekran jako pomoc dla danego skryptu po poleceniu `help nazwa_skryptu`.

M-pliki funkcyjne są to funkcje tworzone przez użytkownika operujące na zmiennych lokalnych i komunikujące się z przestrzenią roboczą poprzez parametry formalne. Takie M-pliki muszą się zaczynać od słowa kluczowego `function`. Pierwsza linia M-pliku funkcyjnego powinna być zapisana następująco:

`function [lista argumentów wyjściowych]=nazwa_funkcji (lista argumentów wejściowych)`

Nazwa funkcji musi być taka sama jak nazwa M-pliku w którym się znajduje. Argumenty oddziela się przecinkami. Funkcja może nie mieć żadnych argumentów, wtedy nawiasy okrągłe obejmują listę pustą.

Argumenty wej i wyj oraz wszystkie zmienne używane wewnątrz mają charakter lokalny i nie są powiązane z przestrzenią roboczą (nawet jeśli mają te same nazwy). Zmienne te są usuwane po zakończeniu funkcji. Podobnie nie są widziane wewnątrz funkcji zmienne utworzone poza nią. Chyba że użyto polecenia `global` ale nie jest to zalecane. Argumenty funkcji są przekazywane przez wartość tzn. w ciele funkcji tworzona jest kopia zmiennych. Aby zwrócić wartość funkcji konieczne jest dokonanie odpowiednich przypisań w ciele funkcji. (jeżeli jest funkcja $Y(x)$ to gdzieś w funkcji musi być $Y=...$). Jeśli nie ma takiego przypisania to funkcja zwraca pustą macierz (nie ma tutaj `ans`).

Dwie zmienne standardowe zawierają liczby przekazywanych w aktualnym wywołaniu parametrów: **nargin** -wejściowych, **nargout** - wyjściowych. Można więc wywoływać funkcje z mniejszą niż zadeklarowana liczbą parametrów ale trzeba odpowiednio ustalić co w takim przypadku ma zrobić funkcja. (właśnie tu przydają się te zmienne).

Komentarze w funkcjach są wprowadzane podobnie jak w skryptach, a komentarz umieszczony bezpośrednio pod nagłówkiem funkcji będzie używany jako help.

FUNKCJE STANDARDOWE

Funkcje do pracy interakcyjnej:

input - tekst jako znak zachęty do prowadzenia danych
keyboard - wywołanie klawiatury w trakcie trwania M-pliku
menu - generowanie okna dla wyboru danych użytkownika
pause - wstrzymanie wykonywania pliku i czekanie na reakcję użytkownika

Funkcje do pomiaru czasu:

date - zwraca łańcuch zawierający aktualną datę,
clock - zwraca wektor postaci [rok miesiąc dzień godz min sek]
cputime - wyznacza czas procesora wykorzystany przez MATLABa od chwili uruchomienia.
 tstart=cputime
 operacje
 toper=cputime-tstart
etime(T1,T2) - zwraca różnicę czasu między parametrami wywołania (T1 późniejsza niż T2), przy czym są one wektorem wyjściowym funkcji **clock**

tic
operacje
toc

mierzy czas wykonania operacji i wyświetla go na ekranie.

Jeżeli zachodzi potrzeba obliczania funkcji zapisanych w postaci ogólnej to korzystamy z funkcji **feval**

feval(nazwa_funkcji, x1,x2...xn)

oblicza ona wartość funkcji o nazwie określonej łańcuchem znakowym dla podanych argumentów, np. **feval('cos',[0:0.01:2pi])**

Podobną funkcją jest **eval**, która pozwala na wykonanie poleceń MATLABa zapisanych w postaci łańcucha znaków.

ELEMENTY GRAFICZNE W MATLAB-ie

GRAFIKA DWUWYMIAROWA

Podstawową funkcją służącą do rysowania wykresów dwuwymiarowych jest funkcja `plot`. Przykładowy sposób jej wywołania to

`plot(x,y,s)`

gdzie `x` i `y` to wektory odwzorowujące funkcje `y(x)` a `s` - łańcuch znaków określający kolor i rodzaj linii.

Wywołanie tej funkcji powoduje otwarcie okna graficznego i umieszczenie w nim wykresu w skalach liniowych. **Skale logarytmiczne i półlogarytmiczne** można uzyskać odmianami funkcji `plot`: `loglog()`, `semilogx()`, `semilogy()`. o tych samych argumentach.

Można pracować z wieloma oknami graficznymi równocześnie tworząc kolejne poleceniem `figure`. Jedno z nich jest zawsze aktywne i wszystkie operacje graficzne dotyczą tego właśnie okna. Zmiana okna aktywnego następuje po funkcji `figure` z numerem okna jako parametrem. Okno aktywne można oczyścić poleceniem `clf`, a dowolne okno graficzne (także zbiór okien) można zamknąć poleceniem `close` akceptującym jako parametr wektor numerów okien do zamknięcia.

Czasami wygodne jest umieszczenie kilku wykresów obok siebie w jednym oknie. Można to uczynić posługując się funkcją `subplot`. Istnieją dwa sposoby wywołania tej funkcji:

`subplot(m,n,p)`

dzieli aktywny rysunek na `m` w poziomie i `n` w pionie części oraz uaktywnia `p`-ty z utworzonych rysunków (`m`, `n`, `p` muszą być naturalne z przedziału $<1,9>$),

`subplot('position',[współrzedne_lewego_dolnego szerokość wysokość])`

tworzy w obrębie aktywnego rysunku nowy układ współrzędnych o podanym położeniu i wymiarach. Parametry te podaje się w postaci ułamków wymiarów okna względem lewego dolnego rogu rysunku (np. `subplot('position',[0.3 0 0.6 0.6])` tworzy układ współrzędnych w prawym dolnym rogu okna na 60% wysokości i szerokości okna)

Skalę wykresu można zmienić poleceniem `axis`. Wymaga ono jednego parametru - czteroelementowego wektora zawierającego zakresy poszczególnych skal [`xmin xmax ymin ymax`]. Argumentem tej funkcji może być również łańcuch znaków zmieniający tryby skalowania (szczegóły patrz User's guide)

„Zatrzymanie” dotychczasowego rysunku w bieżącym oknie uzyskuje się poleceniem `hold on|off`. Funkcja `ishold` zwraca jedynkę gdy tryb jest włączony.

Do opisywania rysunku służą funkcje:

`title('text')`

`xlabel('text')`

`ylabel('text')`

`text(x,y,'text')`

Dodatkowo można umieścić pomocniczą siatkę współrzędnych poleceniem `grid on|off`.

Istnieją też inne rodzaje wykresów: we współrzędnych biegunowych, słupkowy itp.

GRAFIKA TRÓJWYMIAROWA

Wykresy trójwymiarowe wymagają specjalnego przygotowania danych. Do narysowania powierzchni konieczne są trzy macierze X , Y , Z , na podstawie których wyznaczane są współrzędne (x,y,z) poszczególnych punktów powierzchni w przestrzeni: $x=X(ij)$, $y=Y(ij)$, $z=Z(ij)$ (gdzie i,j są indeksami macierzy). Pomocną przy tworzeniu tych macierzy jest funkcja `meshgrid`. Typowy sposób wykorzystania tej funkcji w przypadku powierzchni $z=f(x,y)$ ma postać:

```
[X,Y]=meshgrid(x,y);  
Z=X.^2-Y.^2;
```

W pierwszym kroku powstają macierze X i Y złożone z odpowiednio powielonych wektorów x i y , a następnie budowana jest macierz Z będąca wartościami funkcji $z=f(x,y)$ dla poszczególnych wartości argumentów (stąd operacje tablicowe).

Tak przygotowane dane mogą być wykreślone przy pomocy jednej z dostępnych funkcji:

```
mesh(X,Y,Z)  
meshc(X,Y,Z)  
meshz(X,Y,Z)  
surf(X,Y,Z)  
surfc(X,Y,Z)  
surfl(X,Y,Z)  
waterfall(X,Y,Z)
```

Funkcja `view` pozwala **na zmianę miejsca, z którego oglądamy rysunek**. Jej parametrami mogą być `az` i `el` (azymut i elewacja) lub `x,y,z` (współrzędne punktu obserwacji). Można też wykorzystać gotowe dane wpisując jako parametr 2 (obserwacja dwuwymiarowa), 3 (standardowa obserwacja trójwymiarowa).

Zmiana sposobu kolorowania powierzchni jest możliwa przy pomocy polecenia `shading`:

```
shading flat (domyślny)  
shading interp  
shading faceted
```

Opisywanie wykresów trójwymiarowych jest podobne jak w 2D tylko z uwzględnieniem trzeciej współrzędnej.

WYKRESY POZIOMICOWE

Wykres poziomicowy można uzyskać za pomocą funkcji `contour(X,Y,Z)`. Dodatkowym parametrem może być `n`-liczba poziomic, lub wektor `v` z wysokościami dla kolejnych poziomic.

Jeżeli podany zostanie parametr wyjściowy tj.;

`c=contour(X,Y,Z,v)`

to funkcja zwróci macierz poziomic wykorzystywaną przez funkcję `clabel(c)` do opisu tych poziomic na wykresie.

OBIEKTOWY SYSTEM GRAFICZNY

Funkcje graficzne w MATLABie są funkcjami **wysokiego poziomu**. Tworzą gotowe rysunki **nie dając zazwyczaj użytkownikowi możliwości ich modyfikacji**. Każdy fragment rysunku stanowi jednak pewien obiekt graficzny i ma przypisany swój unikatowy *identyfikator* (handle) będący liczbą rzeczywistą.

Każdy obiekt zawiera strukturę danych – rekord, w którym przechowywane są jego parametry. Pola rekordu są własnościami obiektu czyli zmiennymi (liczbami, macierzami, łańcuchami znaków, zmiennymi typu wyliczeniowego) określającymi jakiś parametr jego wyglądu, sposób działania lub powiązania z innymi obiektami. Każdy typ obiektu posiada odrębny zestaw własności (pewne własności są wspólne dla wszystkich typów - patrz dalej)

Obiekty powiązane są ze sobą w sposób hierarchiczny (drzewiasty) i obiekt ma zawsze jednego przodka i dowolną ilość potomków. Korzeniem drzewa (root) jest ekran komputera i nie ma przodka, a jego identyfikatorem jest liczba 0. Jego potomkami są okna graficzne – rysunki (figures), te z kolei mają potomków w postaci układów współrzędnych (axes) oraz elementów graficznego systemu komunikacji z użytkownikiem, tj. przycisków suwaków menu itp. (tworzonych przez funkcje uicontrol i uimenu). Potomkami układów mogą być linie, powierzchnie teksty, obrazy i płyty. **Taka hierarchia musi być przestrzegana.**

Operacje na obiektach można podzielić na dwie grupy:

- tworzenie i usuwanie obiektów,
- zmiany w rekordzie danych związanych z obiektem

Do odczytywania i zmian własności obiektów służą funkcje **get** i **set**:

get (id) – wyświetla listę własności obiektu i ich wartość
wartość = **get**(id, nazwa_własności) – zwraca wartość wyspecyfikowanej własności.

id – identyfikator obiektu lub wektor identyfikatorów,

nazwa_własności – ciąg znaków (w ' ') zawierający nazwę własności **lub jej jednoznaczny początek**

set (id) – wyświetla listę własności obiektu i ich możliwe wartości
set(id, nazwa_własności, wartość) – zmienia wartość wyspecyfikowanej własności na podaną.

reset(id) – przywraca standardowe własności obiektu.

Do usuwania obiektów służą:

delete(id) – usuwa obiekt *id* wraz z jego wszystkimi potomkami,

close(id) – usuwa obiekt *id*,

close - zamyka aktywne okno graficzne,

clf – czyści aktywne okno graficzne,

cla – czyści obiekty z aktywnego układu współrzędnych.

Do tworzenia obiektów służą oprócz funkcji wysokiego poziomu również funkcje o nazwach identycznych jak nazwy typów obiektów. Wszystkie funkcje zwracają identyfikatory tworzonych obiektów.

id=figure – tworzy okno rysunku i zwraca jego identyfikator,

`id=axes('position',[lewy,dolny,szerokość,wysokość])` – tworzy układ współrzędnych,
`id=line(x,y,z,nazwa_własność1, wartość1, nazwa_własność2, wartość2,...)` -
`id=text(x,y,z,nazwa_własność1, wartość1, nazwa_własność2, wartość2,...)`
`id=patch(x,y,z,c,nazwa_własność1, wartość1, nazwa_własność2, wartość2,...)`
`id=surface(x,y,z,c,nazwa_własność1, wartość1, nazwa_własność2, wartość2,...)`
`id=image(x,y,c)`

Obiekty aktywne:

Obiektem aktywnym jest obiekt w obrębie którego ostatnio kliknięto myszką. W danej chwili tylko jeden obiekt może być aktywnym.

`id=gco` – zwraca identyfikator aktywnego obiektu w aktywnym rysunku,
`id=gcf` – podaje identyfikator aktywnego rysunku,
`id=gca` – podaje identyfikator aktywnego układu współrzędnych,

GRAFICZNY SYSTEM KOMUNIKACJI Z UŻYTKOWNIKIEM

Koncepcja graficznego systemu komunikacji z użytkownikiem polega na realizacji dwóch zasad:

- budowaniu wyglądu aplikacji z prostych gotowych elementów mających intuicyjną interpretację i sugestywny wygląd,
- tworzeniu aplikacji o działaniu sterowanym zdarzeniami.

Ad.1. Korzysta się z gotowych obiektów typu przyciski, suwaki, przełączniki.

Ad.2. Każdy taki obiekt ma własność o nazwie 'CallBack', pod którą podstawia się nazwę procedury która ma być wykonana po wystąpieniu dopuszczalnej dla danego obiektu akcji. Ta procedura to zazwyczaj m-plik.

Tworzenie elementów graficznego systemu komunikacji jest oparte na dwóch funkcjach:

`id=uicontrol(idf,nazwa_własność1, wartość1, nazwa_własność2, wartość2,...)`

– tworzy w rysunku *idf* nowy obiekt, którego rodzaj określa się we własności 'Style', a w kolejnych parach (własność – wartość) określa się jego cechy.

`id=uimenu(idf,nazwa_własność1, wartość1, nazwa_własność2, wartość2,...)`

– tworzy w głównym menu rysunku *idf* dodatkowe menu lub jeżeli *idf* jest identyfikatorem istniejącego menu to tworzy dla niego podmenu. Nazwę tworzonego menu określa się we własności 'label', a w kolejnych parach (własność – wartość) określa się inne cechy.

Wszystkie powstałe w ten sposób elementy są potomkami obiektów typu *figure*. Ich własności mogą być zmieniane tak samo jak innych obiektów przy użyciu funkcji `get` i `set`.