# Matrice în MATLAB

## Cuprinsul



Matricele sunt tipuri de date fundamentale în MATLAB. Ele sunt de fapt tablouri multidimensionale în dublă precizie. Cele mai folosite sunt matricele bidimensionale, care sunt tablouri bidimensionale cu $m$ linii şi $n$ coloane. Vectorii linie ($m = 1$) şi coloană ($n = 1$) sunt cazuri particulare de matrice bidimensionale.

### Generarea matricelor

Matricele pot fi generate explicit. Delimitatorii: [], blanc sau virgula în interiorul liniei, ; sau CR între linii.

```
A = [5 7 9
1 -3 -7]
```

```
A = 2x3
     5     7     9
     1    -3    -7
```

Echivalent

```
A = [5, 7, 9; 1, -3, -7]
```

```
A = 2x3
     5     7     9
     1    -3    -7
```

Dimensiunea unei matrice se obţine cu comanda `size`

```
v = size(A)
```

```
v = 1x2
     2     3
```

```
[r,c] = size(A)
```

```
r = 2
c = 3
```

**Generare de matrice speciale**

| | |
|---|---|
| zeros | Matricea nulă |
| ones | Matrice formata numai din elemente 1 |
| eye | Matricea unitate |
| repmat | Replicarea matricelor |
| rand | Matrice aleatoare cu elemente uniforme in [0,1] |
| randn | Matrice aleatoare cu elemente distribuite normal |
| randi | Matrice aleatoare cu elemente întregi |
| linspace | Vector de elemente echidistante |
| logspace | Vector de elemente spaţiate logaritmic |

Primele şase au aceeaşi sintaxă, de exemplu `zeros(m,n)` sau `zeros([m,n])` generează matricea nulă de tip $m \times n$. `eye(size(A))` generează o matrice unitate de aceeaşi dimensiune ca `A`. `ones(n)` este echivalentă cu `ones(n,n)`.

```
rand
```

```
ans = 0.8147
```

```
rand(3)
```

```
ans = 3x3
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
    0.9134    0.2785    0.9649
```

Reproductibilitatea experimentelor – iniţializarea generatorului : se face cu funcţia `rng`

### Generare pe blocuri

Fie matricea `B`

```
B = [1, 2; 3, 4]
```

```
B = 2x2
     1     2
     3     4
```

Din ea generăm:

```
C=[B, zeros(2); ones(2), eye(2)]
```

```
C = 4x4
     1     2     0     0
     3     4     0     0
     1     1     1     0
     1     1     0     1
```

`blkdiag` permite generarea matricelor diagonale pe blocuri

```
A=blkdiag(2*eye(2),ones(2))
```

```
A = 4x4
     2     0     0     0
     0     2     0     0
     0     0     1     1
     0     0     1     1
```

Funcţia `repmat` permite construirea de matrice prin repetarea de subblocuri: `repmat(A,m,n)` crează o matrice de `m` pe `n` blocuri în care fiecare bloc este o copie a lui `A`. Dacă `n` lipseşte, valoarea sa implicită este `m`. Exemplu:

```
A=repmat(eye(2),2)
```

```
A = 4x4
     1     0     1     0
     0     1     0     1
     1     0     1     0
     0     1     0     1
```

Tabela următoare dă câteva funcţii pentru manipularea matricelor.

| | |
|---|---|
| `reshape` | Schimbarea dimensiunii |
| `diag` | Matrice diagonale şi diagonale ale matricelor |
| `blkdiag` | Matrice diagonală pe blocuri |
| `tril` | Extragerea părţii triunghiulare inferior |
| `triu` | Extragerea părţii triunghiulare superior |
| `fliplr` | Rotire matrice în jurul axei de simetrie verticale |
| `flipud` | Rotire matrice în jurul axei de simetrie orizontale |
| `rot90` | Rotaţia unei matrice cu 90 de grade |

Funcţia `reshape` schimbă dimensiunile unei matrice: `reshape(A,m,n)`produce o matrice `m` pe `n` ale cărei elemente sunt luate coloană cu coloană din `A`. De exemplu:

```
A=[1 4 9; 16 25 36], B=reshape(A,3,2)
```

```
A = 2x3
     1     4     9
    16    25    36

B = 3x2
     1    25
    16     9
     4    36
```

Funcţia `diag` lucrează cu diagonalele unei matrice şi poate avea ca argument o matrice sau un vector. Pentru un vector `x`, `diag(x)` este matricea cu diagonala principală `x`:

```
diag([1,2,3])
```

```
ans = 3x3
     1     0     0
     0     2     0
     0     0     3
```

Mai general, `diag(x,k)` pune `x` pe diagonala cu numărul `k`, unde `k = 0` înseamnă diagonala principală, `k > 0` specifică diagonale situate deasupra diagonalei principale, iar `k < 0` diagonale dedesubtul diagonalei principale:

```
diag([1,2],1)
```

```
ans = 3x3
     0     1     0
     0     0     2
     0     0     0
```

```
diag([3 4],-2)
```

```
ans = 4x4
     0     0     0     0
     0     0     0     0
     3     0     0     0
     0     4     0     0
```

Pentru o matrice `A`, `diag(A)` este vectorul coloană format din elementele de pe diagonala principală a lui `A`. Pentru a produce o matrice diagonală având aceaşi diagonală ca `A` se va utiliza `diag(diag(A))`. Analog cazului vectorial, `diag(A,k)` produce un vector coloană construit din a `k`-a diagonală a lui `A`. Astfel dacă

```
A = [2, 3, 5; 7, 11, 13; 17, 19, 23];
```

atunci

```
diag(A)
```

```
ans = 3x1
     2
    11
    23
```

```
diag(A,-1)
```

```
ans = 2x1
     7
    19
```

`tril(A)` obţine partea triunghiulară inferior a lui `A` (elementele situate pe diagonala principală şi dedesubtul ei şi în rest zero). Analog lucrează `triu(A)` pentru partea triunghiulară superior. Mai general, `tril(A,k)` dă elementele situate pe diagonala a `k`-a a lui `A` şi dedesubtul ei, în timp ce `triu(A,k)` dă elementele situate pe a `k`-a diagonală a lui A şi deasupra ei. Pentru A ca mai sus:

```
tril(A)
```

```
ans = 3x3
     2     0     0
     7    11     0
    17    19    23
```

```
triu(A,1)
```

```
ans = 3x3
     0     3     5
     0     0    13
     0     0     0
```

```
triu(A,-1)
```

```
ans = 3x3
     2     3     5
     7    11    13
     0    19    23
```

MATLAB posedă un set de funcţii pentru generarea unor matrice speciale. Aceste matrice au proprietăţi interesante care le fac utile pentru construirea de exemple şi testarea algoritmilor. Ele sunt date în tabela 1. Funcţia `gallery` asigură accesul la o colecţie bogată de matrice de test creată de Nicholas J. Higham. Pentru detalii vezi `help gallery`.

| | |
|---|---|
| `compan` | matrice companion |
| `gallery` | colecţie de matrice de test |
| `hadamard` | matrice Hadamard |
| `hankel` | matrice Hankel |
| `hilb` | matrice Hilbert |
| `invhilb` | inversa matricei Hilbert |
| `magic` | pătrat magic |
| `pascal` | matricea Pascal (coeficienţi binomiali) |
| `rosser` | matrice simetrică pentru testarea valorilor proprii |
| `toeplitz` | matrice Toeplitz |
| `vander` | matrice Vandermonde |
| `wilkinson` | matricea lui Wilkinson pentru testarea valorilor proprii |

## Indexarea şi notaţia ":"

Pentru `i` şi `j` scalari `i:j` desemnează vectorul `[i, i+1, ..., j] (pasul este 1)`.

Pentru un pas diferit, `s`, se foloseşte `i:s:j`.

```
1:5
```

```
ans = 1x5
     1     2     3     4     5
```

```
4:-1:-2
```

```
ans = 1x7
     4     3     2     1     0    -1    -2
```

```
0:.75:3
```

```
ans = 1x5
         0    0.7500    1.5000    2.2500    3.0000
```

Elementele individuale ale unei matrice se accesează prin `A(i,j)`, `i`$\geq$1, `j`$\geq$1. Indicii zero sau negativi nu sunt admişi în MATLAB.

`A(p:q,r:s)` desemneaza submatricea obţinută din intersecţia liniilor de la `p` la `q` şi coloanelor de la `r` la `s`.

`A(i, :)` linia `i`, `A(:,j)` coloana `j`

`A(v,w)` unde `v` și w sunt vectori selectează submatricea cu liniile date de `v` si coloanele date de `w`

`A(:)` desemnează matricea A privită ca vector coloană (coloană după coloană).

Dacă `A(:)` apare în stânga se completează A păstrându-i forma.

```
A=zeros(3); A(:)=primes(23); A=A'
```

```
A = 3x3
     2     3     5
     7    11    13
    17    19    23
```

`linspace(a,b,n)` generează `n` puncte echidistante in intervalul [a,b]. Implicit `n=100`. Echivalent `a:(b-a)/(n-1):b`.

`[]` desemnează matricea vidă; utilă la ştergeri şi indicator de poziţie într-o listă de argumente.

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

```
A = 2x3
     2     3     5
    17    19    23
```

## Operaţii în sens matricial şi în sens tablou

Operaţiile asupra matricelor se pot realiza în două moduri: în sens matricial, după regulile algebrei matriciale şi în sens tablou, adică element cu element.

| Operaţia | Sens matricial | Sens tablou |
|---|---|---|
| Adunare | + | + |
| Scadere | - | - |
| Inmulţire | * | .* |
| Împărţire | / | ./ |
| Împărţire stângă | \ | .\ |
| Ridicare la putere | ^ | .^ |

`A/B` este soluţia ecuaţiei matriciale `X*B=A`. `A\B` este soluţia ecuaţiei matriciale `A*X=B`.

```
A=[1 2; 3 4], B=ones(2)
```

```
A = 2x2
     1     2
     3     4

B = 2x2
     1     1
     1     1
```

```
A+B
```

```
ans = 2x2
     2     3
     4     5
```

```
A*B
```

```
ans = 2x2
     3     3
     7     7
```

```
A\B
```

```
ans = 2x2
    -1    -1
     1     1
```

```
A.*B, B./A
```

```
ans = 2x2
     1     2
     3     4

ans = 2x2
    1.0000    0.5000
    0.3333    0.2500
```

```
A^2
```

```
ans = 2x2
     7    10
    15    22
```

```
A.^2
```

```
ans = 2x2
     1     4
     9    16
```

Operaţia `.^` permite ca exponentul să fie un tablou când dimensiunile bazei şi exponentului coincid, sau când baza este un scalar:

```
x=[1 2 3]; y=[2 3 4]; Z=[1 2; 3 4];
x.^y
```

```
ans = 1x3
     1     8    81
```

```
2.^x
```

```
ans = 1x3
     2     4     8
```

```
2.^Z
```

```
ans = 2x2
     2     4
     8    16
```

Calculul inversei se poate face cu `inv`, iar al determinantului cu `det`. Dacă exponentul este negativ, `A^n` înseamnă `inv(A)^(-n)`

Transpusa conjugată a matricei `A` se obţine cu `A’`. Dacă `A` este reală, atunci aceasta este transpusa obişnuită. Transpusa fără conjugare se obţine cu `A.’`. Alternative funcţionale: `ctranspose(A)` şi `transpose(A)`

Produsul scalar a doi vectori `dot(x,y)`, sau dacă `x` şi `y` sunt vectori coloană `x’*y`.

Produsul vectorial `cross(x,y)`

Exemple:

```
x=[-1 0 1]'; y=[3 4 5]';
x'*y
```

```
ans = 2
```

```
dot(x,y)
```

```
ans = 2
```

```
cross(x,y)
```

```
ans = 3x1
    -4
     8
    -4
```

**Expandarea scalarilor**: dacă un scalar se adună la o matrice sau dacă se înmulţeşte (împarte) o matrice cu un scalar, scalarul se operează cu fiecare element al matricei:

```
[4,3;2,1]+4
```

```
ans = 2x2
     8     7
     6     5
```

```
[3 4 5; 4 5 6]/12
```

```
ans = 2x3
    0.2500    0.3333    0.4167
    0.3333    0.4167    0.5000
```

Funcţiile elementare se aplică matricelor şi vectorilor punctual (element cu element):

```
sin(A)
```

```
ans = 2x2
    0.8415    0.9093
    0.1411   -0.7568
```

Funcţiile de matrice în sensul algebrei liniare au numele terminat în m: `expm`, `logm`, `sqrtm`, etc.

De exemplu pentru

```
A = [2 2; 0 2]
```

```
A = 2x2
     2     2
     0     2
```

avem

```
sqrt(A)
```

```
ans = 2x2
    1.4142    1.4142
         0    1.4142
```

```
sqrtm(A)
```

```
ans = 2x2
    1.4142    0.7071
         0    1.4142
```

```
ans*ans
```

```
ans = 2x2
    2.0000    2.0000
         0    2.0000
```

### Analiza datelor

| | |
|---|---|
| `max` | Maximul |
| `min` | Minimul |
| `mean` | Media |
| `median` | Mediana |
| `std` | Abaterea medie pătratică |
| `var` | Dispersia |
| `sort` | Sortare în ordine crescătoare |
| `sum` | Suma elementelor |
| `prod` | Produsul elementelor |
| `cumsum` | Suma cumulată |
| `cumprod` | Produsul cumulat |
| `diff` | Diferența elementelor |

Cel mai simplu mod de aplicare – asupra unui vector

```
x=[4 -8 -2 1 0]; [min(x), max(x)]
```

```
ans = 1x2
    -8     4
```

Aplicate asupra matricelor acţionează pe coloană.

```
A = [0, -1,  2; 1, 2, -4; 5 -3 -4]
```

```
A = 3x3
     0    -1     2
     1     2    -4
     5    -3    -4
```

```
max(A)
```

```
ans = 1x3
     5     2     2
```

```
[m,i]=min(A)
```

```
m = 1x3
     0    -3    -4

i = 1x3
     1     3     2
```

```
min(min(A))
```

```
ans = -4
```

```
min(A(:))
```

```
ans = -4
```

Funcţiile `max` şi `min` pot acţiona şi pe linii printr-un al treilea argument.

```
max (A,[],2)
```

```
ans = 3x1
     2
     2
     5
```

Funcţiile `sum` şi `sort` pot fi facute sa acţioneze pe linii printr-un al doilea argument. Vezi `help sort` sau `doc sort`.

```
x=(1:8).^2
```

```
x = 1x8
     1     4     9    16    25    36    49    64
```

```
diff(x)
```

```
ans =
     3     5     7     9    11    13    15
```

## Operatori relaţionali şi logici

`==`, `~=`, `<`, `>`, `<=`, `>=`

Comparaţia între scalari produce 1 dacă relaţia este adevărată şi 0 în caz contrar. La comparaţia între matrice de aceeaşi dimensiune sau între matrice şi scalari rezultatul este o matrice de 0 şi 1. Comparaţia se face element cu element. Exemple:

```
A = [1, 2; 3, 4]; B = 2*ones(2);
A == B
```

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

```
A>B
```

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

`isequal` testează dacă două matrice sunt identice

```
isequal(A,B)
```

```
ans =
   0
```

| | |
|---|---|
| `ischar` | Testează dacă argumentul este şir de caractere (string) |
| `isempty` | Testează dacă argumentul este vid |
| `isequal` | Testează dacă tablourile sunt identice |
| `isfinite` | Testează dacă elementele unui tablou sunt finite |
| `isieee` | Testeză dacă maşina utilizează aritmetica IEEE |
| `isinf` | Testează dacă elementele unui tablou sunt inf |
| `islogical` | Testează dacă argumentul este un tablou logic |
| `isnan` | Test de NaN |
| `isnumeric` | Testează dacă argumentul este numeric |
| `isreal` | Testează dacă argumentul este tablou real |
| `issparse` | Testează dacă argumentul este tablou rar |

operatori logici `&`, `|`, `~`, `xor`, all, `any`. Exemple:

```
x = [-1 1 1]; y = [1 2 -3];
x>0 & y>0
```

```
ans =
   0   1   0
```

```
x>0 | y>0
```

```
ans =
   1   1   1
```

```
xor(x>0,y>0)
```

```
ans =
   1   0   1
```

```
any(x>0)
```

```
ans =
   1
```

```
all(x>0)
```

```
ans =
   0
```

**Precedenţa operatorilor**

Din help-ul MATLAB

1. transpose (.'), power (.^), complex conjugate transpose ('), matrix power (^)
2. unary plus (+), unary minus (-), logical negation (~)
3. multiplication (.*), right division (./), left division (.\), matrix multiplication (*), matrix right division (/), matrix left division (\)
4. addition (+), subtraction (-)
5. colon operator (:)
6. less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), not equal to (~=)
7. element-wise logical AND (&)
8. element-wise logical OR (|)
9. short-circuit logical AND (&&)
10. short-circuit logical OR (||)

`all` şi `any` aplicate matricelor lucrează pe coloane, rezultatul fiind un vector linie.

`all(all(A==B))` este echivalent cu `isequal(A,B)`

`find` returnează indicii corespunzători elementelor nenule ale unui vector

```
x = [-3 1 0 -inf 0];
f = find(x);
```

Rezultatul lui `find` poate fi utilizat apoi la selecţii

```
x(f)
```

```
ans =
    -3     1  -Inf
```

Selectăm elementele finite ale lui `x`

```
x(find(isfinite(x)))
```

```
ans =
    -3     1     0     0
```

```
x(isfinite(x))          %echivalent
```

```
ans =
    -3     1     0     0
```

Înlocuim elementele negative cu 0

```
x(x<0)=0
```

```
x =
     0     1     0     0     0
```

```
x(find(x<0))=0   %echivalent
```

```
x =
     0     1     0     0     0
```

`find` aplicat unei matrice acţionează ca şi cum matricea ar fi transformată într-un vector coloană.

```
A = [4 2 16; 12 4 3], B = [12 3 1; 10 -1 7]
```

```
A =
     4     2    16
    12     4     3

B =
    12     3     1
    10    -1     7
```

```
f = find(A<B)
```

```
f =
     1
     3
     6
```

```
A(f)=0
```

```
A =
     0     0    16
    12     4     0
```

Pentru matrice, `find` se poate utiliza şi sub forma

```
[i,j] = find(A)
```

```
i =
     2
     2
     1

j =
     1
     2
     3
```

Rezultatele operatorilor logici şi ale funcţiilor logice sunt exemple de tablouri logice. Tablourile logice se pot crea şi prin aplicarea funcţiei `logical` unui tablou numeric. Tablourile logice pot fi utilizate la indexare.

```
clear; y = [1 2 0 -3 0]
```

```
y =
     1     2     0    -3     0
```

```
i1 = logical(y)
```

```
i1 =
   1   1   0   1   0
```

```
i2 = ( y~=0 )
```

```
i2 =
   1   1   0   1   0
```

```
i3 = [1 1 0 1 0]
```

```
i3 =
     1     1     0     1     0
```

```
whos
```

```
  Name      Size            Bytes  Class      Attributes

  i1        1x5                 5  logical
  i2        1x5                 5  logical
  i3        1x5                40  double
  y         1x5                40  double
```

```
y(i1),y(i2)
```

```
ans =
     1     2    -3

ans =
     1     2    -3
```

```
isequal(i2,i3)
```

```
ans =
   1
```

**`y(i3)`**

dă eroare: **`???  Subscript indices must either be real positive integers or logicals.`**

## Matrice rare

Este natural să gândim o matrice ca un tablou rectangular de numere. Totuşi, în multe situaţii reale, matricele sunt şi foarte mari şi foarte **rare (sparse)**, aceasta însemnând că majoritatea elementelor lor sunt zero. Opusul lui rar este **dens** (**full**). De exemplu, structura legăturilor din Web poate fi descrisă printr-o matrice de adiacenţă în care $a_{ij}$ este nenul dacă pagina $j$ referă pagina $i$. Evident, orice pagina se leagă cu o fracţiune neglijabilă a tuturor paginilor Web. În astfel de cazuri, este ineficient, sau chiar imposibil, să memorăm fiecare element. În loc de aceasta, am putea profita de avantajul rarităţii, memorând numai intrările nenule şi poziţiile (indicii) lor. Pentru acest scop MATLAB are un tip de date `sparse`. Comenzile `sparse` şi `full` fac conversiile de la dens la rar şi invers:

```
A = vander(1:3);
sparse(A)
```

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

```
full(ans)
```

```
ans =
     1     1     1
     4     2     1
     9     3     1
```

Conversia unei matrice standard dense într-una rară nu este modul standard de a crea matrice rare—trebuie evitată crearea, chiar şi temporară, de matrice dense foarte mari. O alternativă este să dăm direct lui `sparse` datele brute necesare pentru reprezentare. (Aceasta este inversa funcţională a comenzii `find`.)

```
sparse(1:4,8:-2:2,[2 3 5 7])
```

```
ans =
   (4,2)        7
   (3,4)        5
   (2,6)        3
   (1,8)        2
```

O altă alternativă este crearea unei matrice rare cu suficient spaţiu pentru a păstra un număr specificat de elemente nenule şi apoi umplerea ei utilizând atribuiri standard cu indici.

Altă comandă de construire a matricelor rare utilă este `spdiags`, care construieşte o matrice rară dând diagonalele (matrice bandă):

```
M = ones(6,1)*[-20 Inf 10]
```

```
M =
   -20   Inf    10
   -20   Inf    10
   -20   Inf    10
   -20   Inf    10
   -20   Inf    10
   -20   Inf    10
```

```
full( spdiags( M,[-2 0 1],6,6 ) )
```

```
ans =
   Inf    10     0     0     0     0
     0   Inf    10     0     0     0
   -20     0   Inf    10     0     0
     0   -20     0   Inf    10     0
     0     0   -20     0   Inf    10
     0     0     0   -20     0   Inf
```

Comanda `nnz` ne spune câte elemente nenule are o matrice rară dată. Deoarece este nepractic să inspectăm direct toate elementele unei matrice dintr-o problemă reală (chiar şi cele nenule), comanda `spy` ne ajută să producem grafice care indică poziţiile elementelor nenule. De exemplu, `spy(bucky)` ne arată legăturile dintre cei 60 de atomi de carbon ai moleculei de Buckminsterfullerena (sau bucky-ball). MATLAB are o mulţime de funcţii şi facilităţi pentru lucrul cu matrice rare. Operatorii aritmetici `+` , `-` , `*` şi `^` utilizează algoritmi care exploatează caracterul rar şi produc la ieşire matrice rare dacă intrarile sunt rare. Operatorul backslash `\` selectează automat algoritmi specifici când lucreaza cu matrice rare. Există funcţii specifice pentru rezolvarea iterativă a sistemelor liniare, problemelor de vectori şi valori proprii, problemelor de valori singulare. A se vedea secţiunea referitoare la algebra liniară numerică.

```
spy(bucky);
```