

Introducere în Matlab

1 Generalități

- Matlab-ul (denumirea vine de la “matrix laborator”) este un mediu computațional și un limbaj de programare, dezvoltat de firma The Math Works Inc..
- Elementul de bază în Matlab este matricea (scalar, vector, matrice sau tablouri multi-dimensionale).
- Câteva dintre domeniile în care este preferat limbajul de programare Matlab sunt analiza datelor, statistică, analiză numerică, procesarea semnalelor și simulare, datorită facilităților furnizate de calculul matriceal și reprezentările grafice.
- Matlab-ul conține biblioteci de funcții Matlab (fișiere-M), numite TOOLBOX-uri. Acestea permit dezvoltarea mediului de programare de la o versiune la alta, pentru a rezolva probleme din domenii variate.
- Structural, Matlab-ul este realizat sub forma unui nucleu de bază, cu interpretor propriu, în jurul căruia sunt construite toolbox-urile.

2 Modul de lucru în Matlab

- Putem scrie comenzi în mod interactiv în linia de comandă, caz în care fiecare linie este prelucrată imediat și rezultatele sunt afișate.

Exemplu simplu – definirea unei variabile:

A se vizualiza în acest sens Figura 1.

Observații:

1. Matlab-ul reține comenzile folosite și valorile variabilelor create în timpul unei sesiuni.

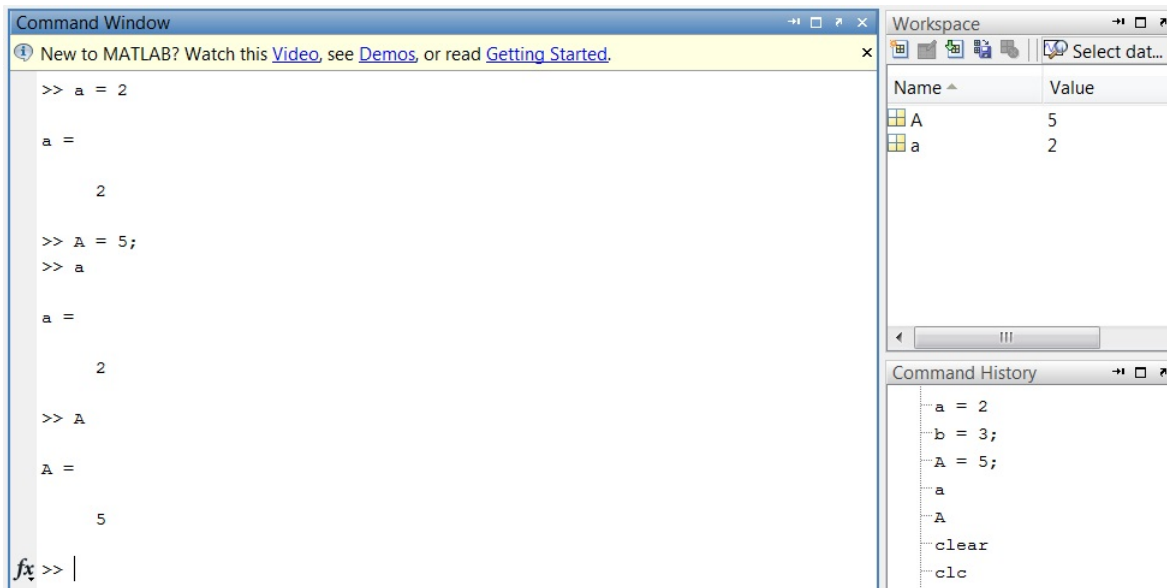


Figura 1: Exemplu: definirea unei variabile în Matlab

2. Variabilele sunt reținute în zona de memorie a Matlab-ului numită *workspace*. Valorile acestor variabile pot fi aflate tastând în linia de comandă numele variabilei fără a folosi vreun semn de punctuație la sfârșitul acesteia.
 3. Trebuie reținut faptul că Matlab-ul este *case sensitive*, deci *Temp*, *temp* sau *TEMP* reprezintă variabile diferite.
 4. În Matlab, comenzile utilizate apar într-o fereastră separată numită *command history*. Acestea pot fi reutilizate sau reeditate în linia de comandă folosind săgețile.
- Putem scrie instrucțiunile în fișiere-M (adică fișierele ce conțin instrucțiuni Matlab; se numesc astfel deoarece au extensia “.m”). Un fișier-M constă dintr-o succesiune de instrucțiuni Matlab; având posibilitatea de a apela alte fișiere-M și a apelării recursive. Un program Matlab poate fi scris sub forma fișierelor “script” sau a fișierelor “funcție”.

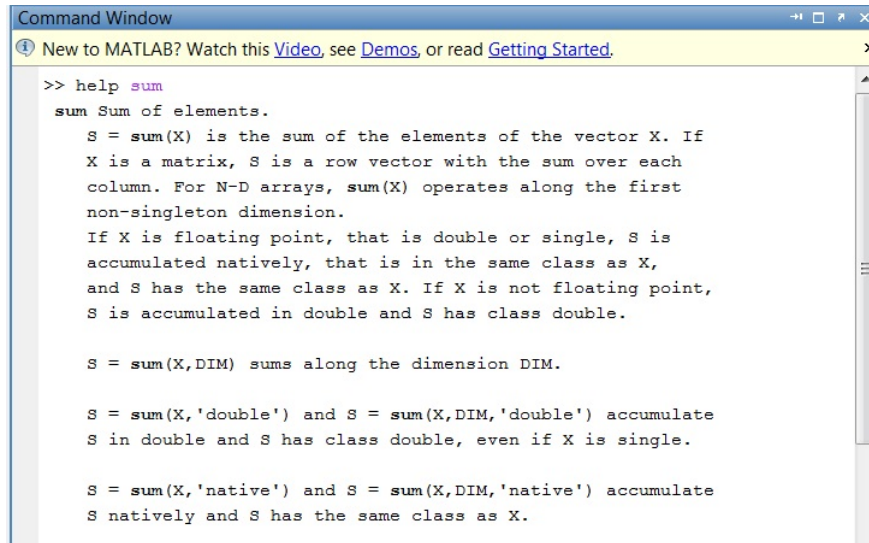
3 Utilizarea help-ului în Matlab

Informații referitoare la funcții Matlab pot fi obținute:

- *în linia de comandă*: tastăm

```
>> help nume
```

unde *nume* este denumirea unei funcții; în linia de comandă vor apărea informațiile de care avem nevoie despre funcția căutată. Se furnizează de asemenea și trimiteri către alte funcții înrudite. Folosirea comenzii *help* este ilustrată în Figura 2.



```
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

>> help sum
sum Sum of elements.

S = sum(X) is the sum of the elements of the vector X. If
X is a matrix, S is a row vector with the sum over each
column. For N-D arrays, sum(X) operates along the first
non-singleton dimension.

If X is floating point, that is double or single, S is
accumulated natively, that is in the same class as X,
and S has the same class as X. If X is not floating point,
S is accumulated in double and S has class double.

S = sum(X,DIM) sums along the dimension DIM.

S = sum(X,'double') and S = sum(X,DIM,'double') accumulate
S in double and S has class double, even if X is single.

S = sum(X,'native') and S = sum(X,DIM,'native') accumulate
S natively and S has the same class as X.
```

Figura 2: Exemplu: folosirea comenzii help

- *din meniu*: fişierele help pot fi accesate şi folosind help-ul din meniu. Informaţiile pot fi obţinute folosind index-ul.

4 Matrice, vectori şi scalari. Declaraţii şi variabile

- *Definirea unei matrice* se realizează ținând cont de următoarele reguli:
 - Elementele unei linii trebuie separate prin blank-uri sau virgule;
 - Liniile se separă prin punct-virgulă “;” ;
 - Elementele matricei sunt cuprinse între paranteze drepte “[]”.

Obs: Indicii matricei încep de la 1.

Exemplu:

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

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

- Notăția $A(i, j)$ semnifică elementul din matricea A aflat la intersecția liniei i cu coloana j .

Exemplu:

```
>> A(2,1)
```

```
ans =
     4
```

- Afișarea unei linii/coloane fără a parcurge linia/coloana respectivă se realizează folosind operatorul “:”.

Example:

```
>> A(2,:)
```

```
ans =  
    4    5    6
```

```
>> A(:,1)
```

```
ans =  
    1  
    4  
    7
```

- Notăția $A(I, J)$, unde I, J sunt vectori-linie, reprezintă submatricea/blocul formată/format din elemente aflate la intersecția liniilor date în vectorul I cu coloanele date în vectorul J .

Example:

```
>> A([1 3], [2 3])
```

```
ans =  
    2    3  
    8    9
```

```
>> A([1 3], [3 2])
```

```
ans =  
    3    2  
    9    8
```

- Notăția $A(:)$ determină introducerea elementelor matricei A într-un vector coloană prin concatenarea coloanelor matricei.

Exemplu:

```
>> A(:)
```

```
ans =  
    1  
    4  
    7  
    2  
    5  
    8  
    3  
    6  
    9
```

5 Generarea automată a vectorilor

Pentru a genera un vector cu elemente în progresie aritmetică și ordonate crescător, Matlab-ul oferă două metode:

- Dacă se cunosc valoarea minimă ($xmin$) și valoarea maximă ($xmax$) din vector și pasul/diferența în modul (pas) dintre două elemente consecutive, se generează vectorul cu instrucțiunea:

```
>> x = xmin : pas : xmax
```

Dacă pasul e negativ atunci e necesar ca $xmin > xmax$. Dacă se omite specificarea valorii pasului, atunci acesta va fi luat implicit egal cu 1.

Exemple:

```
>> x = 1 : 5
```

```
x =  
    1    2    3    4    5
```

```
>> x = -4 : 2 : 6
```

```
x =  
   -4   -2    0    2    4    6
```

- Dacă se cunosc valoarea minimă ($xmin$) și valoarea maximă ($xmax$) din vector și numărul de elemente (N) ale vectorului generat cu pas liniar, atunci se folosește instrucțiunea:

```
>> x = linspace(xmin, xmax, N)
```

Dacă valoarea lui N este omisă, implicit se va lua 100.

Exemplu:

```
>> x = linspace(1, 10, 5)
```

```
x =  
    1.0000    3.2500    5.5000    7.7500   10.0000
```

6 Matrice speciale

Funcțiile ce permit crearea unor matrice speciale sunt de forma:

```
>> nume_functie(n) % returnează o matrice pătratică de ordinul n  
sau
```

```
>> nume_functie(m, n) % returnează o matrice având m linii și n coloane
```

Exemple de funcții ce returnează matrice speciale sunt prezentate mai jos:

- `>> eye(m, n)` % returnează o matrice identitate

Exemplu:

```
>> eye(2,3)
```

```
ans =  
    1    0    0  
    0    1    0
```

- `>> ones(m,n)` % returnează o matrice cu toate elementele egale cu 1

Exemplu:

```
>> ones(2,3)
```

```
ans =  
    1    1    1  
    1    1    1
```

- `>> zeros(m,n)` % returnează o matrice cu toate elementele egale cu 0

Exemplu:

```
>> zeros(2,3)
```

```
ans =  
    0    0    0  
    0    0    0
```

- `>> rand(m,n)` % returnează o matrice având ca elemente valori de selecție asupra unei variabile aleatoare repartizate uniform pe intervalul [0, 1]

Exemplu:

```
>> rand(2,3)
```

```
ans =  
    0.8147    0.1270    0.6324  
    0.9058    0.9134    0.0975
```

- `>> randn(m,n)` % returnează o matrice având ca elemente valori de selecție asupra unei variabile aleatoare repartizate normal de medie 0 și dispersie 1

Exemplu:

```
>> randn(2,3)
```

```
ans =  
   -0.4336    3.5784   -1.3499  
    0.3426    2.7694    3.0349
```

- `>> g = []` % returnează o matrice g de dimensiune 0, dar care există în spațiul de memorie

Exemplu:

```
>> g = [];
```

```
>> g = [g 1]
```

```
g =
```

```

1
>> g = [g 2]
g =
1 2

```

7 Operații cu matrice

- **Operațiile uzuale cu matrice** (din algebra liniară) sunt simbolizate cu semnele grafice: $+$, $-$, $*$, $/$, $^$, $'$, și se efectuează după regulile cunoscute din calculul matriceal.

- $A/B = A * B^{-1} = A * \text{inv}(B)$
- $A^p = A * A * \dots * A$, unde A apare de p ori
- A' este matricea transpusă a lui A

Aplicație:

Fie $A = [1 \ 2; 3 \ 4]$, $B = [5 \ 6; 7 \ 8]$. Să se calculeze:

$C = A/B$, $D = A^2$, $E = A + 2$, $F = B * 4$, $G = A - 2$, $H = A'$.

- **Operațiile “element cu element”** sunt operații aritmetice (înmulțire, împărțire, ridicare la putere, etc.) între elementele situate la aceeași poziție în cele două matrice (văzute ca operanzi).
 - Pentru efectuarea operațiilor “element cu element” se folosesc aceiași operatori ca în operațiile cu scalari, precedați de semnul punct “.”.
 - Pentru a putea fi efectuate operații “element cu element”, trebuie ca dimensiunile matricelor cu care se operează să fie identice.
 - Dacă unul dintre operanzi este un scalar, acesta operează cu fiecare element al matricei și nu mai este necesar semnul “.” în fața operatorului uzual.
 - În cazul operațiilor de adunare și scădere, operatorul uzual nu va mai fi precedat de punct.

Aplicație:

$I = A .* B$, $J = A ./ B$, $L = A.^2$

8 Operatori relaționali și logici

Operatori relaționali

- $>$, $<$, $=$, $<=$, $>=$, $==$, \sim

Example:

```
>> [2 5 1 7] == [2 4 8 7]
```

```
ans =  
     1     0     0     1
```

```
>> [2 5 1 7] ~ [2 4 8 7]
```

```
ans =  
     0     1     1     0
```

Operatori logici

- and, or, not, xor
- find, all, any, true, false

9 Reprezentări grafice elementare

`>> plot(x, y)` plotează vectorul x versus vectorul y (adică reprezintă punctele $(x(i), y(i))$, $i = 1, \dots, \text{length}(x)$ și unește două puncte consecutive prin linii drepte). Vectorii trebuie să aibă aceeași lungime.

`>> plot(x, y, 'marker color')`, unde “marker” este un simbol, iar “color” o culoare, plotează punctele $(x(i), y(i))$, $i = 1, \dots, \text{length}(x)$ fără a le uni/interpola.

`>> plot3(x, y, z, 'marker color')` reprezintă grafic punctele tridimensionale $(x(i), y(i), z(i))$, $i = 1, \dots, \text{length}(x)$.

`>> subplot(m, n, i)` împarte ecranul în $m \times n$ ferestre și introduce plotul curent în fereastra i .

Observații:

1. Axele sunt scalate implicit astfel încât să se permită vizualizarea întregului grafic. Pentru a avea mai mult control asupra graficului se poate utiliza funcția *axis*.
2. De fiecare dată când este apelată funcția *plot*, din figură sunt eliminate (șterse) informațiile anterioare. Pentru a adăuga informații noi unei figuri (precum legende și etichete ale axelor) sau grafice suplimentare se utilizează funcția *hold on* după prima apelare a funcției *plot*.

Aplicație:

Fie codul Matlab următor:

```
x = -pi:0.1:pi;  
y = sin(x);  
subplot(2,2,1);  
plot(x,y,'r');  
xlabel('x');  
ylabel('sin(x)');  
title('Graficul functiei sin(x)');
```

Realizați în celelalte 3 ferestre graficele funcțiilor $\cos(x)$, $\arcsin(u)$, $\arccos(u)$ ($u = -1:0.01:1$) utilizând o varietate cât mai mare de linii și culori.

10 Fișiere script și fișiere funcție

Fișiere script

- Un fișier “script” este un fișier-M care conține o secvență de instrucțiuni Matlab.
- Prin apelarea numelui fișierului în linia de comandă, se execută secvența de instrucțiuni conținută în acesta, executându-se fiecare instrucțiune ca și cum ar fi tastată interactiv în linia de comandă.
- Instrucțiunile unui fișier script au acces la zona de memorie principală (denumită “workspace”), iar după execuția completă a unui fișier script, variabilele definite în fișier rămân în zona de memorie (adică în “workspace”).
- Folosirea fișierelor script este recomandată pentru executarea unor experimente.

Fișiere funcție

- Forma generală a primei linii a unui fișier funcție este:
`function [paramOut1,..., paramOutm] = numeFuncție(paramIn1,..., paramInn)`
unde:
 - function - este cuvântul cheie care definește fișierul ca fișier funcție (prezența lui este obligatorie);
 - numeFuncție - este numele funcției; fișierul trebuie salvat cu acest nume (numele funcției nu trebuie să coincidă cu a unui fișier existent);

- `paramIn1, ..., paramInn` - reprezintă parametrii de intrare
 - `paramOut1, ..., paramOutm` - reprezintă parametrii de ieșire. Dacă funcția nu are parametri de ieșire, parantezele drepte și semnul egal pot să nu mai apară.
- Variabilele definite și manipulate în interiorul fișierului funcție sunt recunoscute doar în cadrul funcției, deci fișierele funcție au o zonă de memorie proprie.
 - Comunicarea de informații dintre zona de memorie alocată funcției și zona de memorie principală se face prin intermediul parametrilor de intrare și al parametrilor de ieșire.
 - Fișierele funcție sunt utilizate pentru extinderea Matlab-ului, prin crearea unor noi toolbox-uri.
 - Este recomandat ca pe liniile care urmează imediat după antetul funcției, să se introducă un comentariu care să dea informații referitoare la funcția definită. Astfel, atunci când un alt utilizator dorește să afle informații despre fișierul respectiv, poate tasta în linia de comandă:


```
>> help numeFuncție
```

 și va apărea în linia de comandă comentariul introdus în fișier.

Aplicații:

1. Scrieți o funcție cu numele *medie* care primește ca argumente 2 numere *a* și *b* și returnează media aritmetică, media geometrică și media armonică a numerelor *a* și *b*.
2. Scrieți o funcție cu numele *medieGen* care primește ca argumente un vector și returnează media aritmetică și media geometrică a componentelor.
3. Scrieți în funcțiile definite la punctele 1 și 2 comentarii referitoare la scopul lor.
4. Fie $v = [1\ 2\ 3\ 4]$ și fie $A = [1\ 2\ 3\ 4; 5\ 6\ 7\ 8; 9\ 10\ 11\ 12]$. Cu ajutorul comenzii `>> help numeFuncție` înțelegeți ce returnează funcțiile de mai jos:


```
w1 = max(v), B1 = min(A)
w2 = mean(v), B2 = mean(A)
w3 = median(v), B3 = median(A)
w4 = sum(v); w5 = cumsum(A)
w6 = prod(v); w7 = cumprod(A)
w8 = sort(v), [w9, w10] = sort(v), B4 = sort(A)
[w11, w12] = size(A), B5 = length(A)
```

11 Instrucțiuni și funcții de control în programe

Instrucțiunile de control logic în Matlab sunt următoarele:

- **if** instrucțiune pentru executarea condiționată a unui set de instrucțiuni;
- **else** clauză asociată cu “if”;
- **elseif** clauză asociată cu “if”;
- **for** instrucțiune pentru realizarea ciclurilor cu un număr determinat de repetări;
- **while** instrucțiune pentru realizarea ciclurilor pe baza unei condiții logice;
- **break** instrucțiune pentru terminarea forțată a unui ciclu;
- **return** instrucțiune pentru returnarea execuției în modulul apelant;
- **error** instrucțiune ce permite afișarea unui mesaj de eroare;
- **end** instrucțiune pentru încheierea instrucțiunilor “for”, “while” și “if”.

În Matlab este indicată utilizarea programării vectoriale (i.e. prelucrarea întregului tablou, fără a folosi instrucțiuni care realizează cicluri pentru a opera asupra unui element) din motive de eficiență.

În cele ce urmează sunt prezentate sintaxele instrucțiunilor ce pot fi folosite în Matlab.

- Forma generală a unei instrucțiuni “if” este următoarea:

```
if expresieLogica1
    secventaInstructiuni1
elseif expresieLogica2
    secventaInstructiuni2
elseif expresieLogica3
    secventaInstructiuni3
.....
else
    secventaInstructiuniN
end
```

- Instrucțiunea “for” permite repetarea de un număr determinat de ori a unui grup de instrucțiuni și are următoarea structură generală:

```
for index = initial : pas : final

    secventaInstruciuniRepetate

end
```

- Instrucțiunea “while” permite executarea repetată a unui grup de instrucțiuni de un număr nedeterminat de ori sub controlul unei condiții logice. Forma generică este:

```
while expresie

    secventaInstruciuni

end
```

Instrucțiunile din bucla “while” sunt executate atâta timp cât *expresie* are elemente nenule. În general, *expresie* are o singură valoare TRUE sau FALSE.

- Instrucțiunea “break” se utilizează pentru a ieși dintr-o buclă înainte ca aceasta să se fi terminat. Instrucțiunea “break” determină oprirea execuției ciclurilor “for” și “while”. În cazul unor cicluri imbricate, “break” determină ieșirea din ciclul cel mai interior. Se apelează cu sintaxa:

```
break
```

- Instrucțiunea “return” determină o ieșire normală din fișierul-M către funcția care l-a apelat sau către tastatură. Se apelează cu sintaxa:

```
return
```

- Instrucțiunea “error” permite afișarea unor mesaje la întâlnirea unei erori. Se apelează cu sintaxa:

```
error('mesaj')
```

Aplicații:

1. Scrieți codul Matlab care trasează graficul funcției următoare:

$$f : [-10, 10] \rightarrow R, f(x) = \begin{cases} 2x + 8, & \text{dacă } x \leq 2 \\ 3x^2, & \text{dacă } x > 2 \end{cases}$$

2. Să se genereze o matrice A cu n linii și $n + 1$ coloane ale cărei elemente sunt:

$$A(i, j) = \begin{cases} 2, & \text{dacă } i = j \\ -1, & \text{dacă } |i - j| = 1 \\ 0, & \text{în rest} \end{cases}$$

3. Să se scrie un program, utilizând o buclă “while”, care calculează suma elementelor vectorului $x = [2 \ -3 \ 8 \ 3 \ 2 \ 1 \ -5 \ 9 \ 7]$ până când se întâlnește un număr mai mare ca 8.

Bibliografie

- [Ghinea, Fireteanu (2007)] M. Ghinea, V. Fireteanu (2007), *Matlab. Calcul numeric. Grafică. Aplicații.*, Editura Teora
- [Nabney (2002)] T. Nabney (2002), *Netlab. Algorithms for Pattern Recognition.*, Ed. Springer