

# Metode Numerice

## Curs 1: Introducere în Matlab. Instrucțiuni și comenzi Matlab

Octavia-Maria BOLOJAN

Babeș-Bolyai University of Cluj-Napoca  
Faculty of Mathematics and Computer Science  
octavia.nica@math.ubbcluj.ro

University of Oradea  
Faculty of Electrical Engineering and Information Technology  
obolojan@uoradea.ro

5 Octombrie 2018

- 1 Metode Numerice - Conținutul cursului
- 2 Cerințe și evaluare
- 3 Curs 1: Introducere în Matlab. Instrucțiuni și comenzi Matlab
- 4 Bibliografie

1. Elemente fundamentale de programare în Matlab
  - Introducere în Matlab
  - Instrucțiuni și comenzi Matlab. Funcții Matlab
  - Grafică în Matlab: reprezentări 2D,3D
2. Erori. Noțiuni introductive
3. Metode Numerice pentru rezolvarea sistemelor algebrice liniare
4. Aproximarea funcțiilor
5. Rezolvarea ecuațiilor neliniare
6. Derivarea și integrarea numerică

- Disciplina Metode Numerice: 2 ore de Curs, respectiv 2 ore de laborator în fiecare săptămână
- Nota finală:
  - Verificare pe parcurs 1: în săpt. 7/8  $\rightarrow$  30%
  - Verificare pe parcurs 2: în săpt. 13/14  $\rightarrow$  30%
  - Evaluare Laborator (activitate, lucrări practice+Examen final de laborator)  $\rightarrow$  40%
- În ceea ce privește VP1&2, nota examen scris = minim nota 5.00 pentru a putea fi promovat; analog în ceea ce privește EL

# Curs 1: Introducere în Matlab. Instrucțiuni și comenzi Matlab

- La deschiderea Matlab -> o fereastră în care sunt mai multe subferestre:

- **Current Directory/Folder**
- **Command History**
- **Command Window**
- **Workspace, Command History**

(Acesta este default-ul, se poate schimba sau reveni, de la Toolbar-ul de sus, de la Desktop -> Desktop Layout sau Layout - în funcție de versiunea instalată)

- Fereastra Command Window este interactivă, la promptul special Matlab-ului, ">>", putem face calcule, atribuiri, etc., al căror rezultat este afișat imediat

```
>> 2+3
```

care produce rezultatul

```
ans =  
5
```

Cum nu i-am dat niciun nume (nicio atribuire) rezultatului, Matlab l-a numit **ans** de la *answer* (*răspuns*). Dacă vrem să-i atribuim un nume:

```
>> a=2+3  
a =  
5
```

o simplă atribuire:

```
>> x=1  
x =  
1
```

Motivul pentru care apare de două ori  $x = 1$  este că Matlab-ul afișează pe ecran rezultatul oricărei execuții (chiar dacă este doar o simplă atribuire), dacă nu se pune simbolul ";" la sfârșit. Dacă nu dorim această afișare, punem ";" la sfârșit:

```
>> x=1;  
>>
```

- numele **MATLAB** vine de la **MATrix LABoratory** → lucrează foarte bine și ușor cu matrice
- Un scalar este, de fapt, interpretat ca o matrice  $1 \times 1$
- O matrice se definește cu paranteze pătrate `[,]`, elementele unei linii fiind despărțite de virgulă sau space, iar pentru a trece la următoarea linie folosim `","`
- Atenție la dimensiuni!!! altfel obținem eroare:

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

```
A =
```

```
1 2 3
```

```
4 5 6
```

```
7 8 9
```

```
0 0 0
```

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

```
??? Error using ==> vertcat
```

All rows in the bracketed expression must have the same number of columns.

- De asemenea, de menționat, când e nevoie de paranteze într-o expresie, se folosesc doar paranteze rotunde, simple, cele pătrate fiind rezervate pentru definirea matricelor



O matrice poate fi uneori, definită mai simplu, fără a lista fiecare element, ci doar

initialvalue : step : endvalue.

Simbolul ":" are semnificația "de la" sau "până la" sau "cu pasul":

```
>> x=1:2:10
```

```
x =
```

```
1 3 5 7 9
```

- Pasul poate și negativ, iar dacă nu se precizează o valoare, default-ul este 1:

```
>> x=5:-0.5:2
```

```
x =
```

```
5.0000 4.5000 4.0000 3.5000 3.0000 2.5000 2.0000
```

```
>> y=1:7
```

```
y =
```

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

Exemplu de definire a elementelor unei matrice folosind simbolul ":":

```
>> A=[1:3;4:-2:0; 1:3:8; 0:2]
```

```
A =
```

```
1 2 3
```

```
4 2 0
```

```
1 4 7
```

```
0 1 2
```

→ pentru a obține o submatrice (inclusiv un element) al unei matrice deja definite:

```
>> A(1:3,2:3)
```

```
ans =
```

```
2 3
```

```
2 0
```

```
4 7
```

conține matricea obținută prin intersecția zonei corespunzătoare liniilor 1; 2; 3 cu coloanele 2; 3 ale matricei A

```
>> A(:,2:3)
```

```
ans =
```

```
2 3
```

```
2 0
```

```
4 7
```

```
1 2
```

conține toate liniile și coloanele 2; 3 ale matricei A, iar

```
>> A(4,2)
```

```
ans =
```

```
1
```

este elementul de pe linia a 4-a și coloana a 2-a a matricei A.

- $A(:, :)$  va genera același lucru cu matricea A

Tot ceea ce am făcut până acum în Command Window se va șterge în momentul în care se încheie sesiunea Matlab

- Cum lucrăm cu fișiere, astfel încât să putem salva ceea ce am lucrat?
- Fișierele Matlab se numesc **M-files** și sunt de două tipuri: **fișiere M de tip script** (sau fișiere de comenzi) și **fișiere M de tip funcție**

**Fișierele M de tip script** → nu au nici un argument de intrare sau ieșire și operează asupra variabilelor din spațiul de lucru

**Fișierele M de tip funcție** → conțin o linie de definiție `function` și pot accepta argumente de intrare și returna argumente de ieșire, iar variabilele lor interne sunt locale funcției (exceptând cazul în care sunt declarate `global`)

- Un fișier script permite memorarea unei secvențe de comenzi care sunt utilizate repetat sau vor fi necesare ulterior

- Din colțul de sus, de la File, select **New-> M-File**
- Se deschide o nouă fereastră Editor - Untitled (deocamdată nu are nume fișierul!)
  - Într-un fișier simplu, listăm toate comenzile pe care vrem să le execute programul la lansarea acestuia (fără o linie de început sau de sfârșit a programului). De exemplu:

```
x=1;  
y=2;  
z=x+y  
t=x*y
```

Observăm că variabilelor  $x$  și  $y$  (când le-am atribuit valori) le-am pus ";" la sfârșit, pentru a nu le mai fi afișate valorile pe ecran (puteam lăsa să le afișeze), iar la definirea variabilelor noi  $z$  și  $t$  nu le-am mai pus ";" la sfârșit astfel încât să le putem vedea valorile

- Salvăm fisierul, de la **Save** din File, sau dând click pe dischetă
- by default, Matlab-ul își salvează M- files-urile în directorul **work** sau **MATLAB** (în funcție de versiunea instalată)
- după ce i s-a dat un nume, fișierul nou creat are extensia **.m**

## Sub ce nume se poate salva un fișier M-file?

- cu orice nume care NU E OCUPAT deja în Matlab (funcțiile uzuale, *sin*, *cos*, *log*, *abs*, *exp*, *input*, *max*, *min*, etc)
  - numele poate conține numere, dar NU poate începe cu, sau conține doar, numere.
- se poate de ex. "l1" sau "lab-met-num-1", dar nu "1" sau "1lab"
- acest lucru se face pentru a nu se confunda cu o variabilă atunci când se lansează execuția
- ATENȚIE!!! dacă se salvează cu un nume care este deja o funcție în Matlab, NU VA DA EROARE, se va salva, dar se va schimba funcția respectivă, ceea ce poate conduce la erori de compilare etc.

## Cum se lansează execuția?

- fie din fereastra editorului, de sus de la Debug -> Run, fie din Command Window, pur și simplu tastând numele fișierului, fără extensia .m, după prompter
- Cea de-a doua metodă e mai simplă, dar trebuie avut grijă ca în Command Window, sus, la Current Directory (unde default-ul e C:\MATLAB\work sau C:\Users\User\Documents\MATLAB, în funcție de versiunea instalată) să fie directorul corect, unde s-a salvat fișierul (se caută de la butonul galben din dreapta)

```
>> lab1mn
```

```
z =
```

```
3
```

```
t =
```

```
2
```



Altfel, Matlab-ul nu-l găsește

```
>> lab1mn
```

```
??? Undefined function or variable 'lab1mn'
```

- Observație: acesta este motivul pentru care numele fișierului nu putea fi un număr sau o expresie care începe cu o cifră/număr, deoarece la tastarea numelui în Command Window, Matlab-ul l-ar fi confundat cu un simplu număr, cu o variabilă sau cu o operație pe care n-ar fi înțeles-o
- dacă dorim să executăm programul respectiv și pentru alte valori ale lui  $x$  și  $y$ , acestea se pot cere de la user, în interiorul programului, cu comanda

input

- Se poate vedea cum funcționează aceasta, tastând în Command Window

```
>> help input
```

Edităm în fișier:

```
x=input('give the value for x= ');  
y=input('give the value for y= ');  
z=x+y  
t=x*y
```

pe ecran apare (se așteaptă valori de la tastatură)

```
>> lab1mn  
give the value for x= 2  
give the value for y= 3  
z =  
    5  
t =  
    6
```

Dacă la input dorim o valoare nenumerică, un șir de caractere, se mai adaugă opțiunea 's' (string of characters), după cum se vede în help input:

```
>> help input
```

INPUT Prompt for user input.

R = INPUT('How many apples') gives the user the prompt in the text string and then waits for input from the keyboard. The input can be any MATLAB expression, which is evaluated, using the variables in the current workspace, and the result returned in R. If the user presses the return key without entering anything, INPUT returns an empty matrix.

R = INPUT('What is your name','s') gives the prompt in the text string and waits for character string input. The typed input is not evaluated; the characters are simply returned as a MATLAB string.

The text string for the prompt may contain one or more '\n'. The '\n' means skip to the beginning of the next line. This allows the prompt string to span several lines. To output just a '\', use '\\'.  
See also keyboard.

Reference page in Help browser

doc input

!!!De reținut: în general, se tastează *help topic* și apar informațiile necesare

- Celălalt tip de M-file e o funcție Matlab. Aceasta se definește exact pe formatul în care sunt definite funcțiile intrinseci ale Matlab-ului.

- Deschidem un nou M-file.

- Aici, de la început trebuie specificat că va fi o funcție:

```
function y=myfun(x)  
y=x+1;
```

Trebuie precizați parametrii de intrare (în cazul nostru, *x*), între paranteze rotunde, parametrii de ieșire (în cazul nostru, *y*), numele pe care îl dăm funcției (în cazul nostru, *myfun*), iar în corpul funcției trebuie atribuite valori tuturor parametrilor de ieșire

- Un fișier de tip funcție se salvează DOAR cu numele pe care l-am dat funcției, altfel Matlab nu-l găsește.
- Așadar, când dorim să salvăm acest fișier, automat va apărea numele *myfun.m*
- Apelul (în Command Window, sau într-un alt M-file) se face în mod obișnuit, cu precizarea, că pentru datele de intrare trebuie să fie atribuite deja valori

```
>> myfun(3)
ans =
4
```

sau se poate atribui valoarea funcției unei variabile (nu neapărat același nume cu cel din M-file)

```
>> x=0;
>> myanswer=myfun(x)
myanswer =
1
>>
```

dar nu se poate simbolic (deci fără o valoare bine definită):

```
>> myfun(y)
??? Undefined function or variable 'y'
```

Pot fi mai multe date de intrare (ca o funcție de mai multe variabile), între paranteze rotunde, sau/și mai multe date de ieșire, între paranteze pătrate (așadar un vector).

Tuturor variabilelor de ieșire trebuie să li se atribuie valori:

```
function [z,t]=myfun(x,y)
z=x+y;
t=x*y;
```

Când se face apelul unei astfel de funcții (cu mai multe date de ieșire), pentru a le vedea pe toate, trebuie făcut apelul complet:

```
>> [a,b]=myfun(2,3)
a =
5
b =
6
```

altfel, cu apel simplu, se va returna DOAR valoarea PRIMEI variabile de ieșire:

```
>> myfun(2,3)
ans =
5
```

# Vectori și matrice în Matlab

Am menționat mai devreme că Matlab-ul lucrează foarte ușor cu vectori și matrice. Astfel, de multe ori se poate evita folosirea unui `for`, se poate lucra direct matriceal. În Command Window, enumerăm câteva exemple:

```
>> x=1:10
```

```
x =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
>> x+2
```

```
ans =
```

```
3 4 5 6 7 8 9 10 11 12
```

```
>> a=[1 2; 3 4]
```

```
a =
```

```
1 2
```

```
3 4
```

```
>> b=[1 0; 0 1]
```

```
b =
```

```
1 0
```

```
0 1
```

```
>> a+b
```

```
ans =
```

```
2 2
```

```
3 5
```

```
>> a*b
```

```
ans =
```

```
1 2
```

```
3 4
```



```
>> a/2  
ans =  
0.5000 1.0000  
1.5000 2.0000
```

Așadar se poate opera direct pe matrice, iar operațiile sunt cele cunoscute. Trebuie avut însă grijă ca dimensiunile matricelor să fie potrivite pentru operația respectivă. Altfel, obținem eroare:

```
>> a+x  
??? Error using ==> plus  
Matrix dimensions must agree.  
  
>> a*x  
??? Error using ==> mtimes  
Inner matrix dimensions must agree.
```

Dacă dorim ca o anumită operație să NU se facă matriceal, ci scalar (termen cu termen, ca adunarea), operația respectivă se definește CU PUNCT în față (*dot operations*).

E vorba despre operațiile care provin din înmulțire, singura care e definită altfel la matrice, decât termen cu termen. Așadar avem "\*" și ".\*" (înmulțire), "^" și ".\*" (ridicare la putere), "/" și "./" (împărțire, adică înmulțire cu inversa).

Exemple:

```
>> x.^2
ans =
1 4 9 16 25 36 49 64 81 100
```

```
>> a.*b
ans =
1 0
0 4
```

Și în cadrul fișierelor, se poate lucra direct matriceal (cu aceeași mențiune, atenție la dimensiunile matricelor și la tipul operațiilor):

```
>> lab1mn
```

```
give the value for x= a
```

```
give the value for y= b
```

```
z =
```

```
2 2
```

```
3 5
```

```
t =
```

```
1 2
```

```
3 4
```

```
sau
```

```
>> [C,D]=myfun(a,b)
```

```
C =
```

```
2 2
```

```
3 5
```

```
D =
```

```
1 2
```

```
3 4
```

```
dar nu
```

```
>> [C,D]=myfun(1:10,1:10)
```

```
??? Error using ==> mtimes. Inner matrix dimensions must  
agree. Error in ==> myfun at 3
```

Întrucât Matlab-ul lucrează foarte bine cu matrice, are câteva matrice speciale, care pot fi generate ușor, de orice ordin:

-`zeros(m, n)` - generează o matrice de zerouri de dimensiunea  $m \times n$ . Dacă dorim doar un vector, una dintre dimensiuni va fi 1 (vector-linie sau vector-coloană). Dacă dăm o singură dimensiune, matricea generată va fi pătratică de acel ordin:

```
>> zeros(3,4)
```

```
ans =
```

```
0 0 0 0
```

```
0 0 0 0
```

```
0 0 0 0
```

```
>> zeros(1,3)
```

```
ans =
```

```
0 0 0
```

```
>> zeros(2,1)
```

```
ans =
```

```
0
```

```
0
```

```
>> zeros(3)
```

```
ans =
```

```
0 0 0
```

```
0 0 0
```

```
0 0 0
```

-ones(m, n) - la fel ca mai sus, doar că valorile sunt de 1, nu 0:

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

```
ans =
```

```
1 1 1
```

```
1 1 1
```

Sunt cazuri în care un vector  $X$  va trebui să conțină distribuția frecvențelor unui set de date, spre exemplu: valoarea 10 apare de 6 ori, valoarea 12 de 15 ori, etc... În loc să tastăm/copiem fiecare valoare de atâtea ori, e mai simplu de folosit matricea **ones**:

```
>> X=[10*ones(1,6), 12*ones(1,15)]
```

```
X =
```

```
Columns 1 through 11
```

```
10 10 10 10 10 10 12 12 12 12 12
```

```
Columns 12 through 21
```

```
12 12 12 12 12 12 12 12 12 12
```

-eye(m,n) - matricea unitate

Dacă dimensiunile corespunzătoare liniilor sau coloanelor nu sunt egale, se completează cu 0 liniile sau coloanele care sunt în plus:

```
>> eye(5)
```

```
ans =
```

```
1 0 0 0 0
```

```
0 1 0 0 0
```

```
0 0 1 0 0
```

```
0 0 0 1 0
```

```
0 0 0 0 1
```

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

```
ans =
```

```
1 0 0
```

```
0 1 0
```

```
>> eye(4,2)
```

```
ans =
```

```
1 0
```

```
0 1
```

```
0 0
```

```
0 0
```

## Comanda fprintf

Deocamdată vom realiza afișarea rezultatelor pe ecran, în Command Window (standard output), așadar nu folosim niciun fid (file identifier). Pentru fiecare variabilă pe care vrem să o afișăm, trebuie să furnizăm un format, cu caracterul "%" în față

Formatul este: "d" pentru întreg, "f" pentru real (float), "e" pentru forma exponențială, "s" pentru string of characters (variabilă non-numerică)

Formatul se consideră între ghilimele simple, ' ', apoi virgulă și înșiruirea variabilelor ce se doresc a fi afișate:

```
>> x=5
```

```
x =
```

```
5
```

```
>> y=0.5
```

```
y =
```

```
0.5000
```

```
>> z=12000
```

```
z =
```

```
12000
```

```
>> fprintf('%d %f %e', x,y,z)
```

```
5 0.500000 1.200000e+004
```

```
>>
```



Formatul poate fi mai explicit, să menționeze nu doar de ce tip sunt datele, ci și din câte cifre, câte înainte, după virgulă, etc. TOT ce apare între ' ' și NU e un format (așadar nu e cu %), apare afișat de asemenea. Astfel, pot fi mesaje, spații, virgule, semne de punctuație, paranteze, etc., tot ce dorim pentru a face display-ul să arate mai bine și mai ușor de citit. De asemenea, vom folosi "\n" pentru newline, "\t" pentru tab (pentru aranjarea datelor una sub cealaltă), etc.,

```
>> fprintf('The answer is: \n %2d, %2.3f, %e\n', x,y,y)
The answer is:
5, 0.500, 5.000000e-001
```

Acest lucru ne va fi foarte folositor la aranjarea datelor sub formă de tabel, interval, etc.

- pentru a afișa rezultatele într-un alt fișier, se precizează fișierul prin fid (file identifier).
- Se deschide (sau creează) acest fișier cu **fopen**

Edităm fișierul în care am lucrat anterior, **lab1mn.m** și mai adăugăm:

```
x=input('give the value for x= ');
y=input('give the value for y= ');
z=x+y;
t=x*y;
fid=fopen('results_file.m','a+');
fprintf(fid, '%%The answers
are:\n%_____\\n');
fprintf(fid, '%%the value of z is %5.3f\\n',z);
fprintf(fid, '%%the value of t is %5.3f\\n',t);
fprintf(fid, '%%%%%%%%%%%%%%\\n\\n');
fclose(fid);
```

care va avea ca urmare crearea fișierului **results\_file.m** cu opțiunea '**a+**', adică va fi creat, se poate citi din el și, dacă fișierul există deja și conține anumite informații, noile rezultate se adaugă la sfârșit (**append**)

Ca urmare a rulării programului, avem:

```
>> lab1mn  
give the value for x= 1.2345  
give the value for y= -72.333
```









În fișierul results\_file.m va apărea:

```
%The answers are:  
%_-----  
%the value of z is -71.099  
%the value of t is -89.295  
%%%%%%%%%
```

## Last remarks

- ❶ În Matlab, comentariile sunt marcate cu simbolul "%" în față
- ❷ Un grup de instrucțiuni pot fi comentate sau decommentate împreună dacă se realizează click dreapta pe porțiunea selectată (grupul de comenzi/instrucțiuni), apoi se alege opțiunea Comment sau Uncomment
- ❸ Lansarea unui program/execuția unui fișier se poate face din Command Window prin tastarea numelui
- ❹ Folosind săgețile up/down de la tastatură, putem naviga/executa din nou comenzile date anterior în Command Window
- ❺ Matlab-ul este case-sensitive; În plus, chiar dacă la tastarea `help topic` comenzile apar cu majuscule, acestea vor funcționa doar dacă sunt tastate cu litere mici
- ❻ Comenzi importante pentru utilizator:
  - `clear all` - șterge variabilele și funcțiile temporare din directorul curent
  - `clear var` - șterge variabila "var" din directorul curent
  - `clf` - creează o nouă fereastră pentru o figură, ștergând figura anterioară
  - `clc` - șterge toate afișările anterioare din Command Window (clear Command Window)

# Bibliografie

-  O. Agratini, I. Chiorean, Gh. Coman, R. Trîmbițaș, *Analiză numerică și teoria aproximării, vol. III*, Presa Universitară Clujeană, 2002.
-  U. M. Ascher, L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia PA, 1998.
-  O.-M. Bolojan, M.-A. Șerban, *Metode numerice. Exerciții și probleme rezolvate în Matlab*, Editura Casa Cărții de Știință, Cluj-Napoca, 2016.
-  T. Căținaș, I. Chiorean, R. Trîmbițaș, *Analiză numerică*, Editura Presa Universitară Clujeană, Cluj-Napoca, 2010.
-  R. Despa, C. Coculescu, *Metode Numerice*, Ed. Universitară, București, 2006.
-  G. Grebenișan, *Metode numerice: aplicații în Matlab: îndrumător de laborator*, Ed. Universității din Oradea, 2008.
-  M. H. Holmes, *Introduction to Scientific Computing and Data Analysis*, Springer International Publishing, Switzerland, 2016.
-  C. Moler, *Numerical Computing in MATLAB*, SIAM, 2004, disponibil online la adresa <http://www.mathworks.com/moler>.

-  C. V. Muraru, *Metode Numerice: Seminarii Matlab*, Ed. EduSoft, Bacău, 2005.
-  S. Nakamura, *Numerical Analysis and Graphic Visualization with Matlab*, The Ohio State University, Columbus, Ohio, 1996.
-  M. Novac, O. Novac, C. Vancea, *Metode Numerice. Îndrumător de laborator pentru uzul studenților*, Ed. Universității din Oradea, 2003.
-  I. Paraschiv-Munteanu, D. Stănică, *Analiză numerică. Exerciții și teme de laborator – Ed. a 2-a rev.*, Ed. Universității din București, 2008.
-  S.S. Rao, *Applied Numerical Methods for Engineers and Scientists*, Pretince Hall, University of Miami, Florida, 2002.
-  Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
-  E. Süli, D.F. Mayers: *An Introduction to Numerical Analysis*, Cambridge University Press, Cambridge, 2003.
-  R.T. Trîmbițaș, *Analiză numerică. O introducere bazată pe Matlab*, Ed. Presa Universitară Clujeană, 2005.
-  C. Vancea, F. Vancea, *Metode Numerice prezentate în Matlab*, Ed. Universității din Oradea, 2001.