

Introducere în MATLAB

Variabile, matrice, funcții, structuri de control și reprezentări grafice

Ideea generală

MATLAB este un mediu de calcul numeric conceput pentru lucrul cu vectori, matrice, funcții și algoritmi numerici.

Numele provine de la *MATrix LABoratory*. Din acest motiv, lucrul cu matrice este natural în MATLAB: chiar și un număr real poate fi privit ca o matrice de dimensiune 1×1 .

MATLAB poate fi utilizat:

- interactiv, prin comenzi scrise în Command Window;
- prin scripturi, adică fișiere care conțin succesiuni de comenzi;
- prin funcții definite de utilizator.

Interfața MATLAB

Zone importante ale interfeței:

- **Command Window**: introducerea și executarea comenzilor;
- **Workspace**: variabilele existente în sesiunea curentă;
- **Current Folder**: directorul de lucru curent;
- **Editor**: scrierea scripturilor și funcțiilor;
- **Command History**: istoricul comenzilor executate.

2 + 3

Dacă rezultatul nu este atribuit unei variabile, MATLAB îl memorează temporar în variabila `ans`.

Comenzi de bază

Câteva comenzi folosite frecvent:

```
clc          % sterge continutul vizibil din Command Window
clear        % elimina variabilele din Workspace
close all    % inchide ferestrele grafice
who          % afiseaza variabilele existente
whos         % afiseaza variabilele si informatii despre ele
help sin     % ajutor scurt pentru functia sin
doc plot     % documentatie detaliata pentru plot
```

Semnul ; de la finalul unei instrucțiuni suprimă afișarea rezultatului.

Variabile și expresii

O variabilă se definește prin atribuire:

```
a = 3;  
b = 5;  
c = a + b;
```

Variabilele nu trebuie declarate în prealabil. MATLAB stabilește tipul și dimensiunea atunci când primește valoarea.

MATLAB face distincție între litere mari și mici:

```
a = 2;  
A = 5;
```

Cele două variabile sunt diferite.

Nume de variabile

Un nume de variabilă trebuie să înceapă cu o literă și poate conține litere, cifre și caracterul `_`.

Este bine să evităm nume care coincid cu funcții sau constante MATLAB:

```
sin  
sum  
max  
pi
```

Dacă folosim un astfel de nume pentru o variabilă, putem masca funcția MATLAB corespunzătoare.

Operații aritmetice

Operatorii aritmetici elementari sunt:

+ - * / ^

Exemplu:

```
a = 4;  
b = 2;  
  
s = a + b;  
d = a - b;  
p = a * b;  
q = a / b;  
r = a^b;
```

Funcții matematice uzuale:

`sqrt(x)`, `abs(x)`, `exp(x)`, `log(x)`, `sin(x)`, `cos(x)`

Vectori

Un vector linie se definește prin:

```
v = [1 2 3 4]  
v = [1, 2, 3, 4]
```

Un vector coloană se definește prin:

```
w = [1; 2; 3; 4]
```

Spațiul sau virgula separă elemente de pe aceeași linie. Punctul și virgula separă linii.

Operatorul două puncte :

Operatorul : construiește vectori cu elemente echidistante.

```
x = 1:5
```

produce vectorul

```
1 2 3 4 5
```

Forma generală este:

```
valoare_initiala : pas : valoare_finala
```

Exemplu:

```
x = 0:0.2:1
```

O alternativă este:

```
x = linspace(0,1,6)
```

care produce 6 valori uniform distribuite între 0 și 1.

Matrice

O matrice se poate introduce direct:

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

Aceasta reprezintă

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}.$$

Matricea poate fi scrisă și pe mai multe linii:

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

Matrice speciale

MATLAB oferă funcții pentru construirea unor matrice speciale:

```
zeros(3,4)    % matrice nula 3 x 4  
ones(2,5)     % matrice 2 x 5 cu toate elementele 1  
eye(4)        % matricea identitate de ordin 4  
rand(3,3)     % matrice 3 x 3 cu elemente aleatoare
```

Funcția `diag` poate construi o matrice diagonală:

```
v = [2 4 6];  
D = diag(v);
```

Dacă `A` este matrice, atunci `diag(A)` extrage diagonala principală.

Dimensiunea unui vector sau a unei matrice

Pentru dimensiune folosim:

```
size(A)
```

Putem salva separat numărul de linii și de coloane:

```
[m,n] = size(A);
```

Pentru vectori:

```
length(v)
```

Numărul total de elemente:

```
numel(A)
```

Accesarea elementelor

Dacă

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

atunci

```
A(2,3)
```

este elementul de pe linia 2 și coloana 3.

Selectarea unei linii sau coloane:

```
A(2,:)    % linia a doua  
A(:,3)    % coloana a treia
```

Selectarea unei submatrice:

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

Operatorul end și modificarea elementelor

Ultimul element al unui vector:

```
v(end)
```

Ultima linie a unei matrice:

```
A(end, :)
```

Modificarea unui element:

```
A(2,3) = 10;
```

Modificarea unei linii:

```
A(1,:) = [2 4 6];
```

Eliminarea unei linii sau coloane:

```
A(2,:) = [];  
A(:,3) = [];
```

Concatenarea vectorilor

Parantezele drepte pot fi utilizate pentru concatenare.

Concatenare orizontală:

```
a = [1 2 3];  
b = [4 5];  
c = [a b];
```

Rezultat:

```
c = [1 2 3 4 5]
```

Concatenare verticală:

```
a = [1 2 3];  
b = [4 5 6];  
C = [a; b];
```

Rezultat:

$$C = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

Concatenarea matricelor

Dacă

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

atunci

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

Pentru concatenare orizontală, matricele trebuie să aibă același număr de linii.

Pentru concatenare verticală, matricele trebuie să aibă același număr de coloane.

Matrice pe blocuri

Se pot construi matrice pe blocuri:

```
A = eye(2);  
B = ones(2,1);  
C = zeros(1,2);  
  
M = [A B; C 3]
```

Se obține:

$$M = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 3 \end{pmatrix}.$$

Această construcție este foarte utilă în algebra liniară numerică.

Transpunerea

Transpusa unei matrice reale se obține prin:

A'

Pentru matrice complexe, A' calculează transpusa conjugată.

Dacă dorim numai transpunerea, fără conjugare, folosim:

$A.^{'}$

Pentru matrice reale, cele două operații coincid.

Operații cu matrice

Operațiile de adunare și scădere se fac element cu element:

$$\begin{aligned} A + B \\ A - B \end{aligned}$$

Produsul matricial se notează prin:

$$A * B$$

Puterea unei matrice pătratice:

$$A^2$$

înseamnă produsul matricial AA .

Operații element cu element

MATLAB distinge între operații matriciale și operații element cu element.

```
A * B    % produs matricial  
A .* B   % inmultire element cu element  
A / B    % impartire matriciala  
A ./ B   % impartire element cu element  
A^2      % putere matriciala  
A.^2     % fiecare element la patrat
```

Pentru funcția $f(x) = x^2 - 2x + 1$ evaluată pe un vector:

```
y = x.^2 - 2*x + 1;
```

nu

```
y = x^2 - 2*x + 1;
```

Funcții aplicate vectorilor și matricelor

Funcțiile uzuale se aplică direct componentelor unui vector:

```
x = [0 pi/2 pi];  
y = sin(x);
```

Alte exemple:

```
cos(x), exp(x), log(x), sqrt(x), abs(x)
```

Pentru o matrice, $\sin(A)$ aplică funcția sinus fiecărui element.

Atenție: exponențiala matricială este $\expm(A)$, nu $\exp(A)$.

Funcții uzuale pentru vectori și matrice

Funcții frecvente:

```
sum(v)  
max(v)  
min(v)  
mean(v)  
norm(v)
```

Pentru matrice, unele funcții operează implicit pe coloane:

```
sum(A)
```

returnează suma elementelor de pe fiecare coloană.

Suma tuturor elementelor:

```
sum(A, 'all')
```

sau, pentru compatibilitate cu versiuni mai vechi:

```
sum(A(:))
```

Indexare liniară

Elementele unei matrice pot fi accesate și printr-un singur indice. Numerotarea se face pe coloane. Dacă

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

atunci

```
A(1) % 1  
A(2) % 3  
A(3) % 2  
A(4) % 4
```

Expresia

```
A(:)
```

transformă matricea într-un vector coloană.

Scripturi MATLAB

Un script este un fișier `.m` care conține o succesiune de comenzi MATLAB.

Exemplu: `exemplu.m`

```
clear;  
clc;  
  
a = 2;  
b = 3;  
  
c = a^2 + b^2;  
  
fprintf('Rezultatul este %g\n',c);
```

Scriptul se rulează cu butonul **Run** sau scriind `exemplu` în Command Window.

Fișiere de tip funcție

Pentru calcule reutilizabile definim funcții.

Fișierul `patrat.m`:

```
function y = patrat(x)
    y = x.^2;
end
```

Apel:

```
patrat(3)

v = [1 2 3 4];
w = patrat(v);
```

Numele fișierului trebuie, în mod obișnuit, să coincidă cu numele funcției principale.

Mai multe intrări și ieșiri

Funcție cu două argumente:

```
function z = suma(a,b)
    z = a + b;
end
```

Funcție cu două rezultate:

```
function [s,p] = suma_produs(a,b)
    s = a + b;
    p = a * b;
end
```

Apel:

```
[s,p] = suma_produs(2,3);
```

Diferența este importantă:

- un script execută instrucțiunile în Workspace-ul curent;
- o funcție are propriul spațiu de lucru;
- o funcție primește date prin argumente de intrare;
- o funcție returnează rezultate prin argumente de ieșire.

Funcțiile permit o organizare mai clară a programelor și sunt mai ușor de reutilizat.

Funcții anonime

Pentru funcții simple putem utiliza funcții anonime.

```
f = @(x) x.^2 - 2*x + 1;
```

Apel:

```
f(3)  
  
x = 0:0.1:2;  
y = f(x);
```

Funcție de două variabile:

```
f = @(x,y) x.^2 + y.^2;
```

Directorul curent și fișierele

MATLAB caută fișiere în directorul curent și în directoarele aflate în path.

Comenzi utile:

```
pwd      % directorul curent
dir      % continutul directorului curent
cd       % schimbarea directorului
path     % directoarele cunoscute de MATLAB
```

Pentru a utiliza o funcție proprie, fișierul trebuie să fie în directorul curent sau într-un director cunoscut de MATLAB.

Comentarii și secțiuni

Comentariile în MATLAB încep cu simbolul %:

```
% Calculul valorii functiei  
y = x.^2 + 1;
```

Pentru organizarea unui script mai lung:

```
%% Definirea datelor  
  
%% Calculul valorilor  
  
%% Afisarea rezultatelor
```

Aceste secțiuni pot fi rulate separat din Editor.

Afișarea rezultatelor

Afișare simplă:

```
disp(x)
```

Afișare formatată:

```
x = 3.1415926;  
fprintf('x = %.3f\n',x);
```

Modul general de afișare:

```
format short  
format long
```

Aceste comenzi schimbă doar afișarea, nu precizia internă a calculelor.

Exemplu complet: calcul pe vector

Considerăm funcția

$$f(x) = x^2 - 2x + 2.$$

Calculăm valorile în 11 puncte din intervalul $[0, 2]$:

```
clear;  
clc;  
  
x = linspace(0,2,11);  
y = x.^2 - 2*x + 2;  
  
m = min(y);  
  
fprintf('Valoarea minima este %.4f\n',m);
```

Instrucțiunea if

Instrucțiunea if permite executarea condiționată a unor comenzi.

```
x = -2;  
  
if x >= 0  
    disp('x este nenegativ');  
else  
    disp('x este negativ');  
end
```

Pentru mai multe cazuri:

```
if x > 0  
    disp('x este pozitiv');  
elseif x < 0  
    disp('x este negativ');  
else  
    disp('x este zero');  
end
```

Operatori relaționali și logici

Operatori relaționali:

< > <= >= == ~=

Atenție la diferența dintre atribuire și test de egalitate:

```
x = 3;    % atribuire  
x == 3    % test de egalitate
```

Operatori logici pentru condiții scalare:

&& || ~

Exemplu:

```
x = 0.5;  
if x >= 0 && x <= 1  
    disp('x apartine intervalului [0,1]');  
end
```

Instrucțiunea for

Instrucțiunea for repetă un grup de comenzi pentru un număr cunoscut de pași.

```
for k = 1:5  
    disp(k);  
end
```

Suma primelor zece numere naturale:

```
s = 0;  
for k = 1:10  
    s = s + k;  
end
```

Cu pas diferit:

```
for k = 0:0.5:2  
    disp(k);  
end
```

Bucă for și vectorizare

Calcul prin buclă:

```
for k = 1:5  
    y(k) = k^2;  
end
```

Formulare vectorizată:

```
k = 1:5;  
y = k.^2;
```

Când construim un vector într-o buclă, este bine să alocăm memoria înainte:

```
n = 100;  
y = zeros(1,n);  
for k = 1:n  
    y(k) = k^2;  
end
```

Bucle for imbricate

Pentru matrice putem utiliza bucle imbricate:

```
A = zeros(3,4);  
  
for i = 1:3  
    for j = 1:4  
        A(i,j) = i + j;  
    end  
end
```

Fiecare element al matricei este calculat separat.

Când este posibil, se preferă o formulare vectorială sau matricială.

Instrucțiunea while

Instrucțiunea while este utilizată atunci când numărul de repetări nu este cunoscut dinainte.

```
x = 1;  
while x < 100  
    x = 2*x;  
end
```

Buclo se oprește atunci când condiția nu mai este satisfăcută.

Trebuie avut grijă ca în interiorul buclei să fie modificată o variabilă care influențează condiția. Altfel se poate obține o buclă infinită.

Exemplu cu while

Determinăm numărul de termeni necesari pentru ca suma

$$1 + \frac{1}{2} + \frac{1}{4} + \dots$$

să depășească valoarea 1.9.

```
s = 0;
termen = 1;
n = 0;

while s < 1.9
    s = s + termen;
    termen = termen/2;
    n = n + 1;
end
```

Instrucțiunile break și continue

break întrerupe executarea unei bucle:

```
for k = 1:100
    if k^2 > 50
        break;
    end
end
```

continue trece direct la următoarea iterație:

```
for k = 1:10
    if mod(k,2) == 0
        continue;
    end
    disp(k);
end
```

În acest exemplu sunt afișate numai numerele impare.

Reprezentarea grafică a curbelor

Pentru o curbă plană folosim comanda:

```
plot(x,y)
```

unde x și y sunt vectori de aceeași lungime.

Exemplu:

```
x = linspace(0,2*pi,200);  
y = sin(x);
```

```
plot(x,y);
```

Cu cât numărul de puncte este mai mare, cu atât curba este reprezentată mai neted.

Titlu, axe și grilă

Un grafic poate fi completat prin:

```
xlabel('x');  
ylabel('sin(x)');  
title('Funcția sinus');  
grid on;
```

Exemplu complet:

```
x = linspace(0,2*pi,200);  
y = sin(x);  
  
plot(x,y);  
xlabel('x');  
ylabel('sin(x)');  
title('Funcția sinus');  
grid on;
```

Mai multe curbe în aceeași figură

Variantă directă:

```
x = linspace(0,2*pi,200);  
plot(x,sin(x),x,cos(x));  
legend('sin(x)', 'cos(x)');
```

Variantă cu hold on:

```
plot(x,sin(x));  
hold on;  
plot(x,cos(x));  
hold off;  
legend('sin(x)', 'cos(x)');  
grid on;
```

Stilul curbei și limitele axelor

Tipuri de linii și markeri:

```
plot(x,y,'-')  
plot(x,y,'--')  
plot(x,y,':')  
plot(x,y,'-.')  
plot(x,y,'o')  
plot(x,y,'-o')
```

Grosimea liniei:

```
plot(x,y,'LineWidth',1.5);
```

Limitarea axelor:

```
xlim([0 2*pi])  
ylim([-1.2 1.2])  
axis([0 2*pi -1.2 1.2])
```

Mai multe grafice în aceeași fereastră

Comanda `subplot` împarte fereastra grafică.

```
x = linspace(0,2*pi,200);
```

```
subplot(2,1,1);  
plot(x,sin(x));  
title('sin(x)');
```

```
subplot(2,1,2);  
plot(x,cos(x));  
title('cos(x)');
```

Prima comandă selectează prima zonă dintr-o împărțire cu 2 linii și 1 coloană.

Curbe parametrizate

Cercul unitate:

$$x(t) = \cos t, \quad y(t) = \sin t.$$

```
t = linspace(0,2*pi,300);  
x = cos(t);  
y = sin(t);  
  
plot(x,y);  
axis equal;  
grid on;
```

Comanda `axis equal` păstrează aceeași scară pe cele două axe.

Curbe în spațiu

O curbă spațială poate fi reprezentată prin `plot3`.

Elicea:

$$x(t) = \cos t, \quad y(t) = \sin t, \quad z(t) = t.$$

```
t = linspace(0,6*pi,500);  
  
x = cos(t);  
y = sin(t);  
z = t;  
  
plot3(x,y,z);  
grid on;  
xlabel('x'); ylabel('y'); zlabel('z');
```

Reprezentarea suprafețelor

Pentru o funcție $z = f(x, y)$ construim mai întâi o rețea de puncte.

```
x = linspace(-2,2,50);  
y = linspace(-2,2,50);  
  
[X,Y] = meshgrid(x,y);
```

Pentru $z = x^2 + y^2$:

```
Z = X.^2 + Y.^2;  
surf(X,Y,Z);
```

Este importantă utilizarea operatorului `.^`.

Comenzile surf și mesh

Reprezentare de tip suprafață:

```
surf(X,Y,Z);  
xlabel('x'); ylabel('y'); zlabel('z');  
title('z = x^2 + y^2');
```

Reprezentare de tip plasă:

```
mesh(X,Y,Z);
```

surf evidențiază suprafața, iar mesh evidențiază rețeaua de reprezentare.

Exemplu: suprafață oscilantă

Pentru

$$z = \sin x \cos y$$

scriem:

```
x = linspace(-pi,pi,100);  
y = linspace(-pi,pi,100);  
  
[X,Y] = meshgrid(x,y);  
Z = sin(X).*cos(Y);  
  
surf(X,Y,Z);  
xlabel('x'); ylabel('y'); zlabel('z');  
title('z = sin(x) cos(y)');
```

Apare din nou operatorul `.*`.

Curbe de nivel

Curbele de nivel se reprezintă prin:

```
contour(X,Y,Z);
```

Pentru $z = x^2 + y^2$:

```
x = linspace(-2,2,100);  
y = linspace(-2,2,100);  
  
[X,Y] = meshgrid(x,y);  
Z = X.^2 + Y.^2;  
  
contour(X,Y,Z);  
axis equal;
```

Curbele obținute sunt cercurile $x^2 + y^2 = c$.

Suprafață și curbe de nivel

Comanda

```
surf(X,Y,Z)
```

combină o suprafață tridimensională cu proiecția curbelor sale de nivel.

```
x = linspace(-2,2,100);  
y = linspace(-2,2,100);  
[X,Y] = meshgrid(x,y);  
Z = exp(-(X.^2 + Y.^2)).*cos(3*X).*sin(3*Y);  
  
surf(X,Y,Z);  
grid on;
```

Bucle și grafice

Putem calcula valorile unei funcții folosind o buclă:

```
x = linspace(0,5,200);  
y = zeros(size(x));  
  
for k = 1:length(x)  
    y(k) = exp(-x(k))*sin(4*x(k));  
end  
  
plot(x,y);  
grid on;
```

Dar forma vectorizată este mai simplă:

```
y = exp(-x).*sin(4*x);
```

Exemplu complet pentru o suprafață

Considerăm

$$f(x, y) = e^{-(x^2+y^2)} \cos(3x) \sin(3y).$$

```
clear; clc; close all;  
  
x = linspace(-2,2,100);  
y = linspace(-2,2,100);  
[X,Y] = meshgrid(x,y);  
  
Z = exp(-(X.^2 + Y.^2)).*cos(3*X).*sin(3*Y);  
  
surf(X,Y,Z);  
xlabel('x'); ylabel('y'); zlabel('f(x,y)');  
title('Suprafata z = f(x,y)');  
grid on;
```

- MATLAB este orientat spre calcul vectorial și matricial.
- Pentru vectori și matrice, diferența dintre `*` și `.*`, respectiv dintre `^` și `.^` este esențială.
- Scripturile grupează comenzi, iar funcțiile organizează calcule reutilizabile.
- `if`, `for` și `while` controlează ordinea de execuție.
- Pentru curbe folosim `plot`; pentru suprafețe folosim `meshgrid`, apoi `surf`, `mesh` sau `contour`.