

Scurtă introducere în Matlab/Octave

MATLAB="matrix laboratory"

<https://www.mathworks.com/help/matlab/language-fundamentals.html>

- În linia de comandă se pot efectua operații simple sau apeluri de funcții.

```
>> 1/3  
ans = 0.3333
```

- Formatul standard de afișare a numerelor zecimale este **short**. Pentru a putea vizualiza mai multe zecimale se poate trece la formatul **long**.

```
>> pi  
ans = 3.14161  
>> format long  
>> pi  
ans = 3.141592653589793
```

- În apelul

```
>> format short  
>> 1/2019  
ans = 4.9529e-04
```

răspunsul, în care apare litera "e" (de la "exponent"), reprezintă numărul: $4.9529 \cdot 10^{-4}$.

- Cu **fprintf** se poate afișa un număr cu câte zecimale se dorește, împreună cu un mesaj.

```
>> fprintf('1/2019 este aproximativ %7.6f.\n', 1/2019)  
1/2019 este aproximativ 0.000495.
```

- Comenzile **help** și **doc** oferă documentațiile funcțiilor/operatorilor.

```
>> help doc
```

- Pentru a crea o funcție se poate deschide un nou script.

```
function out=my_first_function(in)  
out=NaN; %NaN="not a number"  
if in~=1 %~ este operatorul logic "diferit"  
    return  
else  
    disp('Hello World!'); out=1;  
end  
end
```

Script-ul se salvează cu extensia **.m** și cu numele funcției (pentru exemplul de mai sus: "my_first_function.m"). Pentru a putea apela funcția trebuie setat folderul curent cel în care este salvat script-ul.

```
>> x=my_first_function(1)
Hello World!
x = 1
>> x=my_first_function(0)
x = NaN
```

În funcția de mai sus ; permite mai multe comenzi pe același rând și, în același timp, nu permite afișarea atribuirilor de valori (omiterea lui ; poate fi echivalată cu folosirea funcției **disp**). Textele după % sunt comentarii.

- Exemple de alți operatori logici: **==**, **<=**, **>=**, **<**, **>**, **&&**, **||**.
- Exemple de operații/funcții cu matrice:

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

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

```
>> ones(2)
```

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

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

```
A = 1    2
     3    4
```

```
>> B=[5;6]
```

```
B = 5
     6
```

```
>> C=[7, 8, 9]
```

```
C = 7    8    9
```

```
>> D=[A B;C]
```

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

```
>> D'
```

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

```
>> D(1,[3 1])
```

```
ans = 5    1
```

```
>> D(:,2)
```

```
ans = 2
      4
      8
```

● Atenție la operațiile matriciale: $*$ / $^$ fără punct, respectiv cu punct: $.*$ $./$ $.^$!

```
>> D^2
```

```
ans = 42    50    62
      57    70    93
      94   118   164
```

```
>> D.^2
```

```
ans = 1     4    25
      9    16    36
     49    64    81
```

```
>> [nrl,nrc]=size(D)
```

```
nrl = 3
```

```
nrc = 3
```

```
>> d=D(:)'
```

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

```
>> d=d(3:end)
```

```
d = 7     2     4     8     5     6     9
```

● Exemple de vectori cu elemente echidistante:

```
>> v=1:10
```

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

```
>> v=10:-1:1
```

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

```
>> v=0:2:10
```

```
v = 0 2 4 6 8 10
```

```
>> v=3:-1.5:-3
```

```
v = 3.0000 1.5000 0 -1.5000 -3.0000
```

```
>> v=linspace(1,2,11)
```

```
v = 1.0000 1.1000 1.2000 1.3000 1.4000 1.5000 1.6000 1.7000 1.8000 1.9000 2.0000
```

● Exemple de operații/funcții cu vectori:

```
>> v= repmat([1:3],1,3)
```

```
v = 1 2 3 1 2 3 1 2 3
```

```
>> length(v)
```

```
ans = 9
```

```
>> sum(v)
```

```
ans = 18
```

```
>> cumsum(v)
```

```
ans = 1 3 6 7 9 12 13 15 18
```

```
>> diff(v)
```

```
ans = 1 1 -2 1 1 -2 1 1
```

```
>> find(v==1)
```

```
ans = 1 4 7
```

● Trei implementări ale funcției *dublu factorial* (exemple de utilizare a instrucțiunilor **if**, **else**, **elseif**, **for** și **while** și de apel recursiv):

```
function out=double_factorial_v1(n)
% n e numar natural strict pozitiv
out=1;
if mod(n,2)==0
    first=2;
else
    first=1;
end
for step=first:2:n
    out=out*step;
end
end
```

```
function out=double_factorial_v2(n)
% n e numar natural strict pozitiv
out=n;
while n>=3
    out=out*(n-2);
    n=n-2;
end
end
```

```
function out=double_factorial_v3(n)
% n e numar natural strict pozitiv
if n==1
    out=1;
elseif n==2
    out=2;
else
    out=n*double_factorial_v2(n-2);
end
end
```

- Exemplu de “*function handle*”:

```
>> f=@(x) cos(x).^2
```

f = function_handle with value:
 @(x)cos(x).^2

```
>> x=linspace(0,pi,4)
```

```
x = 0    1.0472    2.0944    3.1416
```

```
>> f(x)
```

```
ans = 1.0000    0.2500    0.2500    1.0000
```

- **clear all** șterge variabilele memorate (din *Workspace*); **clc** curăță fereastra de comandă.

Documentații: http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf

<https://octave.org/octave.pdf>

- **randsample** simulează extrageri aleatoare dintr-o listă cu obiecte; funcția este disponibilă în Octave prin încărcarea pachetului *Statistics*:

```
>> pkg load statistics
```

```
>> help randsample
```

În exemplul:

```
>> randsample('PROBABILITY',20,true)
```

```
ans = TITRAIIBIAYOITBLIOIY
```

se simulează 20 extrageri, alegând aleator la fiecare extragere câte o literă din cuvântul 'PROBABILITY' (*sampling with replacement*), iar în exemplul:

```
>> randsample('PROBABILITY',4,false)
```

```
ans = YLBI
```

se simulează 4 extrageri, alegând aleator la fiecare extragere câte o literă din cuvântul 'PROBABILITY' din care s-au șters literele extrase anterior (*sampling without replacement*).

- **rand** simulează alegerea *uniformă* a unui număr aleator x din intervalul $I=[0,1]$ astfel încât pentru orice subinterval $J \subset I$ probabilitatea (geometrică) a evenimentului “ x este în J ” este egală cu lungimea lui J ; de asemenea, **rand** poate genera vectori (sau matrice) de numere aleatoare cu proprietatea descrisă anterior; de exemplu:

```
>> v=rand(1,2000);
```

generează un vector cu 2000 de numere aleatoare; se poate verifica faptul că proporțiile de numere din v care aparțin intervalelor $(0.15,0.6)$, respectiv $(0.3,0.75)$, sunt aproximativ egale cu lungimea acestor intervale, adică 0.45:

```
>> sum(0.15<v&v<0.6)/2000
```

```
ans = 0.4515
```

```
>> sum(0.3<v&v<0.75)/2000
```

```
ans = 0.4580
```

- **plot** desenează puncte, respectiv segmente; exemple:

```
>> Ax=0; Ay=0; Bx=1; By=0; Cx=0; Cy=1;
```

```
>> plot([Ax Bx Cx],[Ay By Cy], 'b', 'LineWidth', 2)
```

desenează cu albastru segmentele [AB] și [BC], unde A, B, C au coordonatele (Ax,Ay), (Bx,By), respectiv (Cx,Cy), iar

```
>> plot([Ax Bx Cx],[Ay By Cy], '*r', 'MarkerSize', 5)
```

desenează cu câte un asterisc roșu * punctele A,B,C.

- **rectangle** desenează un dreptunghi; exemplu:

```
>> rectangle('Position',[0 0 3 2], 'FaceColor', 'g')
```

desenează un dreptunghi cu vârful inferior din partea stângă de coordonate (0,0), de lungime 3 și lățime 2, umplut cu culoarea verde.

- **clf** curăță fereastra grafică; **hold on** păstrează desenele anterioare din fereastra grafică (e.g. prin apelarea funcției **plot** se șterg desenele anterioare, dacă nu este activată anterior instrucțiunea **hold on**; o singură executare a instrucțiunii **hold on** este suficientă, iar dezactivarea ei se face cu **hold off**).

- **axis square** ajustează lungimile axelor astfel încât desenele să fie vizibile într-un pătrat din planul cartezian; **axis off** ascunde axele, dar păstrează proporțiile setate anterior.

- **pdist** returnează distanța Euclidiană dintre două puncte din plan; exemplu:

```
>> O=[0,0]; P=[4,3];
```

```
>> pdist([O;P])
```

```
ans = 5
```