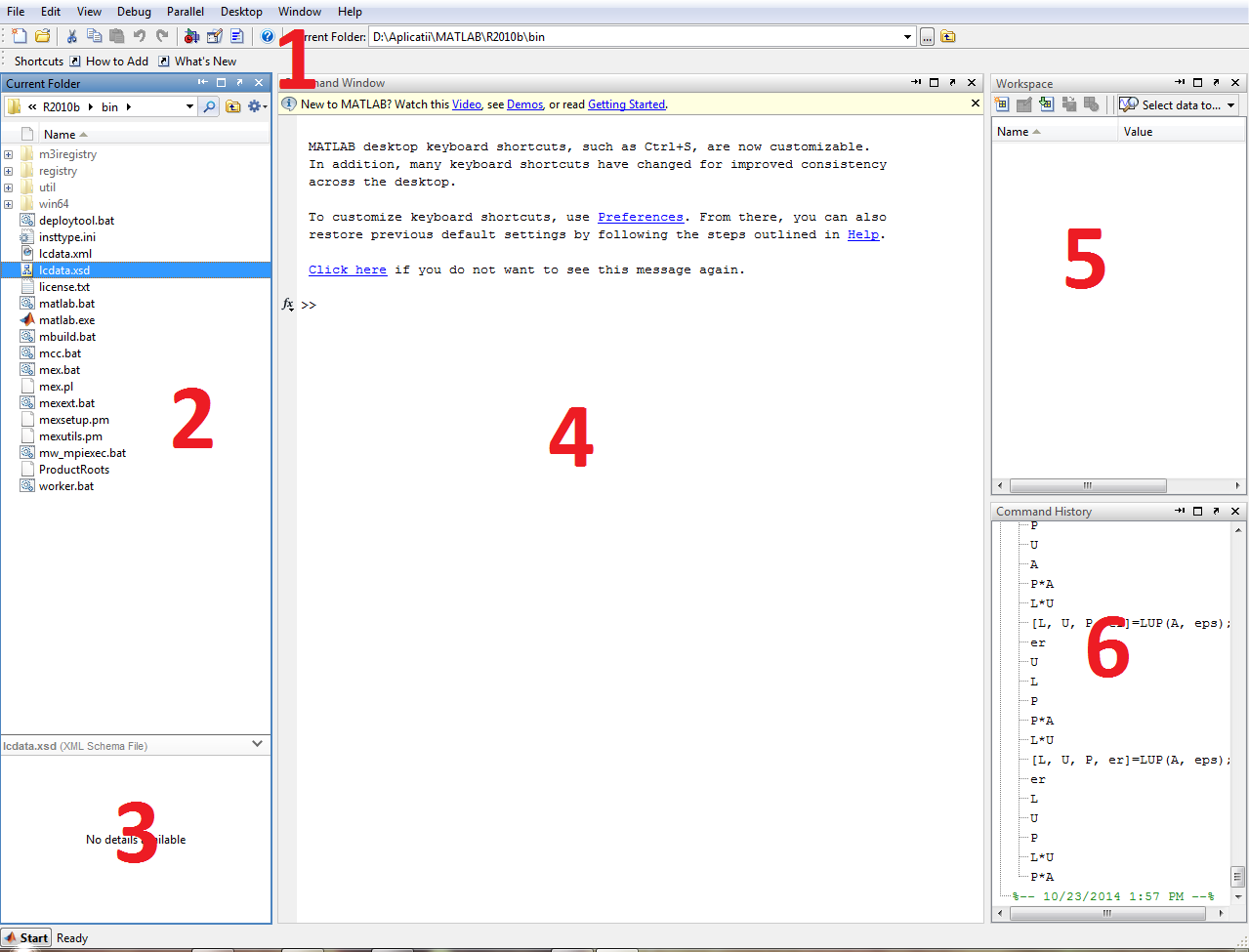
**TC 1. Scurtă prezentare MatLab**

Instrumentul de lucru folosit în cadrul cursului *Programare evolutivă şi algoritmi genetici* este MatLab (MATrix LABoratory). MatLab este un mediu complex pentru calcule ştiinţifice, cu numeroase module şi funcţii care implementează calcule uzuale sau complexe des întîlnite în matematică, fizică, inginerie etc. În MatLab pot fi utilizate comenzi care produc rezultate imediate, pot fi scrise programe simple, pentru utilizare rapidă, de un număr relativ mic de ori (chiar o singură dată) sau pot fi realizate aplicații complexe, cu interfață grafică, destinate utilizării extensive pentru cercetare sau rezolvarea de probleme concrete.

Pentru referințe detaliate se poate studia sistemul de ajutor (*help*) din MatLab, care este foarte detaliat. De asemenea, MatLab pune la dispoziție ghiduri video și text, accesibile prin sistemul de ajutor sau prin ancorele disponibile în partea superioară a ferestrei de comandă (*Video*, *Demos*, *Getting Started*). Acestea sînt destinate inițierii utilizatorului în lucrul cu MatLab și familiarizării acestuia cu elementele de bază.

Pentru o mai bună înțelegere, recomandăm parcurgerea acestui capitol avînd la îndemînă un calculator cu MatLab instalat, cititorul avînd astfel posibilitatea de a accesa, testa și verifica elementele întîlnite, în plus avînd ocazia să creeze și să testeze pe loc exemple noi pentru a se asigura că a înțeles corect.

Interfața tipică MatLab, așa cum apare imediat după pornirea mediului de lucru, este vizibilă în figura de mai jos. O parte din instrumentele de bază ale mediului sunt vizibile în această interfață (meniul și bara de instrumente, exploratorul de fișiere, caseta de detalii, fereastra de comenzi, spațiul de lucru, istoricul comenzilor), în timp ce alte instrumente pot fi deschise la cerere (de exemplu editorul de texte integrat), sistemul de ajutor și documentație, exploratorul de funcții etc.



Conform numerotării din figură, zonele interfeței au următoarele roluri:

1. *Meniul și bara de instrumente* – asemănător tuturor aplicațiilor obișnuite pentru Windows. Există diferențe în componența acestor elemente, în funcție de varianta de MatLab folosită.
2. *Current Folder* (directorul curent) este un explorator de fișiere asemănător celui din sistemul de operare. În această zonă sînt vizibile fișierele din directorul curent. Instrumentul poate fi folosit și pentru explorarea discului și schimbarea directorului curent. În același scop se poate folosi și cîmpul „*Current folder*” din bara de instrumente.

Atunci cînd trebuie să execute o comandă care referă o funcție, MatLab caută fișierul corespunzător în directorul curent și pe o cale de căutare predefinită (asemănător căii de căutare a sistemului de operare) – funcțiile MatLab sînt stocate în fișiere text cu extensia *.m*. Utilizatorul poate să adauge diverse directoare la calea de căutare sau să le excludă. Pentru aceasta trebuie să navigheze (în exploratorul de fișiere) pînă la directorul dorit apoi să execute click-dreapta pe numele directorul și să aleagă din meniul apărut opțiunea „*Add to path*” (adaugă la cale) sau „*Remove from path*” (elimină din cale). Fiecare din cele două opțiuni are două subopțiuni care permit executarea acțiunii strict asupra directorului selectat (*Selected folders*) sau asupra directorului selectat și a tuturor subdirec-toarelor pentru care el este rădăcină (*Selected folders and subfolders*).

1. *Details* (detalii) – în această zonă sunt afișate detalii despre fișierul curent selectat în explorator, dacă sunt disponibile – se afișează detalii specifice MatLab. Dacă fișierul conține funcții MatLab se afișează *linia H1* urmată de numele funcției și lista parametrilor formali (linia H1 va fi prezentată în 1.3). Executînd dublu click pe numele funcției (sau pe numele fișierului în explorator) se deschide editorul de texte integrat pentru editarea funcției respective.
2. *Command window* – fereastra de comenzi ocupă partea cea mai mare a interfeței. Aici se scriu comenzi și se afișează rezultatele execuției lor. O parte a rezultatelor pot fi afișate în ferestre separate, deschise de comanda executată. Asupra tipurilor de comenzi se va reveni în 1.2. În partea superioară a ferestrei de comenzi se află ancorele pentru ghidurile video și text destinate începătorilor (*Video*, *Demos*, *Getting Started*).
3. *Workspace* – spațiul de lucru permite vizualizarea tuturor variabilelor curent definite și a valorilor asociate lor (în formă succintă). Din această zonă pot fi accesate diverse instrumente pentru crearea de noi variabile sau eliminarea unor variabile existente, salvarea unor variabile în fișiere sau încărcarea din fișiere, modificarea variabilelor, vizualizarea grafică a acestora. Prin dublu click pe numele unei variabile se deschide un alt instrument, editorul de variabile (*Variable editor*) care permite vizualizarea în formă de tabel bidimensional a variabile respective și modificarea ei – variabilele utilizate de MatLab sunt considerate implicit matrice.
4. *Command history* este o casetă care prezintă lista comenzilor executate pînă la momentul curent, în ordine cronologică. Lista este păstrată și continuată de la o sesiune de lucru la alta. Această listă permite repetarea rapidă a unor comenzi anterioare (dublu click pe comanda care trebuie executată) și căutarea incrementală a unei comenzi anterioare (atunci cînd această casetă este selectată se apasă tastele corespunzătoare primelor caractere din comanda căutată; după fiecare apăsare este selectată automat prima comandă care corespunde secvenței de căutare introduse, dacă există o astfel de comandă)

MatLab este orientat pe evaluarea expresiile, ca urmare comenzile care se introduc în linia de comandă sînt fie comenzi propriu-zise ale mediului, fie expresii care trebuie evaluate. Expresiile sînt construcții sintactice formate din operanzi și operatori.

Asemănător ferestrei de comandă din sistemul de operare, fereastra de comandă MatLab prezintă un prompter (>>) pe linia curentă. Comenzile pot fi scrise și editate pe linia curentă, interpretarea lor avînd loc imediat după apăsarea tastei Enter, care semnifică terminarea scrierii comenzii (în unele situații, comenzile pot fi scrise pe mai multe linii succesive; dacă la apăsarea tastei Enter MatLab consideră comanda neterminată, se trece pe linia următoare și se așteaptă continuarea scrierii comenzii).

La sfîrșitul unei comenzi se poate introduce caracterul ; (punct și virgulă), care are rolul de a suprima afișarea rezultatului evaluării pe ecran. În lipsa acestuia, rezultatul evaluării este imediat afișat.

După terminarea execuției unei comenzi apare din nou prompterul și se așteaptă introducerea unei noi comenzi.

**Exemple**

>> nmax=10000

nmax =

10000

>> eps=0.000001

eps =

1.0000e-006

>> a=2+3

a =

5

>> b=a+1;

>> b

b =

6

>> b\*3

ans =

18

>> ans+4

ans =

22

>> A=[[-8 1 0]

[3 7 1]

[1 4 6]]

A =

-8 1 0

3 7 1

1 4 6

MatLab este un mediu *case-sensitive*, literele mari fiind considerate diferite de literele mici în identificatorii variabilelor, funcțiilor și comenzilor. În exemplul anterior a și A sînt variabile diferite.

**Literali**

MatLab utilizează literali numerici și de tip șir de caractere, ca orice limbaj de programare generală, dar și cîțiva literali specifici.

Literalii de tip șir de caractere sînt recunoscuți prin încadrarea unui șir de caractere între apostrofuri.

Literalii numerici se scriu folosind notația zecimală convențională, cu punct zecimal. Se poate folosi semnul plus/minus și forma științifică (adăugînd litera e și o putere a lui zece).

MatLab lucrează și cu numere complexe. Pentru partea imaginară se folosește sufixul i sau j (ambele au aceeași semnificație).

**Exemple**

>> a='Acesta este un sir de caractere'

a =

Acesta este un sir de caractere

>> b=-2.5e-2

b =

-0.0250

>> e=5+4i

e =

5.0000 + 4.0000i

Valorile numerice sunt memorate intern implicit în formatul *long* definit de standardul IEEE pentru reprezentarea în virgulă mobilă, cu o precizie de 16 zecimale. Dacă reprezentarea necesită mai multe de 52 biți, valorile vor fi trunchiate, ducînd la rezultate eronate.

**Exemple**

>> d=12345678901234567

d =

1.2346e+016

>> e=12345678901234568

e =

1.2346e+016

>> d==e

ans =

1

În acest exemplu cele două valori, evident diferite, sînt considerate egale datorită limitărilor de reprezentare.

Pentru valorile întregi se poate folosi reprezentarea pe 8, 16, 32 sau 64 biți, folosind uint8, uint16, uint32 sau uint64, ca în exemplul următor.

Exemple

>> d=uint64(12345678901234567)

d =

12345678901234567

>> e=uint64(12345678901234568)

e =

12345678901234568

>> d==e

ans =

0

**Variabile**

În MatLab variabilele sînt create automat atunci cînd sînt întîlnite (de obicei într-o expresie de atribuire), tipurile fiind alese automat, în funcție de valoarea atribuită. Dacă unei variabile care există deja i se atribuie o valoare de alt tip, atunci ea este ștearsă și se alocă altă variabilă, cu același nume, dar cu tipul corespunzător noii valori.

Identificatorii trebuie să respecte următoarele reguli pentru a fi acceptați de MatLab:

* + în componența lor pot intra literele a-z, A-Z, cifrele 0-9 și caracterul \_ (underscore);
  + un identificator trebuie să înceapă cu o literă;
  + literele mari și literele mici sînt considerate diferite (case sensitive).

Identificatorii variabilelor pot avea orice lungime, dar numai primele 63 caractere sînt memorate (doi identificatori cu primele 63 caractere identice vor fi considerați identici, chiar dacă restul caracterelor sînt diferite). Pentru a verifica această valoare se poate executa comanda namelengthmax.

Comanda who poate fi folosită pentru a vizualiza variabilele curent definite. Ele compun mediul de lucru (*workspace*) și sînt vizibile și în fereastra *Workspace*. Comanda whos prezintă variabilele curente împreună cu dimensiunile (conceptuale și spațiu de memorie ocupat) și tipurile lor.

**Exemple**

>> who

Your variables are:

A a b map nri x

X ans eps nmax omega x0

>> whos

Name Size Bytes Class Attributes

A 3x3 72 double

X 257x250 514000 double

a 1x1 8 double

ans 1x1 8 double

b 1x1 8 double

eps 1x1 8 double

map 64x3 1536 double

nmax 1x1 8 double

nri 1x1 8 double

omega 1x1 8 double

x 3x1 24 double

x0 3x1 24 double

O variabilă rămîne definită și utilizabilă pînă cînd se întîlnește una din următoarele situații:

* + se încheie sesiunea curentă de lucru (se închide MatLab);
  + se încheie funcția în care a fost definită;
  + este ștearsă în mod explicit prin utilizarea comenzii clear. Comanda poate fi utilizată pentru a șterge toate variabilele (resetează mediul curent de lucru) sau o anumită variabilă (dacă e urmată de identificatorul acelei variabile).

**Variabila implicită *ans***

În urma evaluării unei expresii se poate obține un rezultat sau mai multe (în cazul apelării unor funcții se pot obține mai multe rezultate). Pentru a păstra rezultatele în vederea utilizării ulterioare, ele sînt asociate unor variabile, prin expresii de atribuire (o expresie de atribuire poate asocia valori uneia sau mai multor variabile). Dacă expresia de evaluat nu este de tip atribuire, deci rezultatul nu este asociat unei variabile explicite, atunci rezultatul este asociat unei variabile implicite, numite ans, care este creată automat. Variabila ans are tipul corespunzător rezultatului care i se asociază.

Ea poate fi utilizată în continuare ca orice altă variabilă, pînă cînd o altă comandă asociază o altă valoare variabile ans. Dacă în urma evaluării se obțin mai multe rezultate, fără a fi asociate explicit unor variabile, atunci doar primul este asociat variabile ans, restul fiind pierdute.

**Operatori uzuali**

MatLab folosește operatorii algebrici uzuali și regulile de precedență obișnuite: adunare +, scădere -, înmulțire \*, împărțire /, schimbarea ordinii de evaluare (). În plus, sunt implementați următorii operatori: ridicare la putere ^, împărțire la dreapta \ (cu semnificație diferită pentru scalari și masive), transpunere conjugată complexă ‘ (apostrof).

În cazul valorilor scalare a\b are aceeași semnificație ca b/a. În cazul masivelor, dacă A este o matrice pătrată, atunci A\B are semnificația A-1\*B (evident, calculul se poate face dacă A este inversabilă și dimensiunile celor două masive îndeplinesc condițiile matematice).

Pentru masive, operațiile aditive se efectuează element cu element, asemănător operațiilor cu valori scalare, însă operațiile multiplicative au altă semnificație. Pentru a forța efectuarea operațiilor element cu element, se utilizează caracterul punct (.) în fața operatorului (oricare din operatorii enumerați mai sus). De exemplu, dacă A și B sunt matrice, A\*B este o operație de înmulțire de matrice (supusă restricțiilor cunoscute), în timp ce A.\*B este o operație prin care se înmulțesc elementele de pe aceeași poziție din cele două matrice, rezultatul fiind o matrice nouă (A și B trebuie să aibă aceleași dimensiuni).

Operatorul de transpunere conjugată complexă realizează transpunerea masivului asupra căruia este aplicat și, în același timp, conjugarea fiecărui element. Dacă masivul are elemente reale, atunci se realizează o transpunere obișnuită.

De asemenea, sînt disponibili următorii operatori relaționali: mai mare >, mai mic <, mai mare sau egal >=, mai mic sau egal < =, egal ==, diferit ~=.

În cazul scalarilor, rezultatul aplicării unui operator relațional este 1 (adevărat) sau 0 (fals).

În cazul masivelor operatorul de verificare a egalității nu verifică dacă cele două masive sunt egale ci unde au elemente egale. Rezultatul este un masiv cu elemente 1 sau 0 (1 în pozițiile unde cele două masive au elemente egale, 0 în rest). Cele două masive trebuie să aibă aceleași dimensiuni.

**Exemple**

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

A =

1 2 3

4 5 6

7 8 9

>> B=[1 9 3; 9 5 9; 7 9 9]

B =

1 9 3

9 5 9

7 9 9

>> A==B

ans =

1 0 1

0 1 0

1 0 1

**Masive**

În exemplul anterior (comanda who), se observă că pentru toate variabilele, inclusiv cele scalare, coloana *Size* prezintă două valori: în MatLab toate variabilele sînt considerate matrice. Vectorii sînt matrice în care una din dimensiuni este 1, ca urmare există vectori linie și vectori coloană, tratați ca atare în operațiile cu masive. Adesea este necesară transpunerea vectorilor (din linie în coloană sau invers) pentru a putea efectua diverse operații (de exemplu înmulțiri cu masive). Valorile simple (scalare) sînt considerate matrice cu o linie și o coloană.

Următoarele elemente sînt des folosite în lucrul cu masive.

**Operatorul de concatenare**

Pentru inițializarea explicită a unui masiv se poate folosi operatorul de concatenare []. Prin concatenare se alătură mai multe valori pentru a forma un masiv. Concatenarea se poate face pe orizontală sau pe verticală. Elementele enumerate între parantezele drepte pot fi separate prin spațiu sau virgulă, dacă se dorește concatenarea pe orizontală. Se poate folosi caracterul ; ca separator acolo unde se dorește concatenarea pe verticală (rezultatul va fi o matrice) – e obligatoriu ca numărul elementelor de pe fiecare linie să fie același.

La inițializarea matricelor se pot folosi și operatori de concatenare imbricați: se concatenează vectori de aceeași lungime formați prin concatenarea orizontală a unor valori, ca în exemplul următor (matricea A). În cazul inițializării matricei A se observă o comandă scrisă pe mai multe rînduri.

Comanda disp(nume) afișează valoarea variabile specificate (*nume*); același efect se obține în fereastra de comandă și prin scrierea numelui variabilei. În loc de numele unei variabile, se poate folosi un literal care se va afișa ca atare.

**Exemple**

>> x=[1 2 3];

>> disp(x)

1 2 3

>> a=x'

a =

1

2

3

>> b=[1;2;3]

b =

1

2

3

>> c=b'

c =

1 2 3

>> d=[1 2;3]

??? Error using ==> vertcat

CAT arguments dimensions are not consistent.

>> A=[[-8 1 0]

[3 7 1]

[1 4 6]]

A =

-8 1 0

3 7 1

1 4 6

>> B=[-8 1 0; 3 7 1; 1 4 6]

B =

-8 1 0

3 7 1

1 4 6

**Operatorul** : **(două puncte)**

Operatorul : poate fi folosit pentru a genera un șir de valori cu distanțe egale între ele, de la o valoare inițială pînă la (cel mult) o valoare finală.

Operatorul poate fi folosit în două forme:

* + cu precizarea valorii inițiale și a valorii finale (pasul fiind implicit 1)

>> x=1:7

x =

1 2 3 4 5 6 7

* + cu precizarea valorii inițiale, a pasului și a valorii finale

>> y=1:0.5:3

y =

1.0000 1.5000 2.0000 2.5000 3.0000

Valorile din șir se obțin adăugînd pasul la valoarea anterioară, prima valoare fiind cea precizată în expresie. Șirul se încheie cînd se obține o valoare strict mai mare decît valoarea finală (aceasta nu face parte din șir).

>> z=1:0.3:2

z =

1.0000 1.3000 1.6000 1.9000

Se poate folosi și pas negativ, în acest caz valoarea finală fiind mai mică decît valoarea inițială.

>> t=0:-0.3:-1

t =

0 -0.3000 -0.6000 -0.9000

În următoarele situații se obține un șir vid (matrice goală cu o linie și zero coloane):

* + valoarea finală este mai mică decît cea inițială și pasul este pozitiv;
  + valoarea finală este mai mare decît cea inițială și pasul este negativ;
  + pasul este zero.

>> 0:-1:10

ans =

Empty matrix: 1-by-0

**Vectorul vid**

Se poate crea un vector vid printr-o inițializare de forma x=[]. Prin înlocuirea unor elemente ale unui vector cu un vector vid se obține efectul invers concatenării, ștergerea de elemente.

**Referirea elementelor unui masiv**

Pentru referirea elementelor unui masiv se folosesc indicii elementelor respective. Construcția sintactică e formată din numele masivului urmat de o pereche de paranteze rotunde între care se trec indicii elementului dorit. Dacă masivul are mai multe dimensiuni, indicii se separă prin virgulă.

Spre deosebire de limbajele generale de programare, unde se poate referi un singur element al masivului, în MatLab se pot referi mai multe elemente, adiacente sau nu. Un indice poate fi o valoare simplă (scalar), un domeniu de valori (se referă elementele al căror indice se află în domeniul precizat) sau o concatenare de valori (se referă elementele al căror indice este precizat).

**Exemple**

>> x=1:2:14

x =

1 3 5 7 9 11 13

>> a=x(3)

a =

5

>> y=x(3:6)

y =

5 7 9 11

>> z=x([2 4 5])

z =

3 7 9

>> z=x([2 4 7])

z =

3 7 13

>> x([2 4])=[]

x =

1 5 9 11 13

În ultimul exemplu au fost șterse elementele cu indicii 2 și 4, prin înlocuirea fiecăruia din ele cu vectorul vid.

**Controlul execuției**

Comenzile MatLab pot fi structurate, folosind structuri de control standard: alternative, repetitive.

*Structura alternativă* este implementată prin instrucțiunea *if-else* (sau *if-elseif*), cu forma generală

if condiție

bloc\_1

else

bloc\_2

end

Dacă rezultatul evaluării condiției este adevărat, se execută bloc\_1, altfel se execută bloc\_2. Cele două blocuri sunt formate din una sau mai multe comenzi. Cuvîntul cheie end marchează încheierea structurii alternative și este pus în corespondență cu cel mai recent if neîncheiat. Se pot folosi structuri alternative imbricate. O variantă este utilizarea clauzei elseif în loc de else. Se pot utiliza clauze elseif multiple.

if condiție\_1

bloc\_1

elseif

condiție\_2

else

bloc\_2

end

În cazul structurilor alternative imbricate, utilizarea clauzei elseif elimină necesitatea încheierii fiecărei instrucțiuni if prin cuvîntul cheie end, ceea ce duce la o mai bună lizibilitate.

**Exemplu**

>> a=1.5;

>> if a<1

disp('a<1');

else

if a<2

disp('1<=a<2');

else

disp('a>=2');

end;

end;

1<=a<2

>> if a<1

disp('a<1');

elseif a<2

disp('1<=a<2');

else

disp('a>=2');

end;

1<=a<2

*Structura alternativă multiplă* este implementată prin instrucțiunea switch, cu următoarea formă generală:

switch expresie

case valoare1

bloc1

case {valoare2, valoare3, valoare4, ...}

bloc2

…

otherwise

blocn

end

Instrucțiunea se execută astfel: se evaluează expresia expresie, și apoi se caută succesiv valoare obținută pe ramurile alternative marcate cu clauza case; la prima apariție a valorii, se execută blocul de pe ramura corespunzătoare apoi se încheie instrucțiunea. Dacă nici una dintre ramuri nu conține valoarea căutată, atunci se execută ramura implicită, marcată cu clauza otherwise, dacă această ramură există.

Rezultatul evaluării expresiei expresie trebuie să fie de tip întreg sau șir de caractere.

Pe ramurile alternative se pot afla valori propriu-zise (literali) sau expresii care includ operatori aritmetici sau logici (dar nu relaționali). Pe o ramură pot fi mai multe valori, pentru oricare dintre ele urmînd a se executa blocul corespunzător ramurii.

**Exemplu**

>> a=5;

>> switch a

case 1

disp('a=1');

case {2,3}

disp('a=2 sau 3');

case 3+2

disp('a=5');

otherwise

disp('altceva');

end;

a=5

*Structura repetitivă condiționată anterior* este implementată prin instrucțiunea while, cu următoarea formă generală:

while expresie

bloc

end

Blocul de comenzi se execută în mod repetat, atît timp cît rezultatul evaluării expresiei este adevărat (diferit de zero). Dacă rezultatul evaluării expresiei este un masiv, atunci toate elementele masivului trebuie să fie diferite de zero (adevărate) pentru ca rezultatul să fie considerat adevărat.

Blocul de comenzi trebuie să afecteze rezultatul evaluării expresiei pentru a evita o bucla infinită.

*Structura repetitivă cu număr cunoscut de pași* este implementată prin instrucțiunea for, cu următoarea formă generală:

for index=valori

bloc

end

Variabila index ia ca valoare, pe rînd, fiecare element din lista valori și pentru fiecare dintre acestea se execută blocul de comenzi. Lista de valori poate fi specificată ca interval cu pasul implicit 1, interval cu pas explicit sau listă explicită de valori.

**Exemple**

>> n=10;

>> for i=1:n

a(i)=i\*i;

end;

>> a

a =

1 4 9 16 25 36 49 64 81 100

>> for i=1:0.5:n

disp(i\*i);

end;

1

2.2500

4

6.2500

9

12.2500

16

20.2500

25

30.2500

36

42.2500

49

56.2500

64

72.2500

81

90.2500

100

>> for i=[1,3,5,7,9]

disp(i\*i)

end;

1

9

25

49

81

**Programare în MatLab**

Fereastra de comandă oferă o modalitate rapidă pentru efectuarea unor calcule interactive, dar uneori este necesară repetarea unor secvențe de calcule (imediat sau într-o altă sesiune de lucru), utilizarea unor parametri în calcule sau utilizarea unor secvențe mai lungi/complicate de calcule. Pentru astfel de situații se pot utiliza programe MatLab. Acestea se prezintă în două forme: *scripturi* și *funcții*. Programele MatLab se salvează în fișiere sursă cu extensia .m și sunt în esență colecții de comenzi. Un fișier sursă Matlab este un fișier text și poate fi creat cu ajutorul editorului integrat sau cu orice alt editor de texte.

În cadrul unui fișier sursă MatLab se află comenzi, instrucțiuni pentru controlul execuției și comentarii (care nu se execută).

Un comentariu poate să ocupe o linie întreagă, ultima parte dintr-o linie sau mai multe linii. Pentru comentariu pe o linie (sau ultima parte din linie) se folosește caracterul special % ca marcator de început de comentariu. Pentru comentarii care ocupă mai multe linii se poate marca fiecare linie cu caracterul % (comentariu pe o linie) sau se poate delimita întregul bloc de comentarii cu combinațiile de caractere %{ și %} (acestea trebuie să se afle pe linii separate).

**Exemple**

disp('test'); % acesta este un comentariu

% acesta este un comentariu pe o linie

%{

Acesta este un comentariu

pe mai multe linii

%}

Un ***script*** este o colecție de comenzi care se execută ca atare. Nu există parametri sau rezultate propriu-zise la nivel de script. Un script lucrează cu variabilele din spațiul de lucru curent și poate modifica acest spațiu de lucru (adăugare sau eliminare de variabile). Variabilele definite într-un script rămîn în spațiul de lucru după terminarea execuției scriptului. Aceste variabile pot fi asociate noțiunii de rezultat al scriptului.

Numele fișierului care conține un script (fără extensie) este considerat un nume de comandă. Pentru executarea scriptului se execută această comandă fără parametri, în fereastra de comandă sau în cadrul altui script sau al unei funcții.

O ***funcție*** este o colecție de comenzi care depinde de un set de parametri. Parametrii primiți și toate variabilele definite constituie un spațiu de lucru local, care există doar pînă la încheierea execuției funcției. Funcțiile întorc un rezultat, care poate fi compus, prin concatenare, din mai multe elemente.

O funcție are un antet, cu următoarea formă generală:

function rezultat = nume\_functie(lista\_parametri)

*Rezultat* este numele formal al rezultatului întors de funcție (poate fi o concatenare de mai multe rezultate individuale). *Lista\_parametri* este o înșiruire de parametri formali pe baza cărora va lucra funcția. Numele fișierului sursă care conține funcția trebuie să fie același cu numele precizat în antetul funcției. E posibil ca într-un fișier să fie scrise mai multe funcții, din care prima este funcția principală, care dă și numele fișierului, celelalte fiind funcții secundare.

După antetul funcției se pot scrie mai multe linii de comentarii care descriu scopul funcției, parametrii și rezultatul. Această secțiune descriptivă este considerată partea de ajutor (*help*) a funcției și este afișată de comanda help nume\_functie. Prima linie din această secțiune descriptivă este numită *linia H1* și este afișată în caseta de *detalii* atunci cînd în caseta *directorul curent* e selectat fișierul care conține funcția sau atunci cînd se utilizează comanda help asupra unui director. Secțiunea descriptivă este opțională și, dacă există, se încheie la prima linie goală sau prima linie executabilă. Dacă secțiunea descriptivă nu există, prima linie de tip comentariu din funcție e considerată linia H1.

Corpul unei funcții se încheie cu cuvîntul rezervat end.

Parametrii reali sunt puși în corespondență cu parametrii formali de la stînga la dreapta. La apel este permisă furnizarea unei liste de parametri reali mai scurtă decît lista formală (mai puțini parametri); în acest caz MatLab generează valori implicite pentru parametrii care lipsesc. Pentru astfel de situații, MatLab oferă o modalitate de verificare a numărului de parametri prin utilizarea variabilei nargin, care are ca valoare numărul de parametri reali furnizați; variabila nargout are ca valoare numărul de rezultate ale funcției.

Funcții și comenzi des utilizate

În exemplele din manual este necesară de multe ori generarea de numere aleatoare. MatLab oferă numeroase funcții pentru generarea de numere aleatoare, atît individual cît și în masive.

**Funcția *random***

*Random* este o funcție generală, care poate genera una sau mai multe valori aleatoare, conform unei distribuții de probabilitate specificată. Primul parametru al funcției este un șir de caractere (dintr-un set predefinit) care identifică distribuția de probabilitate – de exemplu ’norm’ sau ’Normal’ pentru distribuția normală, ’poiss’ sau ’Poisson’ pentru distribuția Poisson etc. Fiecare din distribuțiile posibile sînt caracterizate de 1, 2 sau 3 parametri, care urmează după numele distribuției – de exemplu, pentru distribuția normală sînt necesari doi parametri, în ordine, media și abaterea standard. Opțional, după acești parametri pot să apară și alții, care definesc dimensiunile masivului cu numere aleatoare care trebuie generat. În lipsa lor se generează un singur număr.

**Exemplu**

Apelul următor generează un masiv 4\*5 cu numere aleatoare generate conform distribuției normale de medie 1 și abatere standard 2:

>> random('norm',1,2,4,5)

ans =

2.0753 1.6375 8.1568 2.4508 0.7517

4.6678 -1.6154 6.5389 0.8739 3.9794

-3.5177 0.1328 -1.6998 2.4295 3.8181

2.7243 1.6852 7.0698 0.5901 3.8344

**Funcțiile *unifrnd* și *unidrnd***

Funcția *unifrnd* generează numere aleatoare reale conform distribuției uniforme, din intervalul (sau intervalele) precizat(e). Se poate genera un singur număr aleatoriu sau mai multe numere (un masiv). În al doilea caz, pentru fiecare număr se poate folosi același interval sau intervale diferite.

**Exemple**

Generarea unui număr aleator din intervalul (0,1):

>> unifrnd(0,1)

ans =

0.6557

Generarea unui masiv 3\*3 cu numere aleatoare din intervalul (0,1):

>> unifrnd(0,1,3,3)

ans =

0.0357 0.6787 0.3922

0.8491 0.7577 0.6555

0.9340 0.7431 0.1712

Generarea unui vector cu 9 numere aleatoare, fiecare fiind ales din intervalul (i,i+1), unde *i* ia valori de la 0 la 8:

>> a=0:8

a =

0 1 2 3 4 5 6 7 8

>> b=1:9

b =

1 2 3 4 5 6 7 8 9

>> unifrnd(a,b)

ans =

0.4387 1.3816 2.7655 3.7952 4.1869 5.4898 6.4456 7.6463 8.7094

Generarea unui vector cu 5 numere aleatoare, fiecare fiind ales din intervalul (0,i), unde *i* ia valori de la 1 la 5:

>> a=1:5

a =

1 2 3 4 5

>> unifrnd(0,a)

ans =

0.7547 0.5521 2.0391 2.6204 0.8131

Funcția *unidrnd* generează numere aleatoare întregi pozitive, conform distribuției uniforme, din intervalul (0,N), unde N este primul parametru al funcției. Dacă N este un masiv, se va genera un masiv cu aceleași dimensiuni, pentru fiecare element folosind ca valoare maximă elementul de pe poziția corespunzătoare din N.

**Exemple**

Generarea unui număr aleator întreg mai mic sau egal cu 5:

>> unidrnd(5)

ans =

1

Generarea unui vector cu 5 numere aleatoare întregi, pentru fiecare din ele maximul fiind, respectiv, 1, 2, 3, 4 și 5:

>> unidrnd([1 2 3 4 5])

ans =

1 2 2 3 2

Generarea unui vector cu 5 numere aleatoare întregi, pentru fiecare din ele maximul fiind 10:

>> unidrnd(10,1,5)

ans =

3 8 8 4 6

Generarea unei matrice 5\*5 cu numere aleatoare întregi, pentru fiecare din ele maximul fiind 10:

>> unidrnd(10,5)

ans =

8 10 9 4 4

3 6 3 2 9

6 2 9 3 6

7 2 3 7 6

9 3 10 5 10

**Funcțiile *rand*, *randn* și *randi***

Aceste funcții generează numere pseudoaleatoare, bazate pe generatorul intern de numere pseudoaleatoare al MatLab. Funcția *rand* generează numere din intervalul (0,1) conform distribuției uniforme. Funcția *randn* generează numere din intervalul (0,1) conform distribuției normale. Parametrii funcțiilor *rand* și *randn* definesc dimensiunile masivelor care trebuie generate. Funcția *randi* generează numere întregi conform distribuției normale, mai mici sau egale cu valoarea primului parametru. Următorii parametri definesc dimensiunea masivului care trebuie generat.

Pentru toate aceste funcții, parametrii care definesc dimensiunile masivului care trebuie generat sînt interpretați astfel:

* nici un parametru – se generează o singură valoare
* un parametru – se generează o matrice pătrată de dimensiunea specificată
* mai mulți parametri – se generează un masiv de dimensiunile specificate.

**Exemple**

>> rand

ans =

0.7943

>> rand(3)

ans =

0.3112 0.6020 0.6892

0.5285 0.2630 0.7482

0.1656 0.6541 0.4505

>> rand(1,5)

ans =

0.0838 0.2290 0.9133 0.1524 0.8258

>> randn

ans =

0.1240

>> randn(4)

ans =

1.4367 2.9080 -0.4686 0.7015

-1.9609 0.8252 -0.2725 -2.0518

-0.1977 1.3790 1.0984 -0.3538

-1.2078 -1.0582 -0.2779 -0.8236

>> randn(1,5)

ans =

-1.5771 0.5080 0.2820 0.0335 -1.3337

>> randi(10)

ans =

9

>> randi(10,2)

ans =

7 6

4 5

>> a=5

a =

5

>> b=6

b =

6

>> c=a+(b-a).\*rand(1,5)

c =

5.0760 5.2399 5.1233 5.1839 5.2400