1. Ce este MATLAB-ul?

MATLAB-ul (denumirea vine de la "**mat**rix **lab**oratory") este un mediu computaţional, dezvoltat de firma The Math Works Inc., dedicat calculului numeric şi vizualizării datelor prin intermediul reprezentărilor grafice. Este de asemenea un pachet de programe de înaltă performanţă şi un limbaj de programare al cărui element de bază este matricea (scalar, vector, matrice sau tablouri multi-dimensionale). El integrează analiza numerică, calculul matriceal, procesarea semnalelor şi reprezentările grafice într-un mediu uşor de învăţat şi de folosit.

MATLAB-ul include aplicaţii specifice, numite TOOLBOX-uri. Acestea sunt colecţii extinse de funcţii MATLAB (fişiere-M) care dezvoltă mediul de programare de la o versiune la alta, pentru a rezolva probleme din domenii variate. Structural, **MATLAB**-ul este realizat sub forma unui nucleu de bază, cu interpretor propriu, în jurul căruia sunt construite toolbox-urile. Astfel, una dintre facilităţile pe care le oferă MATLAB-ul constă în posibilitatea de extindere, care constă în crearea unor noi funcţii în limbajul MATLAB care pot fi folosite în acelaşi mod precum funcţiile predefinite în mediul MATLAB. De fapt, toolbox-urile care extind funcţionalitatea MATLAB-ului sunt definite în acest mod.

2. Modul de lucru în MATLAB

Putem scrie comenzi în mod interactiv în linia de comandă, caz în care fiecare linie este prelucrată imediat şi rezultatele sunt afişate, sau le putem salva în fişiere-M. Fişierele ce conţin instrucţiuni MATLAB se numesc fişiere-M (deoarece au extensia „.m”) şi sunt programe MATLAB. Un fişier-M constă dintr-o succesiune de instrucţiuni MATLAB; având posibilitatea de a apela alte fişiere-M şi a apelării recursive.

Un program MATLAB poate fi scris sub forma fişierelor „script” sau a fişierelor „function” (aceste tipuri de fişiere obligatoriu cu extensia „.m” permit crearea de noi funcţii care le pot completa pe cele existente). Prin această facilitate, MATLAB-ul poate fi extins la aplicaţii specifice utilizatorului, care are posibilitatea să scrie noi funcţii.

3. Utilizarea help-ului în MATLAB

Una dintre posibilităţile de a obţine informaţii legate de funcţiile MATLAB sau realizate de un utilizator al pachetului de programe este utilizarea sistemului de *help* a MATLAB-ului. Funcţiile din această secţiune permit obţinerea informaţiilor de interes general referitoare la mediul de lucru MATLAB.

Modalităţi de obţinere de informaţii utilizând help-ul în MATLAB:

a. ***în linia de comandă***: tastăm *help nume*, unde *nume* poate fi un nume de funcţie sau un nume de director. Dacă acesta este un nume de funcţie în linia de comandă vor apărea informaţiile de care avem nevoie despre funcţia căutată, dar acestea nu vor conţine toate posibilităţile şi utilizările acesteia. Se furnizează de asemenea nu doar informaţii despre funcţia căutată, ci se oferă şi trimiteri către alte funcţii înrudite. Dacă *nume* este nume de director, *help-ul* afişează fişierele conţinute în directorul specificat.
b. ***din meniu***: fişierele help pot fi accesate şi folosind help-ul din meniu. Informaţiile pot fi obţinute folosind *index*.

4. Gestiunea fişierelor şi a zonei de memorie

MATLAB-ul reţine comenzile folosite şi valorile variabilelor create în timpul unei sesiuni. Aceste variabile sunt reţinute în zona de memorie a MATLAB-ului numită *workspace*. Valorile acestor variabile pot fi aflate tastând în linia de comandă numele variabilei fără a folosi vreun semn de punctuaţie la sfârşitul acesteia. Trebuie reţinut faptul că MATLAB-ul este *case sensitive*, deci Temp, temp sau TEMP reprezintă variabile diferite. În MATLAB, comenzile utilizate apar într-o fereastră separată numită *command history*. Acestea pot fi reutilizate sau reeditate în linia de comandă folosind săgeţile.

- **Funcţii pentru controlul directoarelor şi fişierelor**

Comanda	utilizare
dir, ls	afişează numele tuturor fişierelor din directorul curent sau din orice alt director precizat ca argument
delete nume_fişier	permite ştergerea unui fişier sau a unui grafic
cd, pwd	returnează numele directorului curent
cd cale/nume_director	schimbă directorul
type nume_fişier	afişează fişierul nume_fişier pe ecran, fără a-l putea modifica
edit nume_fişier	returnează fişierul nume_fişier în care se pot face modificări
which nume_fişier	returnează calea în care este localizat un fişier sau o funcţie MATLAB. Această comandă poate fi utilizată pentru a determina dacă un fişier face parte dintr-un pachet MATLAB standard
what	returnează fişierele **.m, *.mat, *.mex* din directorul curent

- **Comenzi utilizate în gestionarea workspace-ului**

Comanda	utilizare
who	listează variabilele curente din memorie
whos	listează variabilele curente, dimensiunile lor, precum şi tipul lor (reale sau complexe)
clear	şterge toate variabilele din memorie

clear x y	şterge variabilele x şi y din memorie

5. Matrice, vectori şi scalari. Declaraţii şi variabile.

Elementul de bază cu care lucrează MATLAB-ul este matricea. În MATLAB scalarii sunt asimilaţi matricelor de dimensiune *1 × 1* şi vectorii sunt asimilaţi matricelor de dimensiune *1 × n* sau *n × 1*. Elementele unei matrice *A* pot fi identificate prin notaţia *A*(*i*,*j*) şi semnifică elementul de la intersecţia liniei *i* cu coloana *j*. Elementele unei matrice pot fi numere reale (tipul de bază în MATLAB este double) sau complexe, precum şi orice expresie MATLAB.

Introducerea explicită (de la tastatură) a unei matrice se realizează ţinând cont de următoarele reguli:

- Elementele unei linii trebuie separate prin blank-uri sau virgule;
- Liniile se separă prin punct-virgulă „;” ;
- Elementele matricei sunt cuprinse între paranteze drepte “[ ]”.

Exemplu: *O matrice introdusă cu secvenţa A* = [*1 2 3; 4 5 6*] *returnează rezultatul:*

A=

1 2 3

4 5 6

Obs: Indicii matricei încep de la 1.

Dacă se asignează o valoare unui element care ocupă o poziţie în afara dimensiunii maxime a matricei sau vectorului referit, dimensiunea acestuia este mărită automat până la valoarea indicelui noului element, iar elementele nedefinite sunt setate la valoarea zero.

Exemplu: *Fie A* = [*1 2; 3 4*]

Instrucţiunea A(*3,3*) *= 5 generează:*

A = [*1 2 0; 3 4 0; 0 0 5*]

Obs: Folosirea operatorului “:” permite utilizarea sau afişarea unei linii/coloane fără a parcurge linia/coloana respectivă.

Exemplu: *Pentru matricea A = [1 2; 3 4]*

A(2,:) returnează linia

3 4, iar

A(:,1) afişează

1

3

MATLAB-ul este un limbaj de expresii. Expresiile tipărite de utilizator sunt interpretate şi utilizate. Orice instrucţiune se termină în mod normal cu Enter. Dacă ultimul caracter al instrucţiunii este punct-virgulă „;”, instrucţiunea este executată, dar tipărirea rezultatului este suprimată. Dacă expresia este aşa de mare încât declaraţia nu încape pe o singură linie, se utilizează semnul „...”(trei puncte) urmat de Enter, pentru a preciza că instrucţiunea se continuă pe linia următoare.

6. Generarea vectorilor

Pentru a genera un vector cu pas liniar MATLAB -ul oferă două metode:

• Dacă se cunosc limitele intervalului (*xmin* şi *xmax*) şi pasul (*pas*) dintre două elemente, se generează vectorul cu instrucţiunea:

x = xmin : pas : xmax

Dacă pasul e negativ atunci e necesar ca *xmin* > *xmax*. Dacă se omite specificarea valorii pasului, atunci acesta va fi luat implicit egal cu *1*.

• Dacă se cunosc limitele intervalului (amin şi amax) şi numărul de elemente (N) ale vectorului generat cu pas liniar, atunci se foloseşte instrucţiunea:

x = linspace(*xmin, xmax, N*)

Dacă valoarea lui *N* este omisă, implicit se va lua *100*.

Exerciţiu: *Fie v = [1 2 3 4] si fie A = [1 2 3 4; 5 6 7 8; 9 10 11 12]. Cu ajutorul comenzii >>help nume_funcţie aflaţi ce calculează fiecare din următoarele comenzi MATLAB:*

w1=max(v),B1=min(A)

w2=mean(v),B2=mean(A)

w3=median(v),B3=median(A)

w4=sum(v);w5=cumsum(A)

w6=prod(v);w7=cumprod(A)

w8=sort(v), [*w9, w10*]*=sort(v), B4=sort(A)*

size(v),length(A)

7. Matrice speciale

Anumite matrice des utilizate sunt disponibile în MATLAB ca funcţii utilitare:

>>*eye(n),* unde *n* este un număr natural, returnează o matrice identitate de dimensiune $n \times n$. Dacă *n* este un număr întreg negativ, va fi returnată o matrice nulă. Dacă sintaxa este de forma *eye(A)* , unde *A* este o matrice, atunci va fi returnată o matrice identitate de dimensiune egală cu dimensiunea matricei *A*. Dacă avem sintaxa *eye(m, n)*, va fi creată o matrice identitate de dimensiunea maximă posibilă, iar restul va fi completat cu zerouri până se ajunge la o matrice dimensiunea $m \times n$.

>>*ones(n),* unde *n* este un număr natural, returnează o matrice de dimensiune $n \times n$ cu toate elementele egale cu *1*. Dacă *n* este un număr întreg negativ, va fi returnată o matrice nulă. Dacă sintaxa este de forma *ones(A)* , unde *A* este o matrice, atunci va fi returnată o matrice cu toate elementele egale cu *1*, de dimensiune egală cu dimensiunea matricei A. Dacă avem sintaxa *ones(m,n)*, va fi creată o matrice dimensiunea $m \times n$, având toate elementele egale cu *1*.

>>*zeros(n)* **,** unde *n* va fi un număr natural, returnează o matrice de dimensiune $n \times n$ cu toate elementele egale cu zero. Restul observaţiilor de la matricea anterioară sunt valabile şi aici, cu diferenţa că de fiecare dată matricea generată va avea toate elementele egale cu zero.

>>*rand(n)* **,** unde *n* va fi un număr natural, returnează o matrice de dimensiune $n \times n$ având ca elemente numere aleatoare uniform distribuite între *0* şi *1*.

>>*randn(n)* **,** unde n va fi un număr natural, returnează o matrice de dimensiune $n \times n$ având ca elemente numere distribuite normal standard (*media = 0, dispersia =1*).

>>g=[]

va genera o matrice *g* de dimensiune 0, dar care va exista în spaţiul de lucru.

8. Calcul numeric cu MATLAB -ul

Calculele aritmetice asupra tablourilor de date în MATLAB pot fi:

- operaţii după regulile calculului matriceal – operaţii cu matrice;
- operaţii după regulile calculului scalar – operaţii cu tablouri.

Operaţiile cu tablouri sunt operaţii aritmetice (înmulţire, împărţire, ridicare la putere, etc.) între elementele situate în aceeaşi poziţie a tablourilor, cunoscute sub numele de operaţii element cu element. Pentru efectuarea operaţiilor cu tablouri se folosesc aceiaşi operatori ca în operaţiile cu scalari, precedaţi de semnul punct ”.”, semn ce indică efectuarea operaţiilor în ordinea element cu element. Pentru a putea fi efectuate operaţii element cu element, trebuie ca dimensiunile tablourilor cu care se operează să fie identice. Dacă unul dintre operanzi este un scalar, acesta operează cu fiecare element al tabloului.

În cazul operaţiilor de adunare şi scădere, operatorul nu va mai fi precedat de punct.

Exercitiu: *Fie: A = [2 2 5; 1 4 7], B = [3 4 1; 2 5 3], p = 2. Să se calculeze: C=A+B, D = B - A, E = p - A, F = B - p, G = p + A.*

În cazul operaţiei de înmulţire, pentru a preciza că înmulţirea se efectuează element cu element, între componentele a două matrice de aceleaşi dimensiuni, se utilizează operatorul de înmulţire precedat de punct (.*). Instrucţiunea este de forma:

$$C = A .* B$$

Exercitiu: *Fie A = [1 3 2], B = [3 4 6], p = 3. Să se calculeze, folosind calculul element cu element, înmulţirea matricelor A şi B, înmulţirea scalarului p cu matricea A şi înmulţirea matricei B cu scalarul p.*

Operaţia de împărţire la dreapta, element cu element, între două tablouri este simbolizată cu operatorul punct–slash (./). Instrucţiunea este de forma:

$$C = A ./ B$$

şi reprezintă împărţirea la dreapta, element cu element, a tablourilor A şi B, cu aceleaşi dimensiuni, rezultând elementele:

$$C(i,j) = A(i,j) / B(i,j)$$

Operaţia de împărţire la stânga, element cu element, între două tablouri este simbolizată cu operatorul punct–backslash (.\). Instrucţiunea este de forma:

$$C = A.\backslash B$$

şi reprezintă împărţirea la stânga, element cu element, a tablourilor A şi B, cu aceleaşi dimensiuni, rezultând un tablou cu elementele:

$$C(i,j)=A(i,j)\backslash B(i,j)=B(i,j)/A(i,j)$$

Prin urmare: $C = A .\backslash B = B ./ A$

Operaţia de ridicare la putere, element cu element, într-un tablou este simbolizată cu operatorul punct-^ (.^). Instrucţiunea este de forma:

$$C = A.\hat{}B$$

şi reprezintă ridicarea fiecărui element din tabloul *A* la puterea indicată de valoarea elementului din aceeaşi poziţie a tabloului *B*, adică:

$$C(i,j)=A(i,j)\hat{}B(i,j)$$

Dacă *A* e un scalar, se lasă un blank între scalar şi operatorul de ridicare la putere.

Operaţia de transpunere a unui tablou este simbolizată de operatorul punct-apostrof. Instrucţiunea este de forma:

$$B=A.'$$

şi liniile tabloului A vor deveni coloanele tabloului transpus *B*. Acest lucru face ca un tablou *B* , cu dimensiunea $m \times n$, să devină un tablou A cu dimensiunea $n \times m$.

Operaţiile uzuale de algebră liniară cu matrice sunt simbolizate cu semnele grafice: *, /, \, ^, ’, şi se efectuează după regulile cunoscute din calculul matriceal.

Exerciţiu: *Fie A=[1 2; 3 4], B=[5 6; 7 8]. Să se calculeze:*

*C=A/B, D=A\B, E=A^2, F=A./B, G=A.\B, H=A.^2, I=A+2, J=B*4, K=A-2.*

Cu ajutorul comenzii >>help nume_functie aflaţi ce calculează fiecare dintre următoarele comenzi MATLAB:

v=[1 2 3], V=diag(v), W=diag(v,2)

A=[1 2 3;4 5 6;7 8 9], B=diag(diag(A))

C=inv(A), d=det(B), t=trace(B)

D=[1 2 3 4 5 6], E=D(3:5), F=D(2:2:6)

H=[1 2 3 4 5 6;2 3 4 5 6 1;3 4 5 6 1 2;...

4 5 6 1 2 3; 5 6 1 2 3 4; 6 1 2 3 4 5]

I=H(2,:), J=H(:,3), K=H(1:2,4:6), L=H([1,4],[2,4:5]),

m=1:4, n=2:2:6, M=H(m,n)

9. Tipuri de date

Matlab-ul nu conţine multe tipuri de date spre deosebire de multe alte limbaje de programare, dar prezintă totuşi şi alte tipuri înafară de matrice şi string-uri. Cele mai importante sunt:

- tablourile multidimensionale
- matricele de celule
- structurile

Tablouri multidimensionale

Matricele nu sunt restricţionate la două dimensiuni. De exemplu pot fi definite matrice tridimensionale.

Exemplu: Pentru a defini un tablou de dimensiune $2\times3\times4$ de 1 se poate folosi comanda:

>>A = ones(2,3,4); $2\times3\times4$

Matrice de celule

Matricele de celule sunt structuri cu o mai mare flexibilitate, deoarece pot conţine elemente de orice tip (chiar şi alte matrice de celule) şi pot fi de dimensiuni diferite. Matricea de celule are o structură generală similară cu cea a matricelor de date de bază. De exemplu, o matrice de celule 2 × 3 va avea două linii care vor conţine fiecare câte 3 celule. Totuşi, elementele matricei pot fi de dimensiuni sau/şi tipuri diferite. O celulă poate conţine o dată de tip char, alta o dată de tip double sau altele pot fi goale. *O altă caracteristică particulară este aceea că operaţiile matematice nu sunt definite pe mulţimea matricelor de celule.*

Voi prezenta câteva dintre căile de acces la elementele matricei de celule. Pentru a obţine conţinutul unei celule a matricei A se foloseşte notaţia A{ , }, iar pentru a obţine celula se foloseşte notaţia uzuală folosită pentru a accesa un element al matricei. De exemplu, A{1,1} reprezintă conţinutul celulei (de tip double sau char), iar A(1,1) reprezintă insăşi celula şi conţine o dată de tip **cell**. Combinând cele două notaţii putem avea acces la elementele celulei. De exemplu, pentru a obţine primele două elemente ale celulei A{1,1}, presupunând că ea conţine un vector, le putem accesa prin A{1,1} (1:2).
Pentru a construi o matrice de celule folosim comanda:
>>cell(m,n) care returnează un tablou de matrice de dimensiune 0, de dimensiune $m \times n$ după care vom iniţializa elementele matricei de celule.

Exemplu 1: *Comanda **cell(2,3)** va returna o matrice de celule de dimensiune 2 × 3 ale cărei elemente sunt matrice de dimensiune 0.*

```
>> cell(2,3)

ans =

    []    []    []
    []    []    []
```

Pentru a iniţializa de exemplu primul element al matricei cu matricea [1 2 3; 3 4 5] vom folosi comanda:

```
>> a{1,1}=[1 2 3; 2 3 4]

a =
```

[2x3 double] [] []

[] [] []

Exerciţiu: *Accesaţi elementul aflat pe poziţia (1,2) al primei celule.*

Exemplu 2: *Consider matricea X = [1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15; 16 17 18 19 20], folosind funcţia* ***mat2cell*** *(consultaţi help-ul pentru a înţelege cum este apelată această funcţie) împart matricea X într-o matrice conţinând patru celule.*

C = mat2cell(X, [2 2], [3 2]) reprezintă matricea formată din cele patru celule.

C =

[2x3 double] [2x2 double]

[2x3 double] [2x2 double]

Cele patru celule se accesează prin

C{1,1}

ans =

1 2 3

6 7 8

C{1,2}

ans =

4 5

9 10

C{2,1}

ans =

11 12 13

16 17 18

C{2,2}

ans =

14 15

19 20

Exerciţiu: *Să se afişeze prima linie a primei celule, a doua coloană a celei de a doua celule şi primele două elemente ale primei linii din celula a treia.*

O structură reprezintă un ansamblu de variabile la care se face referire cu un singur nume, ceea ce reprezintă un mijloc confortabil de a menţine la un loc informaţii din aceeaşi sferă. Variabilele care alcătuiesc structura sunt denumite membrii sau câmpurile acelei structuri.

O structură este declarată folosind funcţia **struct** astfel:

S = STRUCT('câmp1',valoare1,'câmp2',valoare2,...)

şi returnează o structură având câmpurile şi valorile corespunzătoare specificate prin argumentele setate. Tablourile de valori trebuie să fie tablouri de celule de aceeaşi dimensiune, celule scalare sau simple valori. Tablourile de valori sunt atribuite câmpurilor corespunzătoare.

Exemplu: s = struct('nume',{{'Matei','Andrei'}},'cod',[1 2]);

10. Operatori logici

Matlab-ul dispune de operatorii relaţionali obişnuiţi.

Verificarea egalităţii a doi operanzi se realizează folosind operatorul "= =" , iar operatorul care verifică inegalitatea este notat "~=". Rezultatul operaţiei relaţionale este o matrice conţinând 0 şi 1, în care fiecare intrare adevărată are valoarea 1 şi fiecare intrare falsă conţine valoarea 0.

Funcţia Matlab **find** returnează indicii elementelor matricei care satisfac anumite condiţii logice; aceasta poate fi folosită pentru a selecta şi pentru a modifica elementele care satisfac anumite condiţii.

Exemplu: *Exemplul de mai jos mai întâi determină elementele matricei Y care sunt mai mari decât 3 şi apoi înlocuieşte valorile acestor elemente cu 10.*

```
>> I = find( Y > 3 );
>>Y(I) = 10 * ones(size(I));
```

11. Reprezentări grafice elementare în MATLAB

>>plot(x,y) plotează vectorul x versus vectorul y (adică reprezintă punctele (x(i),y(i)) şi uneşte două puncte consecutive prin linii drepte). Vectorii trebuie să aibă aceeaşi lungime.

>>subplot(*m,n,i*) împarte ecranul în $m \times n$ ferestre şi introduce plotul curent în fereastra *i*.

Exerciţiu: *Fie codul Matlab următor:*

```
x=-pi:0.1:pi;
y=sin(x);
subplot(2,2,1);
```

```
plot(x,y,'or');
xlabel('x');
ylabel('sin(x)'); title('Graficul functiei sin(x)');
```

Realizaţi în celelalte 3 ferestre graficele funcţiilor cos(x), arcsin(u), arccos(u) (u=-1:0.01:1) utilizând o varietate cât mai mare de linii si culori.

Observaţii:

1. Axele sunt scalate implicit astfel încât să se permită vizualizarea întregului grafic. Pentru a avea mai mult control asupra graficului se poate utiliza funcţia **axis**.

   Exerciţiu: *Să se ploteze funcţia sinus la care se adaugă un zgomot repartizat uniform pe intervalul [0,1] pe intervalul [0,1]. Să se seteze intervalul pe care trebuie plotată funcţia folosind funcţia Matlab* ***axis.***
2. De fiecare dată când este apelată funcţia **plot**, din figură sunt eliminate (şterse) informaţiile anterioare. Pentru a adăuga informaţii noi unei figuri (precum legende şi etichete ale axelor) sau grafice suplimentare se utilizează funcţia **hold on** după prima apelare a funcţiei **plot**.
3. Alte tipuri de plot-uri bidimensionale disponibile sunt:
   - **semilogy**: utilizează o scară logaritmică pe axa *y* şi o scară liniară pe axa *x*;
   - **semilogx**: utilizează o scară logaritmică pe axa *x* şi o scară liniară pe axa *y*;
   - **loglog**: utilizează o scară logaritmică pe ambele axe;
   - **imagesc**: reprezintă o matrice printr-o imagine colorată
   - **hist**: afişează histograma unei mulţimi de date.

4. Pentru a reprezenta grafic o funcţie bidimensională, de exemplu, $z = f(x,y)$ putem folosi plot-uri tridimensionale. Pentru a realiza acest grafic, vom construi mai întâi o matrice Z a cărei *i, j* element reprezintă valoarea funcţiei peste o reţea având coordonatele corespunzătoare axelor *x* şi respectiv *y* reţinute în matricele X şi Y.

Funcţiile Matlab care se folosesc sunt:

- **meshgrid**: transformă domeniul specificat prin vectorii x şi y în matricele X şi Y folosite pentru a evalua funcţii de două variabile şi în plotarea

suprafeţelor. Liniile matricei X sunt copii ale vectorului x, iar coloanele matricei Y sunt copii ale vectorului y.

- **surf** şi **mesh:** reprezintă o funcţie de două variabile ca o suprafaţă şi respectiv ca o reţea.
- **contour**: reprezintă punctele suprafeţei aflate la nivelul z, într-un spaţiu bidimensional, adică sunt puncte din mulţimea A = $\{x \in Domf \mid f(x) = z\}$.

Exerciţiu: *Să se reprezinte grafic funcţia de două variabile* $f(x,y) = x\exp(-x^2 - y^2)$ *şi respectiv liniile de contur folosind funcţia* ***contour*** *pe domeniul* $[-2,2]\times[-2,2]$.

12. Fişierele script şi fişierele funcţie

Fişierele script

Un fişier „*script*” este un fişier-M care conţine o secvenţă de comenzi MATLAB. Prin apelarea numelui fişierului în linia de comandă, se execută secvenţa MATLAB conţinută în acesta, executându-se fiecare comandă ca şi cum ar fi tastată interactiv în linia de comandă. Comenzile unui fişier script au acces la zona de memorie principală, iar după execuţia completă a unui fişier script, variabilele cu care acesta a operat rămân în zona de memorie. Fişierele script sunt folosite pentru rezolvarea unor probleme care cer comenzi succesive atât de lungi, încât ar putea deveni greoaie pentru lucrul în mod interactiv, adică în modul linie de comandă. De asemenea, utilizarea lor este recomandată pentru executarea unor experimente.

Fişierele funcţie

Dacă prima linie a fişierului-M conţine cuvântul „*function*”, fişierul respectiv este declarat ca fişier funcţie. O funcţie diferă de un „*script*” prin faptul că poate lucra cu argumente. Variabilele definite şi manipulate în interiorul fişierului funcţie sunt localizate la nivelul acesteia, deci fişierele funcţie au o zonă de memorie proprie. Prin

urmare, la terminarea execuţiei unei funcţii, în memoria calculatorului nu rămân decât variabilele de ieşire ale acesteia. Comunicarea de informaţii dintre zona de memorie alocată funcţiei şi zona de memorie principală se face prin intermediul parametrilor de intrare şi al parametrilor de ieşire. Fişierele funcţie sunt utilizate pentru extinderea MATLAB -ului, adică pentru crearea unor funcţii noi MATLAB.

Forma generală a primei linii a unui fişier funcţie este:

function [param_ieşire 1, …, param_ieşire m] = ***nume_funcţie***(param_intrare 1,…, param_intrare n)

unde:

function – este cuvântul cheie care declară fişierul ca fişier funcţie (prezenţa lui este obligatorie);

nume_funcţie – numele funcţie, adică numele sub care se salvează fişierul fără extensie. Trebuie să avem grijă să nu coincidă cu cel al unui fişier deja existent;

param_ieşire 1, …, param_ieşire m – parametrii de ieşire care trebuie separaţi prin virgule şi cuprinşi între paranteze drepte. Dacă funcţia nu are parametri de ieşire, parantenzele drepte şi semnul egal nu mai au sens.

param_intrare 1, ,…, param_intrare n – parametrii de intrare care trebuie separaţi prin virgule şi cuprinşi între paranteze rotunde. Dacă funcţia nu are parametri de intrare, parantenzele rotunde şi semnul egal nu mai au sens.

Aceste fişiere pot fi adăugate ca funcţii noi în MATLAB.

Comenzile şi funcţiile care sunt utilizate de noua funcţie sunt înregistrate într-un fişier cu extensia .m. Dacă vrem să introducem un comentariu, în cadrul unui fişier, vom preceda comentariul respectiv de semnul procent "%" . Acest lucru e foarte util de reţinut, întrucât, atunci când creăm un fişier funcţie, este bine ca pe liniile care urmează imediat după linia de declarare a fişierului funcţie, să introducem un comentariu prin care să dăm informaţii despre fişierul respectiv. Astfel, atunci când un alt utilizator doreşte să afle informaţii despre fişierul respectiv, poate tasta:

>>help nume_fisier

şi pe ecranul de comenzi va apărea comentariul introdus în fişier.

Exerciţiu: *Screţi o funcţie cu numele medie care primeşte ca argumente 2 numere a şi b şi returnează media aritmetică, media geometrică şi media armonică a numerelor a şi b. Scrieţi o a doua funcţie cu numele medie_gen care primeşte ca argumente un vector şi returnează media aritmetică si media geometrică a componentelor. Scrieţi pentru ambele funcţii comentarii si rolul lor.*

Există o funcţie Matlab specială, pe care o vom folosi atunci când avem de calculat valorile unei funcţii particulare în mai multe puncte şi nu dorim să o reţinem într-un fişier separat. Această funcţie este denumită **funţie inline** şi creează un obiect inline, având ca argument un string care reprezintă anumite expresii matematice sau comenzi pe care dorim ca Matlab-ul să le execute. Ca argumente opţionale putem specifica argumentele obiectului funcţie inline. De exemplu, variabila **func** din exemplul de mai jos reprezintă un obiect inline.

func = inline('sin(2*pi*f + theta)','f','theta');

Această funcţie calculează $\sin(2\pi f + \theta)$ pe baza variabilelor de intrare f şi θ şi poate fi apelată precum orice funcţie Matlab astfel:

```
x = 0:.1:4*pi;
theta= pi/2;
ys = func(x, theta);
```

13. Instrucţiuni şi funcţii de control în programe

Instrucţiunile de control logic în MATLAB sunt următoarele:

if instrucţiune pentru executarea condiţionată a unui set de instrucţiuni;

else clauză asociată cu „if”;

elseif clauză asociată cu „if”;

for instrucţiune pentru realizarea ciclurilor cu un număr determinat de repetări;

while instrucţiune pentru realizarea ciclurilor pe baza unei condiţii logice;

break instrucţiune pentru terminarea forţată a unui ciclu;

return instrucţiune pentru returnarea execuţiei în modulul apelant;

error instrucţiune ce permite afişarea unui mesaj de eroare;

end instrucţiune pentru încheierea ciclurilor „for”, „while” şi „if”.

Pe lângă structura de secvenţă, aceste instrucţiuni de control permit realizarea unor structuri de program fundamentale (ciclul for”, ciclul „while” şi „if..else”) ce permit programarea structurată în MATLAB.

În MATLAB este indicată utilizarea programării vectoriale (i.e. prelucrarea întregului tablou, fără a folosi instrucţiuni care realizează cicluri pentru a opera asupra unui element) din motive de eficienţă.

Instrucţiunea „if” simplă, clauza „else”, clauza „elseif”

În cadrul unui program este uneori necesară executarea unor grupuri de instrucţiuni numai dacă o condiţie exprimată printr-o expresie logică are o anumită valoare de adevăr. Instrucţiunea **„if”** are rolul de a permite construirea unei structuri de program pentru executarea condiţionată a unor grupuri de instrucţiuni.

Instrucţiunea **„if”** poate fi folosită ca instrucţiune simplă în relaţie cu clauza **„end”** sau ca instrucţiune complexă, caz în care include şi clauzele **„else”** şi **„elseif”**.

Forma generală a unei instrucţiuni **„if”** simplă este următoarea:

if *expresie_logică*
    secvenţă_de_instrucţiuni
end

care are următoarea interpretare: dacă expresia_logică are valoarea adevărat atunci se execută secvenţa_de_instrucţiuni care urmează până la clauza **„end”**, altfel dacă expresia_logică este falsă se trece la executarea instrucţiunilor care urmează după clauza **„end”**.

Forma generală a instrucţiunii **„if”** poate fi combinată cu clauza **„else”** (obţinem astfel instrucţiunea **„if-else“**) ca în exemplul următor:

if *expresie_logică*

secvenţă_de_instrucţiuni1

else

secvenţă_de_instrucţiuni2

end

Dacă avem nevoie de mai multe nivele de instrucţiuni **„if-else”** este recomandată folosirea clauzei **„elseif”**, cu sau fără clauza **„else”**.

if *expresie_logică_1*

secvenţă_de_instrucţiuni 1

elseif *expresie_logică_2*

secvenţă_de_instrucţiuni 2

elseif *expresie_logică_3*

secvenţă_de_instrucţiuni 3

……

elseif *expresie_logică_n*

secvenţă_de_instrucţiuni n

end

if *expresie_logică_1*

secvenţă_de_instrucţiuni 1

elseif *expresie_logică_2*

secvenţă_de_instrucţiuni 2

elseif *expresie_logică_3*

secvenţă_de_instrucţiuni 3

……

else

secvenţă_de_instrucţiuni n

end

Instrucţiunea repetitivă FOR

Această instrucţiune permite repetarea de un număr determinat de ori a unui grup de instrucţiuni şi are următoarea structură generală:

for *index = iniţial : pas : final*
    secvenţă de instrucţiuni repetate
end

Instrucţiunea repetitivă WHILE

Această comandă permite execuatarea repetată a unui grup de instrucţiuni de un număr nedeterminat de ori sub controlul unei condiţii logice. Forma generelă este:

while *expresie*
    secvenţă de instrucţiuni
end

Instrucţiunile din bucla while sunt executate atâta timp cât expresie are elemente nenule. De regulă *expresie* are o singură valoare TRUE sau FALSE.

Instrucţiunea break

Instrucţiunea break se utilizează pentru a ieşi dintr-o buclă înainte ca aceasta să se fi terminat. Se recomandă a fi utilizată dacă o condiţie de eroare este detectată în interiorul unei bucle. Instrucţiunea break încetează execuţia ciclurilor for şi while. În cazul unor cicluri imbricate, break determină ieşirea din ciclul cel mai interior. Se apelează cu sintaxa:

break

Instrucţiunea return

Instrucţiunea return determină o ieşire normală din fişierul-M către funcţia care l-a apelat sau către tastatură. Se apelează cu sintaxa:

return

Instrucţiunea error

Instrucţiunea error permite afişarea unor mesaje la întâlnirea unei erori. Se apelează cu sintaxa:

error(‘mesaj’)

Exemplu: procedura următoare verifică dacă funcţia test a fost apelată cu două argumente de intrare şi semnalează eroare dacă nu este îndeplinită această condiţie:

```
function test(x,y)
if nargin~=2
        error(‘Numărul argumentelor de intrare este greşit’)
end.
```

Exerciţii:

Exerciţiul 1: Scrieţi codul MATLAB care trasează graficul funcţiei următoare:

$$f : [-10, 10] \to R,\ f(x) = \begin{cases} 2x + 8 & \text{dacă } x \le 2 \\ 3x^2 & \text{dacă } x > 2 \end{cases}$$

Exerciţiu 2: Să se genereze o matrice A cu n linii şi n+1 coloane ale cărei elemente sunt:

$$A(i,j) = \begin{cases} 2, \text{dacă } i = j \\ -1, \text{dacă } |i - j| = 1 \\ 0, \text{ în rest} \end{cases}$$

Exerciţiu 3: Să se scrie un program, utilizând o buclă while care calculează suma elementelor vectorului x = [2 -3 8 3 2 1 -5 9 7] până când se întâlneşte un număr mai mare ca 8.

14. Şiruri de caractere

1. Funcţii generale

Constanta şir se reprezintă printr-un şir de octeţi în care se păstrează codurile ASCII ale caracterelor şirului respectiv. Codurile ASCII 0 – 31 sunt caractere negrafice, iar codurile 32 – 127 sunt caractere grafice. Un şir de caractere este constituit din unul sau mai multe caractere delimitate prin apostrofuri. Apostrofurile nu fac parte din şirul de caractere. Pentru a introduce 'apostroful' într-un şir de caractere se scriu două apostrofuri.

Funcţiile MATLAB folosite pentru operarea cu şiruri de caractere sunt:

- **abs**

Funcţia **abs** converteşte şirurile de caractere la valorile numerice ale codului ASCII; se apelează cu sintaxa:

$$y = abs('şir\_de\_caractere')$$

- **setstr**

Funcţia **setstr** returnează caracterul corespunzător codului ASCII dat ca argument; se apelează cu sintaxa:

$$s = setstr(t)$$

- **isstr**

Detectarea şirurilor de caractere se face cu funcţia **isstr** , care se apelează cu sintaxa:

$$a = isstr(t)$$

şi returnează 1 pentru modul text şi 0 pentru modul cod numeric ASCII.

- **eval**

Funcţia **eval** interpretează şirurile de caractere conţinând expresii MATLAB; se apelează cu una dintre sintaxele:

x = eval('şir_de_caractere') sau *eval('şir_de_caractere')*

- **feval**

Funcţia **feval** permite evaluarea funcţiilor; se apelează cu una dintre sintaxele:

feval('nume_funţie',x1,..., xn)

[y1,…, ym] = feval('nume_funţie',x1,..., xn)

2. Funcţii de comparare a şirurilor de caractere

- **strcmp**

Funcţia **strcmp** compară două şiruri de caractere; se apelează cu sintaxa:

$$a = strcmp(S_1,S_2)$$

şi returnează 1 dacă acestea sunt identice şi 0 în caz contrar. Funcţia lucrează în mod senzitiv, adică include în procesul de comparare spaţiile dintre caractere şi tipul caracterelor – mari sau mici – folosite.

- **findstr**

Căutarea unui şir de caractere, S_2 într-un alt şir de caractere, S_1, se face cu funcţia **findstr**, care se apelează cu sintaxa:

$$M = findstr(S_1,S_2)$$

- **upper**

Conversia literelor mici dintr-un şir de caractere, în litere mari, se face cu funcţia **upper** care se apelează cu funcţia:

$$t = upper(s)$$

- **lower**

Conversia inversă a literelor mari în litere mici se face cu funcţia **lower**, care se apelează cu sintaxa:

$$t = lower(s)$$

- **isletter**

Determinarea literelor alfabetului dintr-un şir de caractere se face cu funcţia **isletter**, care se apelează cu sintaxa:

$$A = isletter(S)$$

Rezultatul conţine 1 în poziţiile în care elementele şirului sunt litere şi 0 în celelalte poziţii (numere, blank-uri, etc).

3. Conversia şir-număr şi număr-şir

- **num2str**

Conversia numerelor în şiruri de caractere este utilizată la introducerea valorilor numerice în titluri, la etichetarea axelor şi se face cu funcţia **num2str** care se apelează cu sintaxa:

$$s = num2str(x)$$

- **int2str**

Funcţia **int2str** converteşte numerele întregi în şiruri de caractere; se apelează cu sintaxa:

$$s = num2str(x)$$

- **str2num**

Funcţia **str2num** converteşte şirurile care au caracterele ASCII ale valorilor numerice, în numere; se apelează cu sintaxa:

$$x = str2num(s)$$

Exerciţii:

1) Să se determine codurile ASCII ale şirului de caractere “MATLAB”.
2) Să se afişeze caracterele ASCII grafice într-o matrice 3×32.
3) Să se evalueze expresia: $y = 2x^2 + 3$ în valorile x = 1:10.
4) Să se genereze 3 matrice cu elemente numere aleatoare cu numele test1, test2, test3.
5) Realizaţi conversia şirului s = 123.25e-2 în număr.