5. CALCUL NUMERIC ÎN MATLAB

Calculul numeric este o etapă importantă în rezolvarea problemelor inginereşti fiind indispensabil în evaluare cantitativă precum şi pentru interpretările calitative necesare în activităţile de proiectare şi verificare a sistemelor. În general, prin calcul numeric se concretizează o etapă anterioară de *calcul simbolic* unde se operează cu elemente (variabile, constante) simbolice în cadrul modelului de reprezentare a problemei. Calculul numeric presupune elaborarea algoritmului şi înlocuirea (atribuirea) cu valori numerice a variabilelor simbolice.

5.1. Problematica calculului numeric în Matlab

Mediul Matlab pune la dispoziţia utilizatorului o colecţie bogată de programe funcţie pentru abordarea unor clase de metode numerice specifice algebrei liniare, aproximării funcţiilor, integrării funcţiilor, rezolvării ecuaţiilor diferenţiale, pentru prelucrarea datelor şi calcule statistice etc.. Majoritatea acestor funcţii sunt conţinute în subdirectoarele toolbox-ului numit chiar **matlab**, de exemplu: **elmat**, **funfun**, **matfun**, etc.. Programele funcţie au extesia m şi nume care sugerează operaţia sau operaţiile efectuate. Utilizarea acestor funcţii în practică presupune cunoaşterea sintaxei specifice şi înţelegerea semnificaţiei argumentelor din lista de intrare şi a variabilelor din lista de ieşire. Prin urmare, recomandăm în general utilizarea comenzii `help nume_functie`, din linia curentă de comandă, pentru obţinerea detaliilor de utilizare şi apelare a oricărei funcţii sau operator în Matlab. Anumite operaţii de calcul numeric în Matlab, cum sunt o parte din cele destinate manipulării matricelor (extragerea submatricelor) nu folosesc funcţii propriuzise, ci o sintaxă inedită care permite accesul la elementele tablourilor de date în vederea efectuării unor selecţii într-o manieră extrem de eficientă. Aceste particularităţi de lucru asupra tablourilor de date conferă Matlab-ului o flexibilitate de lucru şi o eficienţă de programare deosebite în rezolvarea problemelor.

Abordarea sistematică a funcţiilor de calcul numeric în Matlab se bazează pe clasificarea structurală a principalelor categorii de probleme puse spre rezolvare în domeniul ingineriei. În tabelul 5.1 sunt prezentate sintetic principalele categorii de probleme şi operaţii de calcul numeric ce pot fi abordate cu Matlab.

Tab. 5.1. Clasificarea structurală a calculului numeric în Matlab

1. Calcul matriceal	Operaţii cu matrice	- operaţii aritmetice cu matrice - operaţii cu tablouri de date - vectorizarea calculelor
	Manipularea matricelor	-extragerea submatricelor prin indici -extragerea submatricelor prin vectori cu elemente 0-1 -asamblarea matricelor mari -redimensionarea matricelor -rotirea matricelor (în jurul unei coloane/linii/element)
	Analiza matriceală	-calculul determinantului asociat -inversarea matricelor -evaluarea rangului -calculul urmei unei matrice -calculul normelor, vectorilor şi matricelor -condiţionarea matricelor
	Descompunerea şi factorizarea matricelor	-vectori/valori proprii -calculul valorilor singulare -factorizări (LU, QR, Cholesky, pseudoinversa)
2. Rezolvarea sistemelor de ecuaţii liniare	- prin împărţirea matricelor - prin utilizarea matricei inverse - prin utilizarea factorizărilor	- operaţii aritmetice elementare cu matrice şi tablouri
3. Calcule cu numere complexe	- definirea numerelor complexe - modulul - argumentul - conjugatul - partea reală, partea imaginară	- operaţii cu numere complexe - calcul matriceal cu numere complexe
4. Operaţii şi calcule matematice uzuale	Evaluări matematice uzuale	-funcţia semn -funcţia rest -cel mai mare divizor comun -cel mai mic multiplu comun
	Funcţii matematice elementare	- exponenţiala, logaritm, radical, putere
	Funcţii trigonometrice şi hiperbolice	-funcţii trigonometrice directe -funcţii trigonometrice inverse -funcţii hiperbolice directe -funcţii hiperbolice inverse
5. Prelucrarea datelor şi calcule statistice	Funcţii pentru prelucrarea datelor	-minimul/maximul unui şir de date -sortarea elementelor unei matrice
	Funcţii pentru calcule statistice	medie şi mediană, varianţa şi dispersia datelor, coeficientul de corelaţie a datelor, eliminarea datelor eronate

Tab. 5.1. *(Continuare)*

6. Calcul numeric cu polinoame	Evaluarea polinoamelor	Calculul valorii polinomului
	Operaţii aritmetice cu polinoame	Operaţii cu şi între polinoame
	Calculul derivatei	Calculul ceficienţilor polinomului derivată
	Calculul rădăciniilor/coeficienţilor	
	Polinoame pentru interpolarea şi aproximarea datelor	- interpolarea liniară/tabelară uni- şi bi-dimensională - interpolarea multiplă (cubică, spline) - aproximarea datelor prin regresie liniară şi polinomială
7. Calcule asupra funcţiilor	Minimizarea funcţiilor	- de o singură variabilă - de două sau mai multe variabile
	Integrarea şi derivarea funcţiilor	- derivarea funcţiilor - integrarea numerică a funcţiilor (metoda trapezelor, Simpson)
	Calcul diferenţial	- rezolvarea ecuaţiilor diferenţiale - rezolvarea sistemelor de ecuaţii diferenţiale
8. Calcule cu funcţii matematice speciale	Transformarea Fourier Funcţii euleriene Funcţii Bessel	-transformata Fourier directă şi inversă - funcţia Gamma, funcţia Beta - funcţiile lui Bessel de speţa a I-a şi a II-a
9. Calcul vectorial	Operaţii elementare cu vectori	- produsul scalar a doi vectori - produsul vectorial a doi vectori

5.2. Calcul matriceal

5.2.1 Operaţii cu matrice

Operatorii de calcul cu matrice au simbolistica prezentată în secţiunea 4.3, iar regulile de calcul cu matrice sunt cele din algebra matricelor.

a) Adunarea şi scăderea matricelor

- adunarea/scăderea a două matrice presupune aceleaşi dimensiuni ale matricelor,
- operaţia se face element cu element (ca la tablouri, operatorii + şi – fiind aceiaşi în ambele cazuri),
- un scalar poate opera cu orice matrice.

b) Înmulţirea matricelor

$$Z(i,j) = \sum_k X(i,k) \cdot Y(k,j)$$

- numărul de coloane al matricii X (prima) trebuie să fie egal cu numărul de linii al matricei Y (a doua),
- operaţia de înmulţire cu un scalar este similară cu cea de la tablouri.

c) Împărţirea matricelor

- Împărţirea la dreapta (cu operatorul /): Z=X/Y $\Leftrightarrow$ $X*Y^{-1}$
- Împărţirea la stânga (cu operatorul \): Z=X\Y $\Leftrightarrow$ $X^{-1}*Y$

Operaţia de inversare a matricelor se face cu funcţia Matlab `inv`, astfel: Y^{-1}=`inv(Y)`, respectiv X^{-1}=`inv(X)` cu condiţia ca *determinanţii matricelor argument să fie nenuli*. Determinantul se calculează cu funcţia Matlab `det` şi există doar în mulţimea matricelor pătrate. (Deci şi operaţia de inversare este posibilă numai în mulţimea matricelor pătrate). Prin urmare, la împărţire relaţia între dimensiunile matricelor X,Y trebuie să respecte de asemenea regula de la înmulţire.

d) Ridicarea la putere

Cu instrucţiunea `Z=X^p`, unde $X \in M_n$ (mulţimea matricelor pătrate de ordinul n) şi exponentul p scalar se obţine:

X^p=X*X*…*X   dacă $p \in Z_+$
X^p=inv(X)*inv(X)*…   dacă $p \in Z_-$

Exemplu. Să se calculeze A^p pentru

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

şi `p=-2`.

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

A =
     1     2
     3     4

>> X=A^-2

X =
    5.5000   -2.5000
   -3.7500    1.7500
```

e) Transpunerea matricelor

Cu instrucţiunea de forma Z=Y’ se obţine efectul înlocuirii liniilor cu coloanele matricei Y. Altfel spus liniile matricei Y devin coloanele matricei transpuse Z. Acelaşi efect se obţine şi asupra tablourilor de date:

$$(m \times n) \rightarrow (n \times m)$$

Observaţie. Dacă elementele matricei Y sunt numere reale operaţia de transpunere efectuează Z(i,j)=Y(j,i). Dacă elementele matricei Y sunt numere complexe (z=a+bi) operaţia de transpunere returnează conjugata transpusei, adică:

```
Z(i,j)=conj(Y(j,i))=real(Y(j,i)-i*imag(Y(j,i)))
```

Notă. În Matlab operaţiile cu matrice pot fi efectuate şi în maniera în care operanzii se consideră *tablouri de date*, folosind operatorii precedaţi de un punct (vezi secţiunea 4.3). În acest caz nu mai există restricţiile de înmulţire, împărţire şi ridicare la putere specifice algebrei matricelor, însă se menţin restricţii de ordin dimensional cu privire la corespondenţa elementelor omoloage în tablourile de date.

În tabelul 5.2 sunt date mai multe situaţii de calcul cu matrici şi tablouri ca exemple de operare cu evidenţierea erorilor ce pot să apară datorită incompatibilităţii dimensionale a operanzilor.

Tab. 5.2. Exemple de de cazuri de operare cu matrici/tablouri

<table>
<tr><th rowspan="2">Formatul operanzilor (returnat de Matlab)</th><th colspan="2">Rezultat/Mesaj</th></tr>
<tr><th>Operaţii cu matrice</th><th>Operaţii cu tablouri</th></tr>
<tr><td colspan="3">Ambii operanzi sunt matrice/tablouri dreptunghiulare</td></tr>
<tr><td rowspan="6">X =
1 2 3
4 5 6

Y =
10 20 30
40 50 60</td><td>X+Y =
11 22 33
44 55 66</td><td>X+Y =
11 22 33
44 55 66</td></tr>
<tr><td>X*Y
??? Error using ==> *
Inner matrix dimensions must agree.</td><td>X .*Y =
10 40 90
160 250 360</td></tr>
<tr><td>X/Y =
0.1000 0.0000
-0.0000 0.1000</td><td>X ./Y =
0.1000 0.1000 0.1000
0.1000 0.1000 0.1000</td></tr>
<tr><td>X\Y =
10.0000 5.0000 -0.0000
0 0 0
0.0000 5.0000 10.0000</td><td>X .\Y =
10 10 10
10 10 10</td></tr>
<tr><td>X^2
??? Error using ==> ^
Matrix must be square.</td><td>X .^2 =
1 4 9
16 25 36</td></tr>
<tr><td>X^Y
??? Error using ==> ^
Matrix dimensions must agree.</td><td>X.^Y =
1.0e+046 *
0.0000 0.0000 0.0000
0.0000 0.0000 4.8874</td></tr>
</table>

Tab. 5.2. *(Continuare)*

<table>
<tr><th rowspan="2">Formatul operanzilor (returnat de Matlab)</th><th colspan="2">Rezultat/Mesaj</th></tr>
<tr><th>Operații cu matrice</th><th>Operații cu tablouri</th></tr>
<tr><td colspan="3">Un operator este pătrat</td></tr>
<tr><td rowspan="6">X =
1 2 3
4 5 6
7 8 9

Y =
10 20 30
40 50 60</td><td colspan="2">X+Y

??? Error using ==> +
Matrix dimensions must agree.</td></tr>
<tr><td>X*Y

??? Error using ==> *
Inner matrix dimensions must agree.</td><td>X.*Y

??? Error using ==> .*
Matrix dimensions must agree.</td></tr>
<tr><td>Y*X =

300 360 420
660 810 960</td><td>Y.*X

??? Error using ==> .*
Matrix dimensions must agree.</td></tr>
<tr><td>X/Y =

0.1000 0.0000
-0.0000 0.1000
-0.1000 0.2000</td><td>X ./Y

??? Error using ==> ./
Matrix dimensions must agree.</td></tr>
<tr><td>X\Y

??? Error using ==> \
Matrix dimensions must agree.</td><td>X .\Y

??? Error using ==> .\
Matrix dimensions must agree.</td></tr>
<tr><td>X^2 =

30 36 42
66 81 96
102 126 150

X^3 =

468 576 684
1062 1305 1548
1656 2034 2412</td><td>X .^2 =

1 4 9
16 25 36
49 64 81

X.^3 =

1 8 27
64 125 216
343 512 729</td></tr>
</table>

Tab. 5.2. *(Continuare)*

<table>
<tr><th rowspan="2">Formatul operanzilor (returnat de Matlab)</th><th colspan="2">Rezultat/Mesaj</th></tr>
<tr><th>Operații cu matrice</th><th>Operații cu tablouri</th></tr>
<tr><td colspan="3">Ambii operatori sunt matrici pătrate</td></tr>
<tr><td rowspan="5">X =
1 2 3
4 5 6
7 8 9

Y =
10 20 30
40 50 60
70 80 90</td><td colspan="2">X+Y =

11 22 33
44 55 66
77 88 99</td></tr>
<tr><td>X*Y =

300 360 420
660 810 960
1020 1260 1500</td><td>X .*Y=

10 40 90
160 250 360
490 640 810</td></tr>
<tr><td>X/Y
Warning: Matrix is singular to working precision.

ans =
Inf Inf Inf
Inf Inf Inf
Inf Inf Inf</td><td>X ./Y

ans =

0.1000 0.1000 0.1000
0.1000 0.1000 0.1000
0.1000 0.1000 0.1000</td></tr>
<tr><td>X\Y
Warning: Matrix is close to singular or badly scaled.

Results may be inaccurate. RCOND = 2.055969e-018.

ans =

10.0000 32.0000 -10.0000
0 -54.0000 20.0000
0 32.0000 0</td><td>X .\Y =

10 10 10
10 10 10
10 10 10</td></tr>
<tr><td>X^Y
??? Error using ==> ^
Matrix dimensions must agree.</td><td>X .^Y =

1.0e+085 *
0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
0.0000 0.0000 7.6177</td></tr>
</table>

Tab. 5.2. *(Continuare)*

<table>
<tr><th rowspan="2">Formatul operanzilor (returnat de Matlab)</th><th colspan="2">Rezultat/Mesaj</th></tr>
<tr><th>Operații cu matrice</th><th>Operații cu tablouri</th></tr>
<tr><td colspan="3">Alte situații</td></tr>
<tr><td rowspan="2">X =
1 2 3
4 5 6
7 8 9

Y =
0 20 30
40 50 60
70 80 100</td><td>X/Y =

0.0600 -0.0800 0.0600
0 0.1000 0
-0.0600 0.2800 -0.0600</td><td>X./Y
Warning: Divide by zero.
(Type "warning off MATLAB: divideByZero" to suppress this warning.)

ans =
Inf 0.1000 0.1000
0.1000 0.1000 0.1000
0.1000 0.1000 0.0900</td></tr>
<tr><td>X\Y

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 2.055969e-018.

ans =
1.0e+016 *
4.5036 0.0000 -4.5036
-9.0072 -0.0000 9.0072
4.5036 0.0000 -4.5036</td><td>X.\Y

ans =
0 10.0000 10.0000
10.0000 10.0000 10.0000
10.0000 10.0000 11.1111</td></tr>
<tr><td rowspan="2">X =
0 2 3
4 5 6
7 8 10

Y =
0 20 30
40 50 60
70 80 100</td><td>X/Y =

0.1000 0 0
0 0.1000 0
0.0000 -0.0000 0.1000</td><td>X./Y
Warning: Divide by zero.
(Type "warning off MATLAB: divideByZero" to suppress this warning.)

ans =
NaN 0.1000 0.1000
0.1000 0.1000 0.1000
0.1000 0.1000 0.1000</td></tr>
<tr><td>X\Y =

10.0000 -0.0000 0.0000
0 10.0000 -0.0000
0 -0.0000 10.0000</td><td>X.\Y
Warning: Divide by zero.
(Type "warning off MATLAB: divideByZero" to suppress this warning.)

ans =
NaN 10 10
10 10 10
10 10 10</td></tr>
</table>

Exerciţiu propus. Pentru evidenţierea modului de lucru cu matrice şi tablouri, fiind date A=[1 2 3] , B=[4 5 6] , p=2 să se efectueze următoarele calcule:

C=A+B	F=p-A	H=A.*p	K=A./p
D=A-B	P=A.^p	S=A.’	L=p./A
E=A-p	G=A.*B	I=p.*A	M=A.\B
O=A.^B	R=p.^A	J=A./B	N=A.\p

5.2.2 Vectorizarea calculelor

Datorită faptului că Matlab este conceput ca un mediu de programare orientat pe structuri de date de tip tablou (matrice, vectori), operaţiile cu vectori şi matrice sunt executate mai rapid (cu circa un ordin de mărime) decât operaţiile secvenţiale.

Vectorizarea calculelor constă în înlocuirea secvenţelor de program algoritmice specifice operaţiilor cu scalari efectuate (secvenţial) în cadrul unor cicluri, cu operaţii direct adresate operanzilor de tip matrice. Acest lucru nu este dificil de realizat, întrucât Matlab-ul priveşte implicit orice variabilă ca pe o matrice. Singura restricţie care se pune la vectorizarea calculelor este legată de problema în sine, în sensul dacă aceasta impune în mod explicit o rezolvare algoritmică secvenţială.

Astfel, în general acolo unde este posibil pentru a exploata pe deplin puterea de calcul a Matlab-ului, ciclurile `for` şi `while` trebuie convertite în operaţii cu vectori sau matrice.

Exemplu de vectorizare a calculelor într-o secvenţă de program simplă :

Program bazat pe cicluri (algoritm clasic)	**Program vectorizat**	**Observaţii**

`x=0:0.001:10;`
`n=length(x) ;`
`  for i=1:n`
`  y(i)=sin(x(i));`
`   nd`
`plot(x,y)`
	`X=0:0.001:10;`
	`Y=sin(X);`
	`plot(X,Y)`
		Prin vectorizare, programarea devine mai productivă (cod mai puţin) şi se obţine o viteza de calcul mai mare (timp de execuţie mai scurt).

Observaţie. Dacă vectorizarea calculelor nu este posibilă într-o anumită secvenţă de program, se poate realiza totuşi o creştere semnificativă a vitezei de execuţie a programului prin prealocarea (predefinirea) unor vectori sau matrici cu elemente nule (cu ajutorul funcţiei `zeros`), în care datele calculate vor fi reţinute.

Exemplu:

```
x=0:0.001:10 ;
y=zeros(1,1000)
for i=1:1000
  y(i)=sin(x(i));
end
```

Exemplu. Se evaluează diferite moduri de definire a unei matrice de dimensiuni mari (400×500) cu elemente aleatoare, cu punerea în evidenţă a timpului de calcul (`elapsed_time`) şi a cantităţii de cod scris.

Algoritm clasic cu cicluri	Algoritm cu cicluri cu prealocarea spaţiului	Cu o funcţie predefinită.

`clear`
`tic`
`for i=1:400`
`    for j=1:500`
`     A(i,j)=randn(1);`
`    end`
`end`
`toc`
`disp(A);`
	`clear`
	`A=zeros(400,500);`
	`tic`
	`for i=1:400`
	`    for j=1:500`
	`    A(i,j)=randn(1);`
	`    end`
	`end`
	`toc`
		`clear`
		`tic`
		`A=randn(400,500);`
		`toc`

`elapsed_time =`
 	
`4.0550`
	`elapsed_time =`
	
	`1.5320`
		`elapsed_time =`
		
		`0.0100`

Se remarcă efectul vectorizării calculelor prin lucrul direct cu matrici asupra reducerii considerabile a timpului de calcul.

5.2.3 Manipularea matricelor

Manipularea matricelor presupune efectuarea unor operaţii relativ la elementele acestora utile în rezolvarea problemelor formulate matriceal. Principalele operaţii de manipulare a matricelor sunt extragerea (prin selecţie) a sbmatricelor, operaţia complementară de asamblare (reconstruire) a matricelor, redimensionarea matricelor date şi rotirea matricelor.

a) Extragerea submatricelor prin indici

O matrice nouă B se poate obţine dintr-o matrice definită anterior A cu instrucţiuni de forma:

```
B=A(indici_pentru_linie , indici_pentru_coloana)
```

Indicii pot fi **scalari** sau **vectori**. Astfel se pot face, de exemplu, extrageri de matrici B dintr-o matrice A definită generic:

$$A=\begin{bmatrix} x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \end{bmatrix},$$

de forma următoare:

`B=A(1:5,3)` ⇔ extrage din matricea A o submatrice formată din elementele acesteia situate pe liniile 1-5 şi pe coloana 3, matricea rezultată având dimensiunea 5x1

$$A=\begin{bmatrix} x & x & \mathbf{x} & x & x & x \\ x & x & \mathbf{x} & x & x & x \\ x & x & \mathbf{x} & x & x & x \\ x & x & \mathbf{x} & x & x & x \\ x & x & \mathbf{x} & x & x & x \\ x & x & x & x & x & x \end{bmatrix} \rightarrow B=\begin{bmatrix} x \\ x \\ x \\ x \\ x \end{bmatrix}$$

sau

`B=A(1:5,7:10)` ⇔ extrage din matricea A submatricea compusă din elementele situate pe liniile 1-5 şi pe colonele 7-10; matricea rezultată având deci dimensiunea 5x4;

$$A=\begin{bmatrix} x & x & x & x & x & x & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & x \\ x & x & x & x & x & x & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & x \\ x & x & x & x & x & x & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & x \\ x & x & x & x & x & x & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & x \\ x & x & x & x & x & x & \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} & x \\ x & x & x & x & x & x & x & x & x & x & x \end{bmatrix} \rightarrow B=\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}$$

sau

`B=A(:,3)` ⇔ extrage din A submatricea care conţine elementele de pe **toate** liniile coloanei 3, dimensiunea matricei rezultate fiind (*nr. de linii al matricei A×1*):

$$A=\begin{bmatrix} x & x & \mathbf{x} & x & x & x & x & x & x & x & x \\ x & x & \mathbf{x} & x & x & x & x & x & x & x & x \\ x & x & \mathbf{x} & x & x & x & x & x & x & x & x \\ x & x & \mathbf{x} & x & x & x & x & x & x & x & x \\ x & x & \mathbf{x} & x & x & x & x & x & x & x & x \\ x & x & \mathbf{x} & x & x & x & x & x & x & x & x \end{bmatrix} \rightarrow B=\begin{bmatrix} x \\ x \\ x \\ x \\ x \\ x \end{bmatrix}$$

`B=A(:,:)` ⇔ returnează chiar matricea originală A;

Observaţie. Operaţiile de extracţie solicitate trebuie să fie în acord cu dimensiunile matricelor.

Utilizarea *indicilor vectoriali* permite operaţii „chirurgicale” mult mai flexibile asupra matricelor. Astfel, dacă indicii pentru linii - x, respectiv pentru coloane - y sunt vectori cu componente întregi, atunci, instrucţiunea:

```
B=A(x,y)
```

crează submatrici B formate din elementele liniilor x ale coloanelor y ale matricei A.

De exemplu:

`B=A([1,4],[2,4:6])` selectează din matricea A elementele situate pe liniile 1 si 4 corespunzător coloanelor 2, 4, 5 şi 6, dimensiunea matricei astfel selectate fiind <u>2x4</u>.

$$A=\begin{bmatrix} x & \mathbf{x} & x & \mathbf{x} & \mathbf{x} & \mathbf{x} & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & \mathbf{x} & x & \mathbf{x} & \mathbf{x} & \mathbf{x} & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x & x \end{bmatrix} \rightarrow B=\begin{bmatrix} x & x & x & x \\ x & x & x & x \end{bmatrix}$$

De exemplu, cu instrucţiunea `C([1,3],[2,3])=D(:,1:2)` se realizează o înlocuire în matricea C a elementelor aparţinând liniilor 1 şi 3 şi coloanelor 2 şi 3 cu elemente ale matricei D situate pe coloanele 1, 2, după cum urmează:

```
C =
     1     2     3
     4     5     6
     7     8     9
    10    11    12

D =
    20    30    40
    50    60    70
```

se obţine:

```
C =
     1    20    30
     4     5     6
     7    50    60
    10    11    12
```

Alte *exemple* aplicative

- Inversarea coloanelor unei matrice cu dimensiunea m×n: se face cu instrucţiunea de forma `A=A(:,n:-1:1)`, de exemplu:

```
A =
    20    30    40
    50    60    70
```

Pentru `A=A(:,3:-1:1)` rezultă:

```
A =
     40    30    20
    70    60    50
```

- Inversarea liniilor: `A=A(n:-1:1,:)`

```
A=A(2:-1:1,:)

A =
    50    60    70
    20    30    40
```

- Inversarea liniilor şi a coloanelor:

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

A =
    40    30    20
    70    60    50
```

- Transformarea unei matrice într-un vector coloană a=A(:) observând că aranjarea elementelor în noua formă se face parcurgând coloanele succesiv;

```
A =
  40    30    20
  70    60    50

a=A(:)

a =
   40
   70
   30
   60
   20
   50
```

Notă. Reciproc, transformarea unui vector într-o matrice se poate face doar dacă matricea respectiva există deja (este definită ca dimensiune).

De exemplu, se defineşte matricea `X=zeros(2,3)` şi un vector coloană `b` cu (2×3) elemente:

```
X =
   0      0      0
```

```
  0      0      0

b =
   40
   70
   30
   60
   20
   50
```

Se cere transformarea vectorului b în structura matricei X.

```
X(:)=b

X =
    40    30    20
    70    60    50
```

Cu instrucţiunea `A(:)=xmin:xmax` rezultă o aşezare pe coloane a elementelor vectorului `xmin:xmax` cu pasul 1 (conform dimensiunilor m×n ale matricei A definită în prealabil), de forma:

A=	xmin		:
	xmin+1		:
	xmin+2	…	xmax-2
	:		xmax-1
	:		xmax

De exemplu:

```
X(:)=1:6

X =
     1     3     5
     2     4     6
```

Matricea X a fost definită în prealabil cu `X=zeros(2,3)`.

Notă. Pentru operaţia de extragere de vectori cu elemente decupate din alţi vectori se utilizeaza formele:

`b=a(j:k)` => selectează din `a` elementele [j, j+1,..., k] ale unui vector dacă j<k, iar în caz contrar vectorul rezultat este gol;

sau

`b=a(j:i:k)` => selectează elementele j, j+i, j+2i, ... , k; vectorul rezultat este gol dacă i>0 si j>k sau dacă i<0 si j<k.

b) Extragerea submatricelor prin vectori cu elemente 0 – 1

Fie A o matrice $m \times n$ şi L un vector de lungime *m* fiind un rezultat al unor operaţii logice, deci cu elemente 0 şi 1.

Cu instrucţiunea `B=A(L,:)` se realizează o selecţie a tuturor elementelor matricei A de pe liniile sale pentru care elementul corespunzător ca poziţie în L este 1 (adevărat logic).

Exemplu.

Fie L=[0 1 0] un vector de extragere obţinut în urma unor operaţii logice sau relaţionale.

Instrucţiunea `B=A(L,:)` are ca efect eliminarea liniilor 1 şi 3 ale matricei A, adică:

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 7 & 0 & -5 \\ 0 & 5 & 8 \end{bmatrix} \Leftrightarrow \text{B=[7 0 -5]}$$

În acest exemplu vectorul de selecţie a avut efect asupra liniilor matricei A deoarece a fort situat pe poziţia de specificare a liniilor.

Exemplu de aplicare pentru o matrice M definită astfel:

```
M =
      1      2      3      4
      5      6      7      8
      9     10     11     12
```

Cu secvenţa de instrucţiuni ce presupune operaţii logice următoare:

```
X=randn(1,4)
L=X>0
```

```
N=M(:,L)
```

Se generează vectorul de selecţie (aleator) în 0 şi 1:

```
X =
  1.1909    1.1892   -0.0376    0.3273

L =
     1     1     0     1
```

Selecţia obţinută asupra matricei M este:

```
N =
     1     2     4
     5     6     8
     9    10    12
```

Efectul operaţiei a constat în acest caz în extragerea (eliminarea) coloanei a treia, corespunzătoare elementului 0 al vectorului de extracţie. Vectorul de selecţie L fiind amplasat pe poziţia specificatorilor de coloane.

c) Asamblarea matricelor mari

O matrice se poate constitui cu ajutorul altor matrice de dimensiuni inferioare, ca o compunere a acestora, cu condiţia *consistenţei dimensionale*, adică matricele care se asamblează să constituie blocuri care se integrează compact în tabloul matricei mari. Acest lucru se poate înţelege cel mai bine printr-un exemplu.

Exemplu.

Fie o matrice aleatoare pătrată A(3). Să se construiească o nouă matrice formată din A, din matricea identitate, din matricea unitate şi din pătratul matricei A. Matricea rezultat poate fi compusă cu următoarea structură:

```
C=[rand(3) ones(3);eye(3) A^2]
```

```
>> A=rand(3)
A =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214

>> C=[A ones(3);eye(3) A^2]
C =
    0.9501    0.4860    0.4565    1.0000    1.0000    1.0000
    0.2311    0.8913    0.0185    1.0000    1.0000    1.0000
    0.6068    0.7621    0.8214    1.0000    1.0000    1.0000
    1.0000         0         0    1.2921    1.2428    0.8176
         0    1.0000         0    0.4369    0.9208    0.1372
         0         0    1.0000    1.2512    1.6002    0.9658
```

d) Redimensionarea matricelor

Redimensionarea matricelor constă în modificarea numărului de linii şi/sau de coloane având ca efect conservarea numărului de elemente şi redistribuirea acestora. Transformarea unei matrice date de dimensiuni (m×n) într-o nouă matrice cu dimensiunile diferite (p×q) este posibilă numai dacă există condiţia: (m×n)=(p×q).

Funcţia Matlab care realizează această operaţie de transformare, cu respectarea de către utilizator a condiţiei de mai sus, are sintaxa următoare:

```
Matrice_noua=reshape(Matrice_data,p,q)
```

Exemplu.

Fie matricea (3×4): A=[1 0 -1 2; 10 3 4 0; -7 -8 -9 5]. Să se redimensioneze aceasta în toate modurile posibile.

Rezolvare. Se identifică toate descompunerile posibile ale lui 12 şi se aplică succesiv funcţia `reshape` cu argumentele corespunzătoare astfel:

	p×q =12	
	p	q
Cazul dat	3	4
Variante posibile de redimensionare	4	3
	6	2
	2	6
	12	1
	1	12

În continuare se aplică funcţia şi se pun în evidenţă rezultatele obţinute în modul de lucru linie de comandă.

```
>> A=[1 0 -1 2; 10 3 4 0; -7 -8 -9 5]

A =
    1    0   -1    2
   10    3    4    0
   -7   -8   -9    5

>> M1=reshape(A,4,3)

    M1 =
        1    3   -9
       10   -8    2
       -7   -1    0
        0    4    5

>> M2=reshape(A,6,2)

    M2 =
        1   -1
       10    4
       -7   -9
        0    2
        3    0
       -8    5

>> M3=reshape(A,2,6)

    M3 =
        1   -7    3   -1   -9    0
       10    0   -8    4    2    5
```

```
>> M2=reshape(A,12,1)

M2 =

       1
      10
      -7
       0
       3
      -8
      -1
       4
      -9
       2
       0
       5


>> M2=reshape(A,1,12)

M2 =

 1   10   -7   0   3   -8   -1   4   -9   2  0   5
```

Observaţie. Rearanjarea elementelor prin aplicarea funcţiei `reshape` se face parcurgând tabloul/matricea pe coloane.

e) Rotirea matricelor

Rotirea matricelor este o operaţie care conservă numărul elementelor având ca efect redistribuirea acestora în conformitate cu sensul de rotaţie specificat prin numele funcţiei sau prin argument/parametru de intrare. Matlab pune la dispoziţie o colecţie de funcţii destinate rotirii/inversării în diferite moduri a elementelor unei matrice, după cum urmează:

- Rotirea matricei în jurul unei coloane sau linii se efectuează cu funcţiile `fliplr` – cu efect ce roteşte de la stânga la dreapta matricea (pivotează în jurul ultimei coloane) şi `flipud` – cu efect ce roteşte de sus în jos matricea (pivotează în jurul ultimei linii).

- Rotirea unei matrice cu 90 de grade în jurul unui element se face în sens trigonometric o singură dată sau de k ori succesiv dacă acesta se specifică ca argument, astfel:

```
B=rot90(A)
B=rot90(A,k),
```

unde k este multiplu de 90 grade cu care se roteşte matricea, în sens trigonometric, dacă k este pozitiv şi în sens orar dacă k este negativ.

Exemple de aplicare:

```
A =
    1    0   -1    2
   10    3    4    0
   -7   -8   -9    5

>> X=fliplr(A)

X =
    2   -1    0    1
    0    4    3   10
    5   -9   -8   -7

>> Y=flipud(A)

Y =
   -7   -8   -9    5
   10    3    4    0
    1    0   -1    2
```

```
A =
    1    0   -1    2
   10    3    4    0
   -7   -8   -9    5

>> B=rot90(A)

B =
    2    0    5
   -1    4   -9
    0    3   -8
    1   10   -7

>> B=rot90(B,-1)

B =
    1    0   -1    2
   10    3    4    0
   -7   -8   -9    5
```

Importanţa practică a operaţiilor de rotire a matricelor se regăseşte în problemele de algebră liniară, în aplicaţii de tehnica transmiterii informaţiei şi teoria codurilor, în prelucrarea digitală a imaginilor, etc.