

Laborator 1

1. Ce este MATLAB-ul?

MATLAB-ul (denumirea vine de la "**matrix laboratory**") este un mediu computațional, dezvoltat de firma The Math Works Inc., dedicat calculului numeric și vizualizării datelor prin intermediul reprezentărilor grafice. Este de asemenea un pachet de programe de înaltă performanță și 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 și reprezentările grafice într-un mediu ușor de învățat și de folosit.

MATLAB-ul include aplicații specifice, numite TOOLBOX-uri. Acestea sunt colecții extinse de funcții MATLAB (fișiere-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 facilitățile pe care le oferă MATLAB-ul constă în posibilitatea de extindere, prin crearea unor noi funcții în limbajul MATLAB care pot fi folosite în același mod precum funcțiile predefinite în mediul MATLAB. De fapt, toolbox-urile care extind funcționalitatea 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 și rezultatele sunt afișate, sau le putem salva în fișiere-M. Fișierele ce conțin instrucțiuni MATLAB se numesc fișiere-M (deoarece au extensia „.m”) și sunt programe MATLAB. Un fișier-M constă dintr-o succesiune de

instrucțiuni MATLAB; având posibilitatea de a apela alte fișiere-M și a apelării recursive.

Un program MATLAB poate fi scris sub forma fișierelor „script” sau a fișierelor „function” (aceste tipuri de fișiere obligatoriu cu extensia „.m” permit crearea de noi funcții care le pot completa pe cele existente). Prin această facilitate, MATLAB-ul poate fi extins la aplicații specifice utilizatorului.

3. Utilizarea help-ului în MATLAB

Una dintre posibilitățile de a obține informații legate de funcțiile MATLAB sau realizate de un utilizator al pachetului de programe este utilizarea sistemului de *help* a MATLAB-ului. Funcțiile din această secțiune permit obținerea informațiilor de interes general referitoare la mediul de lucru MATLAB.

Modalități de obținere de informații utilizând *help*-ul în MATLAB:

- a. **în linia de comandă:** tastăm *help nume*, unde *nume* poate fi un nume de funcție sau un nume de director. Dacă acesta este un nume de funcție în linia de comandă vor apărea informațiile de care avem nevoie despre funcția căutată, dar acestea nu vor conține toate posibilitățile și utilizările acesteia. Se furnizează de asemenea nu doar informații despre funcția căutată, ci se oferă și trimiteri către alte funcții înrudite. Dacă *nume* este nume de director, *help-ul* afișează fișierele conținute în directorul specificat.
- b. **din meniu:** fișierele *help* pot fi accesate și folosind *help-ul* din meniu. Informațiile pot fi obținute 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
<i>dir, ls</i>	afişează numele tuturor fişierelor din directorul curent sau din orice alt director precizat ca argument
<i>delete nume_fişier</i>	permite ştergerea unui fişier sau a unui grafic
<i>cd, pwd</i>	returnează numele directorului curent
<i>cd cale/nume_director</i>	schimbă directorul
<i>type nume_fişier</i>	afişează fişierul nume_fişier pe ecran, fără a-l putea modifica
<i>edit nume_fişier</i>	returnează fişierul nume_fişier în care se pot face modificări
<i>which nume_fişier</i>	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
<i>what</i>	returnează fişierele *.m, *.mat, *.mex din

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

Comanda	utilizare
<i>who</i>	listează variabilele curente din memorie
<i>whos</i>	listează variabilele curente, dimensiunile lor, precum și tipul lor (reale sau complexe)
<i>clear</i>	șterge toate variabilele din memorie
<i>clear x y</i>	șterge variabilele x și y din memorie

5. Matrice, vectori și scalari. Declarații și variabile.

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

Introducerea explicită (de la tastatură) a unei matrice se realizează ținând 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 secvența $A = [1 \ 2 \ 3; 4 \ 5 \ 6]$ 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 poziție 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 $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

Instrucțiunea $A(3,3) = 5$ generează:

$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 0 \\ 0 & 0 & 5 \end{bmatrix}$

Obs: Folosirea operatorului ":" permite utilizarea sau afișarea unei linii/coloane fără a parcurge linia/coloana respectivă.

Exemplu: Pentru matricea $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

$A(2,:)$ returnează linia

3 4, iar

$A(:,1)$ afișează

1

3

MATLAB-ul este un limbaj de expresii. Expresiile tipărite de utilizator sunt interpretate și utilizate. Orice instrucțiune se termină în mod normal cu Enter. Dacă ultimul caracter al instrucțiunii este punct-virgulă „;”, instrucțiunea este executată, dar tipărirea rezultatului este suprimată. Dacă expresia este așa de mare încât declarația nu încapă pe o singură linie, se utilizează semnul „...” (trei puncte) urmat de Enter, pentru a preciza că instrucțiunea 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 ($xmin$ și $xmax$) și pasul (pas) dintre două elemente, se generează vectorul cu instrucțiunea:

$$x = x_{min} : pas : x_{max}$$

Dacă pasul e negativ atunci e necesar ca $x_{min} > x_{max}$. Dacă se omite specificarea valorii pasului, atunci acesta va fi luat implicit egal cu 1.

- Dacă se cunosc limitele intervalului (x_{min} și x_{max}) și numărul de elemente (N) ale vectorului generat cu pas liniar, atunci se folosește instrucțiunea:

$$x = \text{linspace}(x_{min}, x_{max}, N)$$

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

Exercițiu: Fie $v = [1\ 2\ 3\ 4]$ și fie $A = [1\ 2\ 3\ 4; 5\ 6\ 7\ 8; 9\ 10\ 11\ 12]$. Cu ajutorul comenzii `>>help nume_funcție` aflați ce calculează fiecare din următoarele comenzi MATLAB:

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

$$w2 = \text{mean}(v), B2 = \text{mean}(A)$$

$$w3 = \text{median}(v), B3 = \text{median}(A)$$

$$w4 = \text{sum}(v); w5 = \text{cumsum}(A)$$

$$w6 = \text{prod}(v); w7 = \text{cumprod}(A)$$

$$w8 = \text{sort}(v), [w9, w10] = \text{sort}(v), B4 = \text{sort}(A)$$

$$\text{size}(v), \text{length}(A)$$

6. Matrice speciale

Anumite matrice des utilizate sunt disponibile în MATLAB ca funcții 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 observațiilor de la matricea anterioară sunt valabile și aici, cu diferența 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 și 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 spațiul de lucru.

8. Calcul numeric cu MATLAB – ul

Calcululele aritmetice asupra tablourilor de date în MATLAB pot fi:

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

Operațiile cu tablouri sunt operații aritmetice (înmulțire, împărțire, ridicare la putere, etc.) între elementele situate în aceeași poziție a tablourilor, cunoscute sub numele de operații element cu element. Pentru efectuarea operațiilor cu

tablouri se folosesc aceiași operatori ca în operațiile cu scalari, precedați de semnul punct ".", semn ce indică efectuarea operațiilor în ordinea element cu element. Pentru a putea fi efectuate operații 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 operațiilor de adunare și scădere, operatorul nu va mai fi precedat de punct.

Exercitiu: Fie: $A = \begin{bmatrix} 2 & 2 & 5 \\ 1 & 4 & 7 \end{bmatrix}$, $B = \begin{bmatrix} 3 & 4 & 1 \\ 2 & 5 & 3 \end{bmatrix}$, $p = 2$. Să se calculeze: $C=A+B$, $D = B - A$, $E = p - A$, $F = B - p$, $G = p + A$.

În cazul operației de înmulțire, pentru a preciza că înmulțirea se efectuează element cu element, între componentele a două matrice de aceleași dimensiuni, se utilizează operatorul de înmulțire precedat de punct (*). Instrucțiunea este de forma:

$$C = A .* B$$

Exercitiu: Fie $A = \begin{bmatrix} 1 & 3 & 2 \end{bmatrix}$, $B = \begin{bmatrix} 3 & 4 & 6 \end{bmatrix}$, $p = 3$. Să se calculeze, folosind calculul element cu element, înmulțirea matricelor A și B , înmulțirea scalarului p cu matricea A și înmulțirea matricei B cu scalarul p .

Operația de împărțire la dreapta, element cu element, între două tablouri este simbolizată cu operatorul punct-slash (/). Instrucțiunea este de forma:

$$C = A ./ B$$

și reprezintă împărțirea la dreapta, element cu element, a tablourilor A și B , cu aceleași dimensiuni, rezultând elementele:

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

Operația de împărțire la stânga, element cu element, între două tablouri este simbolizată cu operatorul punct-backslash (\). Instrucțiunea este de forma:

$$C = A .\ B$$

și reprezintă împărțirea la stânga, element cu element, a tablourilor A și B, cu aceleași dimensiuni, rezultând un tablou cu elementele:

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

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

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

$$C = A.^B$$

și reprezintă ridicarea fiecărui element din tabloul A la puterea indicată de valoarea elementului din aceeași poziție a tabloului B, adică:

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

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

Operația de transpunere a unui tablou este simbolizată de operatorul punct-apostrof. Instrucțiunea este de forma:

$$B=A.'$$

și 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 $n \times m$.

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

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

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

Cu ajutorul comenzii `>>help nume_functie` aflați 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 conține multe tipuri de date spre deosebire de multe alte limbaje de programare, dar prezintă totuși și alte tipuri în afară de matrice și string-uri. Cele mai importante sunt:

- tablourile multidimensionale
- matricele de celule
- structurile

Tablouri multidimensionale

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

Exemplu: Pentru a defini un tablou de dimensiune $2 \times 3 \times 4$ de 1 se poate folosi comanda:

```
>>A = ones(2,3,4); 2×3×4
```

Matrice de celule

Matricele de celule sunt structuri cu o mai mare flexibilitate, deoarece pot conține elemente de orice tip (chiar și alte matrice de celule) și 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 conține fiecare câte 3 celule. Totuși, elementele matricei pot fi de dimensiuni sau/și tipuri diferite. O celulă poate conține o dată de tip char, alta o dată de tip double sau altele pot fi goale. O altă caracteristică particulară este aceea că operațiile matematice nu sunt definite pe mulțimea matricelor de celule.

Voi prezenta câteva dintre căile de acces la elementele matricei de celule. Pentru a obține conținutul unei celule a matricei A se folosește notația $A\{ , \}$, iar

pentru a obține celula se folosește notația uzuală folosită pentru a accesa un element al matricei. De exemplu, $A\{1,1\}$ reprezintă conținutul celulei (de tip double sau char), iar $A(1,1)$ reprezintă însăși celula și conține o dată de tip **cell**. Combinând cele două notații putem avea acces la elementele celulei. De exemplu, pentru a obține primele două elemente ale celulei $A\{1,1\}$, presupunând că ea conține 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 inițializa 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 inițializa 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] [] []
[] [] []

```

Exercițiu: Accesați elementul aflat pe poziția (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 funcția **mat2cell** (consultați help-ul pentru a înțelege cum este apelată această funcție) împart matricea X într-o matrice conținând 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

Exercițiu: *Să se afișeze prima linie a primei celule, a doua coloană a celei de a doua celule și 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 menține la un loc informații din aceeași sferă. Variabilele care alcătuiesc structura sunt denumite membrii sau câmpurile acelei structuri.

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

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

și returnează o structură având câmpurile și valorile corespunzătoare specificate prin argumentele setate. Tablourile de valori trebuie să fie tablouri de celule de aceeași 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 relaționali obișnuiți.

Verificarea egalității a doi operanzi se realizează folosind operatorul “==” , iar operatorul care verifică inegalitatea este notat “~=”. Rezultatul operației relaționale este o matrice conținând 0 și 1, în care fiecare intrare adevărată are valoarea 1 și fiecare intrare falsă conține valoarea 0.

Funcția Matlab **find** returnează indicii elementelor matricei care satisfac anumite condiții logice; aceasta poate fi folosită pentru a selecta și pentru a modifica elementele care satisfac anumite condiții.

Exemplu: *Exemplul de mai jos mai întâi determină elementele matricei Y care sunt mai mari decât 3 și apoi înlocuiește 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)) și unește două puncte consecutive prin linii drepte). Vectorii trebuie să aibă aceeași lungime.

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

Exercițiu: *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)');
```

Realizați în celelalte 3 ferestre graficele funcțiilor $\cos(x)$, $\arcsin(u)$, $\arccos(u)$ ($u=-1:0.01:1$) utilizând o varietate cât mai mare de linii și culori.

Observații:

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

Exercițiu: Să se ploteze funcția 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 funcția folosind funcția Matlab **axis**.

2. De fiecare dată când este apelată funcția **plot**, din figură sunt eliminate (șterse) informațiile anterioare. Pentru a adăuga informații noi unei figuri (precum legende și etichete ale axelor) sau grafice suplimentare se utilizează funcția **hold on** după prima apelare a funcției **plot**.
3. Alte tipuri de plot-uri bidimensionale disponibile sunt:
 - **semilogy**: utilizează o scară logaritmică pe axa y și o scară liniară pe axa x ;
 - **semilogx**: utilizează o scară logaritmică pe axa x și o scară liniară pe axa y ;
 - **loglog**: utilizează o scară logaritmică pe ambele axe;
 - **imagesc**: reprezintă o matrice printr-o imagine colorată
 - **hist**: afișează histograma unei mulțimi de date.

4. Pentru a reprezenta grafic o funcție 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 funcției peste o rețea având coordonatele corespunzătoare axelor x și respectiv y reținute în matricele X și Y .

Funcțiile Matlab care se folosesc sunt:

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

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

- **surf** și **mesh**: reprezintă o funcție de două variabile ca o suprafață și respectiv ca o rețea.
- **contour**: reprezintă punctele suprafeței aflate la nivelul z, într-un spațiu bidimensional, adică sunt puncte din mulțimea $A = \{x \in \text{Dom}f \mid f(x) = z\}$.

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

12. Fișierele script și fișierele funcție

Fișierele script

Un fișier „*script*” este un fișier-M care conține o secvență de comenzi MATLAB. Prin apelarea numelui fișierului în linia de comandă, se execută secvența MATLAB conținută în acesta, executându-se fiecare comandă ca și cum ar fi tastată interactiv în linia de comandă. Comenzile unui fișier script au acces la zona de memorie principală, iar după execuția completă a unui fișier script, variabilele cu care acesta a operat rămân în zona de memorie. Fișierele 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.

Fișierele funcție

Dacă prima linie a fișierului-M conține cuvântul „*function*”, fișierul respectiv este declarat ca fișier funcție. O funcție diferă de un „*script*” prin faptul că poate lucra cu argumente. Variabilele definite și manipulate în interiorul fișierului funcție

sunt localizate la nivelul acesteia, deci fișierele funcție au o zonă de memorie proprie. Prin urmare, la terminarea execuției unei funcții, în memoria calculatorului nu rămân decât variabilele de ieșire ale acesteia. Comunicarea de informații dintre zona de memorie alocată funcției și zona de memorie principală se face prin intermediul parametrilor de intrare și al parametrilor de ieșire. Fișierele funcție sunt utilizate pentru extinderea MATLAB -ului, adică pentru crearea unor funcții noi MATLAB.

Forma generală a primei linii a unui fișier funcție este:

function [param_ieșire 1, ..., param_ieșire m] = ***nume_funcție***(param_intrare 1,..., param_intrare n)

unde:

function – este cuvântul cheie care declară fișierul ca fișier funcție (prezența lui este obligatorie);

nume_funcție – numele funcție, adică numele sub care se salvează fișierul fără extensie. Trebuie să avem grijă să nu coincidă cu cel al unui fișier deja existent;

param_ieșire 1, ..., param_ieșire m – parametrii de ieșire care trebuie separați prin virgule și cuprinși între paranteze drepte. Dacă funcția nu are parametri de ieșire, parantezele drepte și semnul egal nu mai au sens.

param_intrare 1, ..., param_intrare n – parametrii de intrare care trebuie separați prin virgule și cuprinși între paranteze rotunde. Dacă funcția nu are parametri de intrare, parantezele rotunde și semnul egal nu mai au sens.

Aceste fișiere pot fi adăugate ca funcții noi în MATLAB.

Comenzile și funcțiile care sunt utilizate de noua funcție sunt înregistrate într-un fișier cu extensia .m. Dacă vrem să introducem un comentariu, în cadrul unui fișier, vom preceda comentariul respectiv de semnul procent “%” . Acest lucru e foarte util de reținut, întrucât, atunci când creăm un fișier funcție, este bine ca pe liniile care urmează imediat după linia de declarare a fișierului funcție, să introducem un comentariu prin care să dăm informații despre fișierul respectiv. Astfel, atunci când un alt utilizator dorește să afle informații despre fișierul respectiv, poate tasta:

>>help nume_fisier

și pe ecranul de comenzi va apărea comentariul introdus în fișier.

Exercițiu: *Scrieți o funcție cu numele medie care primește ca argumente 2 numere a și b și returnează media aritmetică, media geometrică și media armonică a numerelor a și b. Scrieți o a doua funcție cu numele medie_gen care primește ca argumente un vector și returnează media aritmetică și media geometrică a componentelor. Scrieți pentru ambele funcții comentarii și rolul lor.*

Există o funcție Matlab specială, pe care o vom folosi atunci când avem de calculat valorile unei funcții particulare în mai multe puncte și nu dorim să o reținem într-un fișier separat. Această funcție este denumită **funcție inline** și 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 opționale putem specifica argumentele obiectului funcție inline. De exemplu, variabila **func** din exemplul de mai jos reprezintă un obiect inline.

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

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

x = 0:1:4*pi;

theta= pi/2;

ys = func(x, theta);

12. Instrucțiuni și funcții de control în programe

Instrucțiunile de control logic în MATLAB sunt următoarele:

if instrucțiune pentru executarea condiționată a unui set de instrucțiuni;

else clauză asociată cu „if”;

elseif clauză asociată cu „if”;

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

while instrucțiune pentru realizarea ciclurilor pe baza unei condiții logice;

break instrucțiune pentru terminarea forțată a unui ciclu;

return instrucțiune pentru returnarea execuției în modulul apelant;

error instrucțiune ce permite afișarea unui mesaj de eroare;

end instrucțiune pentru încheierea ciclurilor „for”, „while” și „if”.

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

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

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

În cadrul unui program este uneori necesară executarea unor grupuri de instrucțiuni numai dacă o condiție exprimată printr-o expresie logică are o anumită valoare de adevăr. Instrucțiunea „if” are rolul de a permite construirea unei structuri de program pentru executarea condiționată a unor grupuri de instrucțiuni.

Instrucțiunea „if” poate fi folosită ca instrucțiune simplă în relație cu clauza „end” sau ca instrucțiune complexă, caz în care include și clauzele „else” și „elseif”.

Forma generală a unei instrucțiuni „if” simplă este următoarea:

```
if expresie_logică
    secvență_de_instrucțiuni
end
```

care are următoarea interpretare: dacă expresia_logică are valoarea adevărat atunci se execută secvența_de_instrucțiuni care urmează până la clauza „end”, altfel

dacă expresia_logică este falsă se trece la executarea instrucțiunilor care urmează după clauza „**end**”.

Forma generală a instrucțiunii „**if**” poate fi combinată cu clauza „**else**” (obținem astfel instrucțiunea „**if-else**”) ca în exemplul următor:

```
if expresie_logică
    secvență_de_instrucțiuni1
else
    secvență_de_instrucțiuni2
end
```

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

```
if expresie_logică_1
    secvență_de_instrucțiuni 1
elseif expresie_logică_2
    secvență_de_instrucțiuni 2
elseif expresie_logică_3
    secvență_de_instrucțiuni 3
.....
elseif expresie_logică_n
    secvență_de_instrucțiuni n
end
```

```
if expresie_logică_1
    secvență_de_instrucțiuni 1
elseif expresie_logică_2
    secvență_de_instrucțiuni 2
elseif expresie_logică_3
    secvență_de_instrucțiuni 3
```

```
.....  
else  
    secvență_de_instrucțiuni n  
end
```

Instrucțiunea repetitivă FOR

Această instrucțiune permite repetarea de un număr determinat de ori a unui grup de instrucțiuni și are următoarea structură generală:

```
for index = inițial : pas : final  
    secvență de instrucțiuni repetate  
end
```

Instrucțiunea repetitivă WHILE

Această comandă permite executarea repetată a unui grup de instrucțiuni de un număr nedeterminat de ori sub controlul unei condiții logice. Forma generelă este:

```
while expresie  
    secvență de instrucțiuni  
end
```

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

Instrucțiunea break

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

```
break
```

Instrucțiunea return

Instrucțiunea return determină o ieșire normală din fișierul-M către funcția care l-a apelat sau către tastatură. Se apelează cu sintaxa:

```
return
```

Instrucțiunea error

Instrucțiunea error permite afișarea unor mesaje la întâlnirea unei erori. Se apelează cu sintaxa:

```
error('mesaj')
```

Exemplu: procedura următoare verifică dacă funcția test a fost apelată cu două argumente de intrare și semnalează eroare dacă nu este îndeplinită această condiție:

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

Exerciții:

Exercițiul 1: Scrieți codul MATLAB care trasează graficul funcției următoare:

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

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

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

Exercițiu 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 întâlnește un număr mai mare ca 8.

14. Șiruri de caractere

1. Funcții generale

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

Funcțiile MATLAB folosite pentru operarea cu șiruri de caractere sunt:

➤ **abs**

Funcția **abs** convertește șirurile de caractere la valorile numerice ale codului ASCII; se apelează cu sintaxa:

$$y = abs('șir_de_caractere')$$

➤ **setstr**

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

$$s = setstr(t)$$

➤ **isstr**

Detectarea șirurilor de caractere se face cu funcția **isstr**, care se apelează cu sintaxa:

$$a = isstr(t)$$

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

➤ **eval**

Funcția **eval** interpretează șirurile de caractere conținând expresii MATLAB; se apelează cu una dintre sintaxele:

$$x = eval('șir_de_caractere') \quad \text{sau} \quad eval('șir_de_caractere')$$

➤ **feval**

Funcția **feval** permite evaluarea funcțiilor; se apelează cu una dintre sintaxele:

$$feval('nume_funție', x_1, \dots, x_n)$$

$$[y_1, \dots, y_m] = feval('nume_funție', x_1, \dots, x_n)$$

2. Funcții de comparare a șirurilor de caractere

➤ **strcmp**

Funcția **strcmp** compară două șiruri de caractere; se apelează cu sintaxa:

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

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

➤ **findstr**

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

$$M = \text{findstr}(S_1, S_2)$$

➤ **upper**

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

$$t = \text{upper}(s)$$

➤ **lower**

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

$$t = \text{lower}(s)$$

➤ **isletter**

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

$$A = \text{isletter}(S)$$

Rezultatul conține 1 în pozițiile în care elementele șirului sunt litere și 0 în celelalte poziții (numere, blank-uri, etc).

3. Conversia șir-număr și număr-șir

➤ **num2str**

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

$$s = \text{num2str}(x)$$

➤ **int2str**

Funcția **int2str** convertește numerele întregi în șiruri de caractere; se apelează cu sintaxa:

$$s = \text{num2str}(x)$$

➤ **str2num**

Funcția **str2num** convertește șirurile care au caracterele ASCII ale valorilor numerice, în numere; se apelează cu sintaxa:

$$x = \text{str2num}(s)$$

Exerciții:

- 1) Să se determine codurile ASCII ale șirului de caractere "MATLAB".
- 2) Să se afișeze 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) Realizați conversia șirului $s = 123.25e-2$ în număr.

Bibliografie:

1. M. Ghinea, V. Fireșteanu, Matlab. Calcul numeric. Grafică. Aplicații., Editura Teora, 2007