4. PROGRAMAREA ÎN MEDIUL MATLAB

4.1 Elemente de sintaxa

Sintaxa Matlab-ului este mai simplă în comparaţie cu cea a altor limbaje de programare, fiind mai apropiată de modul de descriere a modelelor matematice pentru rezolvarea problemelor. Cuvintele cheie rezervate pentru funcţii şi instrucţiuni în Matlab sunt în general comune cu cele întâlnite în alte limbaje de programare de nivel înalt.

a) Reguli sintactice generale

- Entitatea de bază pentru lucru este *expresia Matlab*, care poate fi sub forma unei instrucţiuni, declaraţie de funcţie, apel de funcţie, nume de variabilă, etc.;
- Orice instrucţiune se termină în general cu apăsarea tastei "Enter", ceea ce determină fie execuţia ei în modul de lucru linie de comandă, fie trecerea pe linia următoare în procesul de editare a programului;
- Dacă ultimul caracter din instrucţiune este ";" aceasta este executată, dar tipărirea (afişarea) pe ecran este suprimată;
- În cazul expresiilor lungi, interpretarea lor ca o singură linie de program se face prin utilizarea unei succesiuni de trei puncte "…" urmată de apăsarea tastei "Enter" determinând astfel continuarea ei pe linia următoare;
- Spaţiile din vecinătatea semnelor = , + , – şi respectiv din vecinătatea oricărui simbol de operator sau paranteză sunt opţionale şi nu conduc la erori de sintaxă;
- Numele variabilelor au ca prim caracter o literă (urmată de litere, cifre sau caracterul "_");
- Numele variabilei poate fi oricât de lung, dar în Matlab se iau în considerare numai primele N=63 caractere. Cu funcţia `namelengthmax` se poate afla valoarea de mai sus. Această limită se aplică oricărui nume de funcţie sau structură de date în Matlab.
- Implicit, Matlab-ul este senzitiv la literele mari şi mici (adică face distincţie între caracterele mari şi cele mici, de exemplu : `A` şi `a` nu reprezintă aceeaşi variabilă);
- Numele de funcţii se scriu obligatoriu cu litere mici.

b) Numere şi formate

În Matlab, ca şi în oricare limbaj de programare sunt două aspecte legate de numere: reprezentarea internă a datelor (stocarea numerelor in memorie) şi afişarea (scrierea) numerelor pe dispozitivele standard de ieşire. Din punctul de vedere al reprezentării şi prelucrării numerelor, remarcăm că în Matlab toate calculele se efectuează în *dublă precizie*, iar în memoria internă, toate numerele sunt stocate în formatul `long`, (în conformitate cu standardul IEEE de reprezentare în virgulă mobilă folosind 16 cifre semnificative). Limitele (domeniul) numerelor reale folosite în Matlab sunt de ordinul: 10^{-308} respectiv 10^{308}.

Clasele de reprezentare internă a datelor numerice în Matlab sunt următoarele:

Tipul	Clasa	Dimensiunea binară
Intregi fără semn	uint8 uint16 uint32 uint64	1 bytes 2 bytes 4 bytes 8 bytes
Întregi cu semn	int8 int16 int32 int64	1 bytes 2 bytes 4 bytes 8 bytes
Dublă precizie	double	8 bytes
Simplă precizie	single	4 bytes

Fiind un mediu de calcul ingineresc, clasele de date numerice care se folosesc pentru reprezentarea semnalelor, a mărimilor fizice, trebuie să fie acceptate de diverse funcţii de prelucrare. Dacă această compatibilitate nu există, atunci aplicaţia respectivă raportează eroare. Situaţia este întâlnită în special în aplicaţiile de modelare cu Simulink, motiv pentru care a fost creată o funcţie bloc pentru compatibilizarea (transformarea) tipurilor de date: `Data Type Conversion`.

În ceea ce priveşte scrierea şi afişarea numerelor, aceasta poate fi controlată cu funcţia de control general `format` *`optiune`*. Opţiunile pentru diverse formate admise în Matlab pot fi aflate cu `help format`. Subliniem că această funcţie afectează doar modul de afişare a numerelor, nu şi precizia calculelor.

Matlab utilizează implicit pentru scrierea şi afişarea numerelor notaţia convenţională cu punct zecimal şi semn (formatul `short` cu 5 cifre

sau `long` cu 15 cifre), de exemplu: `-0.00034589122245`. Notația ştiințifică, cu puteri ale lui zece (formatul `short e` respectiv `long e`) include caracterul `e` pentru exponent şi simbolul plus sau minus pentru semnul acestuia. Puterea este un factor de scală inclus ca sufix astfel: `-3.4589e-004`;

Alte formate de afişare sunt posibile cu opțiunile `hex` (hexazecimal), `rat` (sub formă de fracție), `bank` (cu două cifre zecimale), şi altele.

c) Declarații de variabile şi expresii

O *variabilă* se declară în Matlab prin definirea unei expresii de atribuire, de forma:

```
variabila=expresie
```

În Matlab, spre deosebire de alte limbaje nu este necesară declararea tipului sau dimensiunii variabilelor. Se spune că Matlab este un limbaj de expresii, orice expresie introdusă de utilizator fiind interpretată şi evaluată. Evaluarea expresiei produce *o matrice* care este afişată pe ecran şi atribuită variabilei respective. Când Matlab-ul primeşte un nume de variabilă, el crează automat variabila respectivă şi îi alocă un spațiu de memorie potrivit.

Chiar dacă numele variabilei şi semnul egal (“variabila=”) sunt omise, mediul Matlab crează automat o variabilă specială (generică) denumită “`ans`”, căreia îi asociază valoarea expresiei şi o afişează conform formatului definit. De exemplu:

Expresia tipărită:	`3/4`
Returnează:	`ans=0.7500`

O *expresie* Matlab este o colecție de simboluri conform sintaxei limbajului reprezentând expresii matematice ce pot include şi nume de funcții, declarații de variabile, atribuiri de valori, sau pur şi simplu valori numerice şi nenumerice.

d) Variabile speciale şi constante în Matlab

Variabilele speciale în Matlab sunt globale (accesibile global, adică “vizibile” în orice fişier cu extensia `m`) şi nu pot fi declarate de utilizator. Acestea, precum şi anumite constante sunt introduse în sintaxa funcțiilor Matlab pentru a returna valori scalare utile, după cum urmează:

ans – variabilă care operează (se autodeclară) automat, în care este returnat rezultatul unui calcul, atunci când expresia nu a avut asigurat un nume (de variabilă);

eps - variabilă permanentă în care este memorată eroarea relativă prin calcule efectuate în virgulă mobilă. Valoarea implicită este `2.220446049250313e-016`, dar poate fi redefinită după necesitățile de calcul;

pi - variabilă permanentă care are asignată valoarea `3.14159265358979`;

i – constantă folosită pentru definirea numerelor complexe reprezentând unitatea imaginară i=$\sqrt{-1}$. În acelaşi scop se foloseşte şi caracterul *j*.

inf - variabilă folosită pentru reprezentarea lui $+\infty$ (rezultatul împărțirii `1.0/0.0`);

NaN - variabilă folosită pentru reprezentarea lui Not-a-Number, ca rezultat al operației nedefinite `0.0/0.0`;

nargin - variabilă permanentă pentru testarea numărului argumentelor de intrare ce trebuie introduse pentru apelarea unei funcții;

nargout - variabilă permanentă pentru testarea numărului argumentelor de ieşire ale unei funcții;

computer - variabilă folosită pentru obținerea informațiilor referitoare la tipul calculatorului şi numărul maxim de elemente pe care le poate gestiona versiunea curentă de Matlab;

De exemplu: `[calculator,elemente]=computer`
Returnează: `calculator=PCWIN`
`elemente=2.147483647000000e+009`

realmax - cea mai mare valoare pozitivă în virgulă mobilă care poate fi folosită în calcule: `1.797693134862316e+308`;

realmin - cea mai mică valoare pozitivă în virgulă mobilă care poate fi folosită în calcule: `2.225073858507201e-308`;

version sau *ver* - funcții pentru determinarea versiunii Matlab respectiv a toolbox-urilor instalate sub el.

4.2. Structuri de date

Structura fundamentală de date cu care operează Matlab-ul este *matricea numerică rectangulară* de dimensiuni (m×n) cu elemente reale sau complexe. Matricea este un tablou de date bidimensional care reprezintă conceptul fundamental al *algebrei liniare*. După cum se cunoaşte, algebra liniară operează cu spaţii vectoriale (numite şi spaţii liniare) având ca obiectiv calculul cu vectori prin *transformări liniare* şi sisteme de ecuaţii liniare. Geometria analitică reprezintă o concretizare a algebrei liniare. Astfel, importanţa reprezentărilor vectoriale în ştiinţă şi inginerie, precum şi considerente legate de productivitatea calculelor au impus în Matlab următoarele structuri de date:

Structura de date	Forma	Observaţii
Tabloul de date (array)	Colecţie de date ordonată mono- sau bi-dimensional	Se foloseşte în afara cadrului algebrei liniare
Matricea (matrix)	Colecţie de date ordonată mono- sau bi-dimensional	Se foloseşte în contextul algebrei liniare
Vectorul	vector linie = matrice (1×n), sau vector coloană = matrice (n×1)	Cazuri particulare de matrice
Scalarul	matrice (1×1)	Orice valoare singulară este asimilată cu o matrice (1×1)

Elementele unei matrice se reprezintă ca un tablou în general cu *m* linii şi *n* coloane. Dintre notaţiile uzuale prin care se face referire la elementele unei matrice: A_{ij}, A[i,j], A(i,j), în Matlab este adoptată forma A(i,j), unde *i* semnifică numărul de liniei, iar *j* desemnează numărul coloanei, astfel că elementul matricei este identificat ca fiind cel de la intersecţia liniei *i* cu coloana *j*.

Alături de matrice ca tip fundamental de structură de date, în Matlab sunt recunoscute şi alte structuri de date mai generale. Acestea sunt :

- Tablourile multidimensionale
- Celula de tablouri
- Caractere şi şiruri de caractere (text)
- Structuri

Prezentăm mai jos în sinteză definiția şi proprietățile lor. Aspecte legate de sintaxa şi reprezentarea acestora vor fi subliniate în subcapitolul următor.

Alte structuri în Matlab	Definiția	Proprietăți
Tablouri multidimensionale	Colecții de date ordonate cu mai mult de două dimensiuni (tablouri N-dimensionale)	Dimensiunea N>2 Elemente de forma: A(i,j,k,l,…)
Celula de tablouri	O macrostructură de date de tip celulă multidimensională ale cărui elemente sunt tablouri	Celulele conțin obiecte diverse, în general tablouri de date, inclusiv matrici goale.
Caractere şi şiruri de caractere	Date sub formă de caractere ASCII	Setul de caractere ASCII tipăribile de bază au corespondent numeric în gama 32 :127
Structuri	Tablouri multidimensionale cu elemente specificate prin câmpuri text.	În denumirea structurii, separarea câmpurilor text se face cu punct.

4.2.1 Generarea vectorilor şi matricelor

Definirea matricelor simple formate din elemente singulare se face prin următoarele metode:

- introducere explicită a listei de elemente;
- generarea prin instrucțiuni şi funcții Matlab predefinite;
- crearea matricelor cu programe Matlab proprii;
- încărcarea din fişiere de date externe.

Elementele matricelor pot fi:

- numere reale sau complexe;
- orice expresie Matlab.

Pentru introducerea explicită a listei de elemente, în general se foloseşte formatul următor:

```
A=[enumerare elemente linie1;enumerare elemente linie2;...],
```

sau A se defineşte ca o matrice remarcabilă generată printr-o funcție Matlab de forma: `ones(m,n)`, `zeros(m,n)`, `rand(m,n)`, `hadamard(n)`, `toeplitz(c)`, etc., (vezi Tab. 4.1).

a) Matricea goală

În Matlab nu se declară tipuri de variabile, iar matricele se *autodimensionează* în timpul rulării programelor. Această operație este destul de cronofagă. Pentru a creşte viteza de execuție a programelor Matlab uneori se crează o matrice goală la începutul sesiunii de lucru, cu efect de prealocare a spațiului de memorie:

`X=[]` având dimensiunea 0x0

Cu funcția `isempty` se testează dacă o matrice este goală, în sensul că matricea nu are nici un element. De exemplu, `isempty(X)` returnează 1 (adevărat).

b) Matricea zero

Prin definiție: Z(i,j)=0 ; ∀i,j∈N. Această matrice se generează cu funcția `zeros` prin următoarele sintaxe posibile:

`Z=zeros(n)` matricea pătrată n x n
`Z=zeros(m,n)` matrice m x n

sau

`Z=zeros(size(A))` matricea zero având dimensiunea unei matrice A (care a fost definită în prealabil).

Exemplu:

```
>> Z=zeros(3)

Z =
     0     0     0
     0     0     0
     0     0     0
```

c) Matricea unitate

Prin definiție: U(i,j)=1 ; ∀i,j∈N

```
U=ones(n)
U=ones(m,n)
U=ones(size(A))
```

Exemplu:

```
>> U=ones(3)
U =
     1     1     1
     1     1     1
     1     1     1
```

d) Matricea identitate

Prin definiție: $I(i, j) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$. Această matrice se generează cu funcția `eye`.

```
I=eye(n)
I=eye(m,n)
I=eye(size(A))
```

Exemplu:

```
>> I=eye(3)
I =
     1     0     0
     0     1     0
     0     0     1
```

e) Matricea aleatoare

Există două funcții pentru generarea de numere (pseudo)aleatoare:
- `rand` – generează tablouri de numere aleatoare cu *distribuție uniformă* în intervalul (0,1);
- `randn` – generează tablouri de numere aleatoare cu *distribuție normală* (Gaussiană) de medie 0 şi varianță egală cu 1 - *N(0,1)*.

În general, repartiția numerelor aleatoare cu distribuție normală se exprimă ca $N(\mu,\sigma)$, unde μ este media aritmetică a datelor, iar σ este abaterea medie pătratică sau dispersia datelor. Varianța unui set de date se defineşte ca pătratul dispersiei:

$$\sigma^2 = \frac{\sum_{k=1}^{N}(x_k - \mu)^2}{N-1}$$

Funcțiile Matlab pentru generarea matricelor cu numere aleatoare se pot apela cu următoarele sintaxe:

```
Ru=rand(n)          Rn=randn(n)
Ru=rand(m,n)        Rn=randn(m,n)
Ru=rand(size(A))    Rn=randn(size(A))
```

Exemple:

```
>> R=rand(3)

R =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
```

```
>> R=randn(3)

R =
   -0.4326    0.2877    1.1892
   -1.6656   -1.1465   -0.0376
    0.1253    1.1909    0.3273
```

Observație. Pentru simularea experiențelor aleatoare care comportă aceleaşi condiții se generează serii aleatoare la care se controlează un parametru de inițializare al generatorului denumit *seed*.

Instrucțiunile `rand('seed')` şi `randn('seed')` returnează valoarea curentă a numărului *seed*, iar cu instrucțiunile `rand('seed',n)` respectiv `randn('seed',n)` se poate seta numărul *seed* la valoarea *n*, aşa cum se exemplifică în continuare:

```
>> rand('seed')

ans =
   931316785

>> rand('seed', 1000)
>> rand('seed')

ans =
        1000
```

4.2.2 Matrici speciale

În Matlab există un set de funcţii care generează matrici speciale (remarcabile) folosite în aplicaţii de algebră liniară şi în probleme de prelucrare a semnalelor. În tabelul 4.1 sunt prezentate o serie de funcţii Matlab pentru lucrul cu matrice speciale.

Tab. 4.1 Funcţii pentru matrici speciale

Matrice, definiţie	Sintaxa funcţiei	Exemplificare
Matricea diagonală şi operaţii cu diagonalele matricelor.	D=diag(X) D=diag(X,k) Specificarea lui k determină extragerea unei diagonale secundare. (Implicit k=0). k=0 diagonala principală, k>0 diagonala k de deasupra celei principale, k<0 indică diagonala k de sub cea principală,	A=[1 2 3;4 5 6;7 8 9] A = 1 2 3 4 5 6 7 8 9 >> B=diag(A) B = 1 5 9
Matricea triunghiulară (inferioară sau superioară).	Y=tril(X) Y=tril(X,k) Idem pentru funcţia `triu`. Specificarea lui k determină înlocuirea cu 0 a tuturor elementelor de deasupra respectiv de sub diagonala k.	>> A=ones(4) A = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 >> Y=tril(A) Y = 1 0 0 0 1 1 0 0 1 1 1 0 1 1 1 1
Matricea Hadamard este o matrice pătrată cu elemente 1 sau -1 aşezate astfel încât linia j este identică cu coloana j.	H2n=hadamard(n) n- este ordinul matricei care trebuie ales ca putere a lui 2.	>> H2n=hadamard(2) H2n = 1 1 1 -1 >> H4n=hadamard(4) H4n = 1 1 1 1 1 -1 1 -1 1 1 -1 -1 1 -1 -1 1

Tab.4.1. *(Continuare)*

Matricea Toeplitz se defineşte cu ajutorul unor vectori care definesc prima coloană şi eventual prima linie.	T=toeplitz(c) T=toeplitz(c,l) c- vectorul primei coloane l – vectorul primei linii Notă. Cei doi vectori definitorii pot să aibă dimensiuni diferite, rezultând astfel matrice Toeplitz dreptunghiulare. Primul element al fiecăruia dintre cei doi vectori trebuie să aibă aceeaşi valoare. În caz contrar va fi semnalat un avertisment (conflict diagonal), iar primul element al vectorului coloană va fi dominant.	>> c=[0 2 -1 4] c = 0 2 -1 4 >> T=toeplitz(c) T = 0 2 -1 4 2 0 2 -1 -1 2 0 2 4 -1 2 0 >> l=[10 20 0] l = 10 20 0 >> T=toeplitz(c,l) T = 0 20 0 2 0 20 -1 2 0 4 -1 2
Matricea Hankel este o matrice anti-simetrică şi anti-diagonală care se poate defini pe baza unui vector sau doi vectori generatori.	H=hankel(c,l) c- vectorul primei coloane l – vectorul primei linii Obs. Prima coloană este definită de primul vector argument, iar ultima linie de elementele celui de-al doilea vector argument. Apare avertisment dacă ultimul element al primei coloane nu coincide cu primul al ultimei coloane (conflict anti-diagonal). Elementul coloanei este dominant şi conflictul este rezolvat.	Cu aceiaşi vectori de definiție de mai sus se obține: H = 0 2 -1 2 -1 4 -1 4 20 4 20 0
Matricea Hilbert are elementele definite de relația $H(i,j)=\frac{1}{i+j-1}$ Este o matrice pătrată, slab condiționată.	H=hilb(n)	>> H2=hilb(2) H2 = 1.000 0.500 0.500 0.333 >> H3=hilb(3) H3 = 1.000 0.500 0.333 0.500 0.333 0.250 0.333 0.250 0.200

Tab.4.1. *(Continuare)*

Matricea Pascal conține elementele triunghiului lui Pascal (coeficienții binomiali ai descompunerii $(a+b)^n$ pentru orice număr natural n)	P=pascal(n) Obs. Elementele din partea stângă a antidiagonalei principale sunt numerele triunghiului lui Pascal. Funcția poate fi utilizată şi în alte sintaxe care permit aranjări speciale ale matricei Pascal.	>> P3=pascal(3) P3 = 1 1 1 1 2 3 1 3 6 >> P4=pascal(4) P4 = 1 1 1 1 1 2 3 4 1 3 6 10 1 4 10 20
Matricea Vandermonde se defineşte prin expresia $V(i,j)=c(i)^{n-j}$ unde c este vectorul de definiție asociat penultimei coloane având n elemente.	V=vander(c)	>> c=[0 1 2 3 4] c = 0 1 2 3 4 >> V=vander(c) V = 0 0 0 0 1 1 1 1 1 1 16 8 4 2 1 81 27 9 3 1 256 64 16 4 1
Matricea Wilkinson se defineşte prin relația următoare: $W(i,j)=\begin{cases}\frac{\lvert n-2i+1\rvert}{2}, & i=j\\ 1, & i=j+1\\ 0, & \text{in rest}\end{cases}$, unde $i\in[1,n]$	W=wilkinson(n)	>> W3=wilkinson(3) W3 = 1 1 0 1 0 1 0 1 1 >> W4=wilkinson(4) W4 = 1.50 1.00 0 0 1.00 0.50 1.00 0 0 1.00 0.50 1.00 0 0 1.0 1.50 >> W5=wilkinson(5) W5 = 2 1 0 0 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 0 0 1 2

Tab.4.1. *(Continuare)*

Tensorul Kroneker este o matrice formată prin produsul dintre două matrice în toate modurile posibile, astfel: K=[X(1,1)*Y X(1,2)*Y ... X(2,1)*Y ..]	K=kron(X,Y) Obs. Dacă X este o matrice [m×n] şi Y o matrice [l×p] atunci dimensiunea tensorului rezultant va fi [ml×np]. Nota. Produsul tensorial nu este comutativ	>> X=[1 2; 1 3; 0 -1] X = 1 2 1 3 0 -1 >> Y=[1 2; 1 -1] Y = 1 2 1 -1 >> K=kron(X,Y) K = 1 2 2 4 1 -1 2 -2 1 2 3 6 1 -1 3 -3 0 0 -1 -2 0 0 -1 1
Pătratul magic de ordinul n este o matrice pătrată construită cu întregii de la 1 la n^2, la care sumele elementelor de pe fiecare linie, coloană, diagonală sau anti-diagonală principală sunt egale.	A=magic(n), unde $n \geq 3$.	>> A3 = magic(3) A3 = 8 1 6 3 5 7 4 9 2 >> A4 = magic(4) A4 = 16 2 3 13 5 11 10 8 9 7 6 12 4 14 15 1
Companionul matriceal se defineşte în legătură cu un polinom astfel: prima linie este constituită din numere reprezentând câtul cu semn schimbat ai împărţirii coeficienţiilor polinomului în ordinea descreşterii puterilor de la (n-1) la coeficientul puterii n.	C=compan(p) unde: $p=[p_n\ p_{n-1}\ \ldots p_2\ p_1\ p_0]$ este vectorul coeficienţilor polinomului: $P(x)= p_n x^n+p_{n-1}x^{n-1}+\ldots$ $\ldots+ p_2x^2+ p_1x+ p_0$	>> p=[10 3 -2 5] p = 10 3 -2 5 >> C=compan(p) C = -0.30 0.20 -0.50 1.00 0 0 0 1.00 0

4.2.3 Alte structuri de date în Matlab

În continuare se exemplifică tipurile de structuri de date speciale prezentate succint în prima parte a acestui capitol.

a) Tablouri multidimensionale

Exemplu de generare a unui tablou 3-dimensional cu (3×4×5) = 60 elemente aleatoare:

```
>> R=randn(3,4,5)

R(:,:,1) =

   -0.4326    0.2877    1.1892    0.1746
   -1.6656   -1.1465   -0.0376   -0.1867
    0.1253    1.1909    0.3273    0.7258

R(:,:,2) =

   -0.5883    0.1139   -0.0956   -1.3362
    2.1832    1.0668   -0.8323    0.7143
   -0.1364    0.0593    0.2944    1.6236

R(:,:,3) =

   -0.6918   -1.5937   -0.3999    0.7119
    0.8580   -1.4410    0.6900    1.2902
    1.2540    0.5711    0.8156    0.6686

R(:,:,4) =

    1.1908   -0.1567   -1.0565    0.5287
   -1.2025   -1.6041    1.4151    0.2193
   -0.0198    0.2573   -0.8051   -0.9219

R(:,:,5) =

   -2.1707    0.6145    0.5913   -1.0091
   -0.0592    0.5077   -0.6436   -0.0195
   -1.0106    1.6924    0.3803   -0.0482
```

Exemplu pentru un tablou 4-dimensional cu (3×2×3×2) = 36 elemente aleatoare:

```
>> A=rand(3,2,3,2)

A(:,:,1,1) =

    0.3046    0.6822
    0.1897    0.3028
    0.1934    0.5417

A(:,:,2,1) =

    0.1509    0.8600
    0.6979    0.8537
    0.3784    0.5936

A(:,:,3,1) =

    0.4966    0.6449
    0.8998    0.8180
    0.8216    0.6602

A(:,:,1,2) =

    0.3420    0.5341
    0.2897    0.7271
    0.3412    0.3093

A(:,:,2,2) =

    0.8385    0.7027
    0.5681    0.5466
    0.3704    0.4449

A(:,:,3,2) =

    0.6946    0.9568
    0.6213    0.5226
    0.7948    0.8801
```

```
>> whos
  Name      Size                Bytes    Class

  A         4-D                   288    double array
  R         3x4x5                 480    double array
```

b) Celula de tablouri

Cu functia `cell` se generează o macrostrucură de date de tip tablou multidimensional. Astfel, se pot genera celule bidimensionale sau

multidimensionale ale căror obiecte pot fi matrici goale sau alte tablouri, după cum se arată în exemplele următoare:

Exemplu de celula pătratică de ordinul 4 de matrici vide.	Exemplu de celulă multidimensională (2×2 ×3) de matrici vide.

`>> C=cell(4)`
 	
`C =`
 	
`[] [] [] []`
`[] [] [] []`
`[] [] [] []`
`[] [] [] []`
 	
`>> whos`
 	
`Name Size Bytes Class`
`C 4x4 64 cell array`
	`>> A=cell(2,2,3)`
	
	`A(:,:,1) =`
	
	`[] []`
	`[] []`
	
	`A(:,:,2) =`
	
	`[] []`
	`[] []`
	
	`A(:,:,3) =`
	
	`[] []`
	`[] []`
	
	`>> whos`
	
	`Name Size Bytes Class`
	`A 2x2x3 48 cell array`

Se pot defini celule ca o colecţie de tablouri diverse incluzând elementele între acolade, conform sintaxei următoare :

```
>> M={C A ones(3) magic(2)}

M =
    {4x4 cell}    {2x2x3 cell}    [3x3 double]    [2x2 double]

>> whos
  Name        Size                    Bytes  Class
  A           2x2x3                      48  cell array
  C           4x4                        64  cell array
  M           1x4                       456  cell array
```

Regăsirea (referirea) unui obiect anume din structura celulei se face specificând numărul de ordine al acestuia în cadrul structurii între acolade. De exemplu, obiectul al doilea - matricea unitară se regăseşte astfel :

```
>> M{3}

ans =

     1     1     1
     1     1     1
     1     1     1
```

c) Caractere şi şiruri de caractere (text)

Aceste structuri de date împreună cu funcţiile de prelucrare a lor vor fi tratate în subcapitolul 4.5.

d) Structuri

În Matlab există de asemenea posibilitatea de a reprezenta structuri de date cu ajutorul unor specificatori de tip text separaţi prin punct ce desemnează fiecare o anumită componentă. Aceste structuri sunt de asemenea interpretate ca tablouri multidimensionale.

De exemplu, structura construită în continuare în mod succesiv este interpretată ca un singur obiect de mărime 1×1.

```
>> Structura.nume='Ionescu George'

Structura =
    nume: 'Ionescu George'

>> Structura.varsta= 47

Structura =
      nume: 'Ionescu George'
    varsta: 47

>> Structura.calificativ='FB'

Structura =
           nume: 'Ionescu George'
         varsta: 47
    calificativ: 'FB'
```

Ca orice obiect în Matlab, structurile sunt tablouri la care se pot însera elemente adiţionale. De pildă, la structura deja creată anterior mai adăugăm o matrice aleatoare pătrată cu dimensiunea (3×3), cu instrucţiunea:

```
>> Structura.R=rand(3)
```

astfel structura devine:

```
Structura =

           nume: 'Ionescu George'
         varsta: 47
    calificativ: 'FB'
              R: [3x3 double]
```

Diferite elemente pot fi constituite într-o structură de date printr-o singură declarație, folosind funcția `struct`. Astfel, cu instrucțiunea:

```
Structura=struct('nume','Ionescu George','varsta',47,'calificativ','FB','R', rand(3))
```

se obține aceeaşi structură de mai sus.

Apelul componentelor (obiectelor) acestei structuri de date se face prin invocarea numelui de variabilă folosind compunerea cu simbolul punct. Accesul la componentele structurii se realizează prin apeluri de forma:

```
>> Structura.R

ans =

    0.9501    0.4860
0.4565
    0.2311    0.8913
0.0185
    0.6068    0.7621
0.8214
```

```
>> Structura.R(2,2)

ans =

    0.8913
```

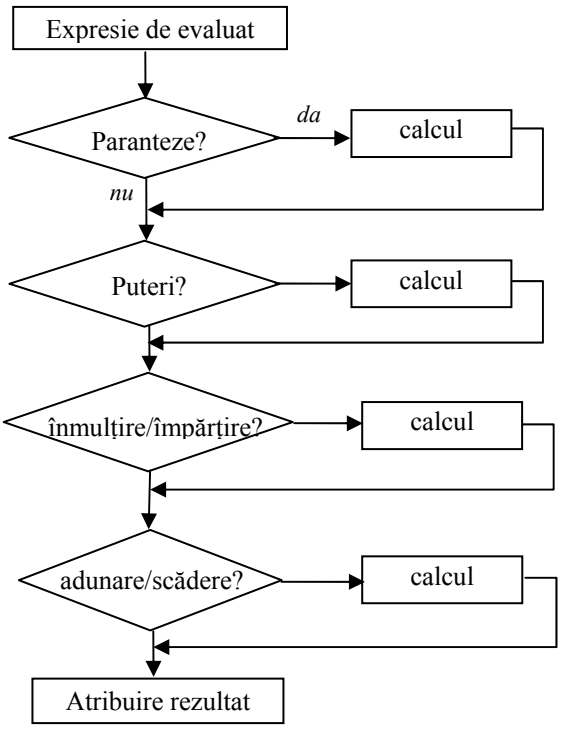
sau

```
>> Structura.nume

ans =

  Ionescu George
```

4.3. Operații şi operatori

Expresiile aritmetice conțin operanzi şi **operatori aritmetici**, iar în urma efectuării operației rezultatul se atribuie unei variabile. Limitele rezultatelor numerice sunt în concordanță cu valorile admisibile predefinite (`realmax, realmin, inf, NaN`).

Ordinea operațiilor este aceeaşi ca cea din matematica elementară. Mecanismul de evaluare a unei expresii matematice poate fi reprezentată prin secvențe de calcul cu următoarea structură logică:



Având în vedere modul de operare orientat pe structuri de date de tip tablou (matrice), în Matlab calculele se pot efectua în două maniere:

1. Calculul matriceal pe baza regulilor de *operare cu matrice*;
2. Calculul cu *scalari*, care se extinde pentru *operarea cu tablouri*.

Modul de calcul se distinge prin sintaxa operatorilor, aşa cum se arată în tabelul 4.2.

Tabelul 4.2. Operaţii şi operatori aritmetici în Matlab

Operaţia	Operanzi Scalari	Operanzi matriceali	Operanzi tablou
Adunare Scădere	+ -	+ -	+ -
Înmulţire	*	*	.*
Împărţire la stânga	/	/	./
Împărţire la dreapta	\	\	.\
Ridicare la putere	^	^	.^
Transpunerea	nu are sens	‘	.‘
Specifică ordinea de evaluare	( )		

Calculul cu scalari se efectueaza cu ajutorul operatorilor: + , - , * , / împărţirea la dreapta cu semnificaţia: a/b ⇔ a:b, \ împărţirea la stânga cu semnificaţia: a\b ⇔ b:a, ^ ridicarea la putere a^b.

Operaţiile aritmetice cu tablouri sunt permise în Matlab şi se bazează pe calculul cu scalari la nivelul tablourilor de date, în modul *element cu element* (între elementele corespondente, situate în aceeaşi poziţie a tablourilor). De pildă, operaţia de înmulţire a matricelor se efectuează după regula cunoscută respectând regula ca numărul de coloane al primei matrice (înmulţitor) să fie egal cu numărul de linii al matricei deînmulţit. Astfel, produsul matricelor nu este acelaşi lucru cu produsul tablourilor respective unde operaţia de înmulţire se aplică între fiecare două elemente omoloage.

Mai multe exemple pentru evidenţierea comparativă a modului de lucru cu operatorii în Matlab sunt date în subcapitolul *5.2. Calcul matriceal*.

Alte categorii de operatori care intervin în expresiile Matlab, cum sunt *operatorii logici* şi *operatorii relaţionali*.

Operatorii logici se folosesc la combinarea logică a două sau mai multe expresii ce definesc fie propoziţii de sine stătătoare, fie anumite condiţii logice necesare în multe probleme concrete. Operatorii logici sunt:

Operator	Simbol	Prioritatea
NU	~	1
SI	&	2
SAU	|	3
SAU EXCLUSIV	XOR(S,T)	3

Operatorii &, | precum şi XOR operează între doi scalari sau cu două matrice de dimensiuni egale, element cu element.

Operatorii relaţionali intervin în construcţia condiţiilor logice. În Matlab există şase operatori relaţionali utilizaţi în mod general pentru a compara două matrice de dimensiuni egale (sau două expresii matriceale):

Operator	Simbol
Mai mic	<
Mai mic sau egal	<=
Mai mare	>
Mai mare sau egal	>=
Identic	==
Diferit	~=

Modul de lucru efectiv al operatorilor relaţionali: compară element cu element două matrice, cu condiţia ca acestea să aibă aceeaşi dimensiune. Aplicarea acestor operatori este detaliată în secţiunea *4.7. Instrucţiuni Matlab*.

În Anexa 1 sunt prezentaţi în sinteză operatorii şi caracterele speciale recunoscute de mediul Matlab.

4.4. Funcții pentru aproximarea rezultatelor calculelor

Aproximarea numerelor zecimale se poate face prin lipsă sau prin adaos astfel:

- cu întregi (prin trunchiere sau prin rotunjire);
- cu numere raționale;
- cu fracții continue.

a) Aproximarea cu întregi se face utilizând funcțiile `fix`, `floor`, `ceil` sau `round`, cu următoarea sintaxă generală:

`nume_functie(`*`argument`*`)`

unde *`argument`* este numărul (zecimal) care trebuie aproximat prin rotunjire astfel:

- `round` - la cel mai apropiat întreg,
- `fix` - la cel mai apropiat întreg spre zero,
- `ceil` - la cel mai apropiat întreg spre $+\infty$,
- `floor` - la cel mai apropiat întreg spre $-\infty$.

Observație. Pentru numere complexe funcțiile de mai sus operează independent asupra celor două componente numerice (partea reală şi partea imaginară).

Exemple.

Aplicație

```
%aproximari numere reale
v=[0 2 2.3 4.7 -5.2]
a=round(v)
b=fix(v)
c=ceil(v)
d=floor(v)
```

Rezultate

```
v =
     0    2.000    2.300    4.700   -5.200

a =
     0     2     2     5    -5

b =
     0     2     2     4    -5

c =
     0     2     3     5    -5

d =
     0     2     2     4    -6
```

```
%aproxim numere complexe
A=[1.25+2.59i 7.3-5.3i;
-4.2+1.8i -2.6-1.4i]
a=ceil(A)
b=fix(A)
c=floor(A)
d=round(A)
```

```
A =
   1.2500 + 2.5900i   7.3000 - 5.3000i
  -4.2000 + 1.8000i  -2.6000 - 1.4000i

a =
   2.0000 + 3.0000i   8.0000 - 5.0000i
  -4.0000 + 2.0000i  -2.0000 - 1.0000i

b =
   1.0000 + 2.0000i   7.0000 - 5.0000i
  -4.0000 + 1.0000i  -2.0000 - 1.0000i

c =
   1.0000 + 2.0000i   7.0000 - 6.0000i
  -5.0000 + 1.0000i  -3.0000 - 2.0000i

d =
   1.0000 + 3.0000i   7.0000 - 5.0000i
  -4.0000 + 2.0000i  -3.0000 - 1.0000i
```

b) Aproximarea cu numere raţionale

Termenul raţional vine de la *raţie* sau *raport*, adică o exprimare sub formă de fracţii a/b. Funcţia `rats` se apelează cu sintaxele:

```
y=rats(x)
y=rats(x, len)
```

şi returnează rezultatul aproximării cu rapoarte numerice (fracţii) a variabilei argument interpretată simbolic sub formă de şir de caractere cu lungime controlată prin variabila `len`, care alocă spaţiul de scriere pentru variabila de ieşire. Rezultatul funcţiei va consta deci în rapoarte de numere diferite (ca ordin de mărime şi implicit ca lungime a şirului de caractere), în funcţie de valoarea argumentului `len` care implicit este 13. În cazul în care spaţiul de afişare alocat de variabila `len` nu poate cuprinde şirul rezultat se afişează simbolul ∗.

Exemplu: aproximarea cu numere raţionale a numărului π pentru diferite valori ale argumentului `len`.

```
>> pi

ans =
    3.1416

>> y=rats(pi)

y =
    355/113
```

```
>> pi

ans =
    3.1416

>> =rats(pi,4)

y =
     *
```

```
>> pi

ans =
    3.1416

>> y=rats(pi,17)

y =
   104348/33215
```

c) Aproximarea cu fracţii continue

Acest procedeu se referă la exprimarea numerelor ca *expresii de fracţii ireductibile*, sub forma:

$$a = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_k}}}$$

De pildă, astfel de descompuneri cu diferite grade de aproximare prin fracţii pot fi:

$$0.25=1/4, \quad 1.25=1+1/4\ , \quad 1.343 = 1 + \cfrac{1}{3 + \cfrac{1}{-12 + \cfrac{1}{6}}}\ , \text{ etc.}$$

În Matlab se utilizează funcţia `rat` cu sintaxele:

```
y=rat(x)
Y=rat(x,tol)
[a,b]=rat(x)
[a,b]=rat(x,tol)
```

unde:

x - este numărul sau vectorul ale cărui elemente trebuie aproximate cu fracţii continue;

tol - reprezintă toleranţa acceptată între numărul x şi aproximarea (descompunerea) sa $y - x \le tol$; unde implicit $tol = 10^{-6}$;

y – este variabila şir de caractere ce conţine exprimarea lui x ca fracţie continuă;

a şi b - numărătorul şi numitorul fracţiei care aproximează pe x cu toleranţa dată (`tol`).

Exemplu: aproximarea cu fracţii continue a numărului π cu două toleranţe diferite: cea implicită şi respectiv 10^{-12}.

```
>> pi
ans =
    3.1416

>> y=rat(pi)
y =
 3 + 1/(7 + 1/(16))
```

```
>> pi
ans =
    3.1416

>> y=rat(pi,1e-12)
y =
3 + 1/(7 + 1/(16 + 1/(-294 + 1/(3 + 1/(-4
+ 1/(5))))))
```

Observaţie. Cu sintaxele:

```
[a,b]=rat(x)
```

respectiv

```
[a,b]=rat(x,tol)
```

se obţine efectul funcţiei `rats` prin returnarea în mod explicit a numărătorului, respectiv a numitorului fracţiei care aproximează numărul argument, (eventual specificând toleranţa de aproximare).

De exemplu:

```
>> [a,b]=rat(pi,1e-12)

a =
      5419351
b =
      1725033
```

4.5. Funcţii pentru lucrul cu şiruri de caractere

Constanta *şir* se reprezintă ca o succesiune de octeţi ce conţine codurile ASCII (American Standard Code for Information Interchange) ale caracterelor componente din şirul respectiv. Codurile ASCII sunt grupate astfel: de la 0 la 31 desemnează caractere negrafice (comenzi, săgeţi, etc.), iar de la 32 la 127 desemnează caractere grafice (litere, cifre, semne de punctuaţie, simboluri). În continuare, până la 256 se găseşte codul ASCII extins (alte caractere).

Operaţiile cu şiruri sunt: conversii, evaluări, comparaţii, declaraţii, etc. Regula sintactică de bază în Matlab cu privire la declaraţia şirurilor de caractere este utilizarea ghilimelelor simple.

O suită de funcţii pentru lucrul cu variabile şir de caractere este prezentată în tabelul 4.3.

Tab. 4.3 Operaţii cu variabile şir

Funcţia	Descriere	Exemplu de aplicare

‘ ‘	Declaraţia de şir de caractere (variabilă *text*).	`>> text='Matlab 6.5'`
		`text =`
		`Matlab 6.5`

abs	Converteşte şirurile de caractere în valorile numerice(0-127). Trecere *din text în numeric.*	`>> a=abs('internet')`
		`a =`
		`105 110 116 101 114 110 101 116`

setstr	Converteşte valorile numerice ale codului ASCII în caractere. Trecere *din numeric în text*.	`>> S=setstr(32:127)`
		`S =`
		`!"#$%&'()*+,./0123456789:;<=>?@ABCDEFG HIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklm nopqrstuvwxyz{|}~`

isstr	Detectează (testează) dacă o variabilă este şir de caractere sau cod numeric. Returnează: 1 dacă variabila este text şi 0 dacă aceasta este cod numeric ASCII	`>> x=isstr(text)`
		`x =`
		`1`
		`>> x=isstr(a)`
		`x =`
		`0`

Tab. 4.3 *(Continuare)*

str2mat
 	
char
	Formează matrice text din şiruri de caractere individuale (luate ca rânduri), cu număr de linii dat de numărul de şiruri individuale şi număr de coloane dat de numărul de caractere al celui mai lung şir individual.	`>> Mat_sir=str2mat('Acesta','este','un program','"Matlab"')`
		
		`Mat_sir =`
		`Acesta`
		`este`
		`un program`
		`"Matlab"`

blanks	Crează spaţii între şirurile de caractere. Se poate utiliza şi cu funcţia `disp`, deplasând cursorul spre dreapta cu n poziţii.	`>> Mat_sir=str2mat('Acesta',blanks(2), 'este','un program',blanks(5),'"Matlab"')`
		
		`Mat_sir =`
		`Acesta`
		
		`este`
		`un program`
		
		`"Matlab"`

deblank	Elimină spaţiile libere de la sfârşitul şirurilor de caractere	`Deblank(s)`

eval	Evaluează şirurile de caractere conţinând expresii matematice recunoscute în Matlab.	`>> X=10`
		
		`X =`
		`10`
		
		`>> eval('X^2+2')`
		
		`ans =`
		`102`

feval	Evaluează funcţii Matlab specificate prin nume	`>> y=feval('sin',pi/6)`
		
		`y =`
		`0.5000`

strcmp	Compară şiruri de caractere şi returnează 1 – dacă acestea sunt identice;
	0 – dacă nu sunt identice.
		`>> a=strcmp('Test','Test')`
		`a =`
		`1`
		
		`>> a=strcmp('test','Test')`
		`a =`
		`0`

findstr	Caută un şir de caractere într-un alt şir de caractere. Funcţia returnează un vector ce conţine poziţiile de la care începe şirul S2 în şirul S1.	`>> sir1='cAuta cArActerul A in Acest sir'`
		
		`sir1 =`
		`cAuta cArActerul A in Acest sir`
		
		`>> M=findstr(sir1, 'A')`
		`M =`
		`2 8 10 18 23`

Tab. 4.3 *(Continuare)*

upper	Converteşte literele unui şir de caractere în litere mari	`>> SIR=upper(sir1)`
		`SIR =`
		`CAUTA CARACTERUL A IN ACEST SIR`

lower	Converteşte literele unui şir de caractere în litere mici	`>> sir=lower(SIR)`
		`sir =`
		`cauta caracterul a in acest sir`

isletter	Verifică dacă elementele unui vector sunt sau nu literele alfabetului. Returnează matricea A cu elemente 1 în poziţiile în care elementele şirului sunt litere şi 0 în rest	`>> sir_mixt='Data: 03.02.2006'`
		`sir_mixt =`
		`Data: 03.02.2006`
		`>> L=isletter(sir_mixt)`
		`L =`
		`1 1 1 1 0 0 0 0 0 0`
		`0 0 0 0 0 0`

num2str	Conversie număr-şir de caractere (text).	`>> cifra=num2str(pi)`
		`cifra =`
		`3.1416`

int2str	Conversie număr întreg – şir de caractere (text). Exclude punctul zecimal.	`>> intreg=int2str(pi)`
		`intreg =`
		`3`

str2num	Converteşte un şir de caractere într-un număr	`>> numar=str2num('5.12E-3')`
		`numar =`
		`0.0051`

dec2hex	Converteşte un întreg zecimal într-un şir hexazecimal	`>>X=dec2hex(15)`
		`X =`
		`F`

hex2dec	Converteşte un şir hexazecimal într-un întreg zecimal	`>> A=hex2dec('3F8')`
		`A =`
		`1016`

sprintf	Converteşte un număr într-un şir sub controlul formatului *(vezi caracterele de control)*	`>> Y=sprintf('%6.5e\t %6.5e',X)`
		`Y =`
		`1.12345e+000 5.12300e+000`