LABORATOR 1&2: INTRODUCERE IN MATLAB

1. PREZENTARE MATLAB

- **Limbaj performant** folosit la rezolvarea numerică şi/sau analitică (i.e. prin calcul simbolic) a problemelor de matematică
- Contine: **calcul computational**, **vizualizarea** (**reprezentare grafica**) **rezultatetor**, **programare** – toate intr-un mediu usor de utilizat si in care problemele si solutiile acestora sunt exprimate prin notatii matematice familiare
- **Sistem interactiv**, avand ca elemente de baza **tablourile** (**arrays**) ale caror dimensiuni nu trebuie declarate aprioric
- **MATLAB** = **MAT**rix **LAB**oratory – a fost creat initial pentru un calcul matricial facil
- Contine pachete de programe/software (colectii de functii **MATLAB**, numite **fisiere de tip M**) specializate pe anumite domenii, numite **toolboxes**, e.g. procesare de imagini, controlul sistemelor, retele neuronale, sisteme fuzzy, wavelets, simulare etc.
- Pe platforma Windows, o sesiune MATLAB **se deschide** prin dublu click pe icon-ul MATLAB
- **Ferestrele MATLAB:**

FEREASTRA (WINDOW)	SCOP
Command Window Fereastra de comenzi	Fereastră principală - se pot introduce variabile, evalua expresii, rula programe
Figure Window Fereastra editare figuri	Se deschide automat când sunt executate comenzi grafice si contine outputul acestora
Editor Window Fereastra de editare	Se pot crea si verifica/corect fişierele program (script) si/sau funcţiilor
Help Window Fereastră Help	Contine help-ul MATLAB-ului
Command History Window Fereastra de istoric al comenzilor	Stocheaza comenzile utilizate in fereastra de comenzi
Workspace Window Fereastra pentru spaţiul de lucru	Furnizeaza date si/sau informaţii despre folosirea variabilelor
Current Directory Window	Contine fisierele din directorul curent

- O sesiune MATLAB **se inchide** prin comanda `exit` sau `quit` introdusa in fereastra de comanda (**Command Window**)
- Fişierele program (script) au extensia '*.m', iar figurile '*.fig'
- Rularea si salvarea unui program: **Debug /Save and Run** (**F5**)
- Daca lipsesc anumite ferestre, acestea se pot deschide din meniul **Desktop**
- Daca la sfarsitul unei comenzi MATLAB se tipareste caracterul punct si virgula (;), nu se afiseaza outputul comenzii; in caz contrar, se afiseaza outputul comenzii
- Simbolul procent (%) tiparit inaintea unei linii – Linia devine linie comentariu (nu este perceputa ca o comanda MATLAB)
- Comanda MATLAB **`clc`** sterge continutul ferestrei de comanda (**Command Window**)

2. OPERATII ARITMETICE CU SCALARI

+	adunare;	-	scădere;		
*	inmulţire;	/	impărţire la dreapta;	\	impărţire la stânga;
^	ridicare la putere;				

Ordinea operatiilor

1. Operatiile din paranteze (de la interior spre exterior)
2. Ridicarea la putere

3. Inmultirea. Impartirea
4. Adunarea. Scaderea

3. FUNCTII MATEMATICE PREDEFINITE

FUNCTIA	DESCRIERE	EXEMPLU
`sqrt(x)`	$\sqrt{x}$	`>> sqrt(81)`
`exp(x)`	e^x	`>> exp(5)`
`abs(x)`	$\|x\|$	`>> abs(-24)`
`log(x)`	$\ln(x)$	`>> log(1000)`
`log10(x)`	$\log_{10}(x)$	`>> log10(1000)`
`factorial(x)`	$x!$	`>> factorial(5)`
`sin(x)`	$\sin(x)$, x in radiani	`>> sin(pi/6)`
`cos(x)`	$\cos(x)$, x in radiani	`>> cos(pi/6)`
`tan(x)`	$\tan(x)$, x in radiani	`>> tan(pi/6)`
`cot(x)`	$\cot(x)$, x in radiani	`>> cot(pi/6)`
`sind(x)`	$\sin(x)$, x in grade	`>> sin(30)`
`cosd(x)`	$\cos(x)$, x in grade	`>> cos(30)`
`tand(x)`	$\tan(x)$, x in grade	`>> tan(30)`
`cotd(x)`	$\cot(x)$, x in grade	`>> cot(30)`
`round(x)`	Rotunjire la cel mai apropiat întreg	`>> round(17/5)`
`fix(x)`	Rotunjire spre 0	`>> fix(13/5)`
`ceil(x)`	Rotunjire spre $+\infty$	`>> ceil(11/5)`
`floor(x)`	Rotunjire spre $-\infty$	`>> floor(-9/4)`
`rem(x,y)`	Restul împărţirii x/y	`>> rem(13,5)`

4. FORMATUL NUMERELOR

COMANDA	DESCRIERE	EXEMPLU
`format short`	Virgula fixa, 4 zecimale, $0.001 \le$ numar ≤ 1000	`27.1828`
`format long`	Virgula fixa, 15 zecimale, $0.001 \le$ numar ≤ 1000	`27.18281828459045`
`format short e`	Virgula mobila, 4 zecimale (format stiintific)	`2.7183e+001`
`format long e`	Virgula mobila, 15 zecimale (format stiintific)	`2.718281828459045e+001`
`format short g`	Cel mai bun format dintre virgula fixa sau virgula mobila, cu 5 zecimale	`27.183`
`format long g`	Cel mai bun format dintre virgula fixa sau virgula mobila, cu 15 zecimale	`27.1828182845905`
`format bank`	Virgula fixa, 2 zecimale	`27.18`
`format compact`	Elimina liniile libere pentru a permite afisajul pe ecran a mai multor linii de mesaj	
`format loose`	Adauga linii libere (opusul formatului `compact`).	

Exercitiu:
Afisati expresia 1/7 in MATLAB folosind formatele disponibile si comenzile de mai jos:

```
>> clear
>> a = 1/7;
>> format short;
>> a
```

5. VARIABILE SCALARE

Variabila este un nume alcatuit dintr-o litera sau o combinatie de cel mult 63 de caractere (litere, cifre si semnul _), care incepe mereu cu o litera.
Unei variabile ii este desemnata/atribuita o valoare numerica, poate fi folosita in expresii matematice, functii si declaratii sau comenzi MATLAB.
Operatorul de atribuire: Semnul egal (=)

Instructiunea de atribuire

var = expr

var - numele variabilei
expr – valoare numerica sau expresia atribuita variabilei **var**

Daca o variabila a fost deja definita, prin tiparirea numelui acelei variabile si apasarea tastei **ENTER**, vor fi afisate, pe urmatoarele doua linii, variabila si, respectiv, valoarea acesteia.

Variabile predefinite

ans	Variabila de răspuns pentru calculul unei expresii
pi	π
eps	Cea mai mica diferenta intre doua numere: $2^{-52} = 2.2204e-016$
Inf	∞
i	$\sqrt{-1}$
j	$\sqrt{-1}$
NaN	Not-a-Number – folosit cand MATLAB o valoare numerica valida, i.e. atribuit operatiilor 0/0 sau ∞/∞

Comenzi MATLAB utile

COMANDA	SEMNIFICATIE
clear	Sterge din memorie valorile alocate tuturor variabilelor
clear VAR	Sterge din memorie valoarea alocata variabilei VAR
who	Afiseaza lista variablelor curente din memorie
whos	Afiseaza lista variablelor curente din memorie, dimensiunile lor, ca si informatii despre clasa variabilelor si no. bytes alocati

Exercitiu: Se considera urmatoarele variabile: a = 1.0e+308, b = 1.10e+308 si c = -1.001e+308. Calculati in MATLAB urmatoarele expresii: $a + (b + c)$ si $(a + b) + c$.

Exercitiu: Calculati in MATLAB expresia $((1 + x) - 1)/ x$ pentru x = 1.0e-15.

6. TABLOURI (ARRAYS) DE DIMENSIUNE 1: VECTORI

Tabloul (**array**) este forma fundamentala utilizata de MATLAB pentru a stoca si manipula datele.

> **CONVENTII**
> 1. Parantezele patrate [ ] delimiteaza un tablou
> 2. Spatiile sau caracterul virgula (,) separa elementele liniei unei tablou
> 3. Caracterul punct si virgula (;) separa doua linii ale unei tablou

Vectorii sunt considerati tablouri de dimensiune 1 si pot fi **vectori linie** (i.e. matrice de tip $1\times n$) sau **vectori coloana** (i.e. ca matrice de tip $n\times 1$).

Generarea vectorilor

- *Generarea directa a vectorilor* (i.e. prin introducerea elementelor):

```
>> a = [1 5 7 11]        %vector linie ( matrice de tip (1,n) )
>> b = [2,31,47,103]     %vector linie ( matrice de tip (1,n) )
>> c = b'                %transpusul vectorului c
>> d = [2; 4; 6; 8; 10]  %vector coloana( matrice de tip (n,1) )
```

- *Generarea vectorilor cu pasul dat* (operatorul **`:`**)

`vec = xi : step : xf`

xi – primul element al vectorului ***vec***
xf – ultimul emement al vectorului ***vec***
step – pasul dintre doua elemente consecutive ale vectorului ***vec***
***vec*(i) = *xi* + (i - 1)*step, i = 1, 2,…**
***vec*(i) <= *xf*, i = 1, 2,…**

`vec = xi : xf`

xi – primul element al vectorului ***vec***
xf – ultimul emement al vectorului ***vec***
***step* = 1** – se subintelege
***vec*(i) = *xi* + (i - 1), i = 1, 2,…**
***vec*(i) <= *xf*, i = 1, 2,…**

- *Generarea vectorilor cu n elemente liniare echidistante* (comanda MATLAB **`linspace)`**

`vec = linspace(xi, xf, N)`

xi – primul element al vectorului ***vec***
xf – ultimul element al vectorului ***vec***
N – numarul de elemente ale vectorului ***vec***
vec*(i) = *xi* + (i - 1)*(*xf* - *xi*)/(*N* - 1), i = 1,2,…,*N
OBSERVATIE: In acest caz, ***step* = (xf - xi)/(*N* -1)**

Apelarea elementelor unui vector

`v(k)`	Apelarea elementului vectorului `v` de pe pozitia `k`
`v(:)`	Apelarea tuturor elementelor vectorului `v`
`v(m:n)`	Apelarea elementelor vectorului `v` de pe pozitiile `m, m+1,…, n`
`v([k,m,n])`	Apelarea elementelor vectorului `v` de pe pozitiile `k`, `m` si `n`
`v([k,m:n])`	Apelarea elementelor vectorului `v` de pe pozitiile `k` si `m, m+1,…, n`

```
>> va(4)
>> u(20:44)
>> vb([2,7,10])
>> vb([2,7:10])
>> u(5)^u(8) + sqrt(vb(7))
```

Adaugarea de elemente unui vector

```
>> v(5:10) = 10:5:35
>> va(10) = 4
>> vc(5) = 24
```

Stergerea elementelor unui vector

```
>> v = [1, 2, 3, 4, 5, 6]
>> size(v)
>> v(3) = []
>> size(v)

>> w = [1 3 5 7 9 11]
>> size(w)
>> w(2:4) = []
>> w(3) = []
>> size(w)

>> u = linspace(1, 21, 11)
>> size(u)
>> u([2,7,10]) = []
>> size(u)
```

7. TABLOURI (ARRAYS) DE DIMENSIUNE 2: MATRICE

Generarea matricelor

```
>> A = [5 35 43; 4 76 81; 21 32 40]
>> size(A)
>> B = [7 2 76 33 8
1 98 6 25 6
5 54 68 9 0]
>> C = [1:2:11; 0:5:25; linspace(10,60,6); 67 2 43 68 4 13]
>> D = C'                        %D = transpusa matricei C
```

Apelarea elementelor unei matrice

`A(m,n)`	Apelarea elementului de pe linia `m` si coloana `n` ale matricei `A`
`A(:,n)`	Apelarea tuturor elementelor din coloana `n` a matricei `A`
`A(n,:)`	Apelarea tuturor elementelor din linia `n` a matricei `A`
`A(:,m:n)`	Apelarea tuturor elementelor din coloanele `m, m+1,…, n` ale matricei `A`
`A(m:n,:)`	Apelarea tuturor elementelor din liniile `m, m+1,…, n` ale matricei `A`
`A(m:n,p:q)`	Apelarea tuturor elementelor din liniile `m, m+1,…, n` si coloanele `p, p+1,…, q` ale matricei `A`
`A(:,[m,n])`	Apelarea tuturor elementelor din coloanele `m` si `n` ale matricei `A`

`A([m,n],:)`	Apelarea tuturor elementelor din liniile `m` si `n` ale matricei `A`
`A([m,n],[p,q])`	Apelarea tuturor elementelor din liniile `m` si `n` si coloanele `p` si `q` ale matricei `A`

```
>> C(2,3)
>> C(1,:)
>> C(:,2)
>> C(2:4,:)
>> C([2,4],:)
>> C(:,1:3)
>> C(:,[1,3])
>> C(2:4,1:3)
>> C(:)
```

Variabile matriciale predefinite

`zeros(m,n)`	Matricea nula $\mathbf{O} \in \mathbf{R}^{m\times n}$
`ones(m,n)`	Matricea $\mathbf{A} = (a_{ij}) \in \mathbf{R}^{m\times n}$, unde $a_{ij} = 1$
`eye(m,n)`	Matricea $\mathbf{A} = (a_{ij}) \in \mathbf{R}^{m\times n}$, unde $a_{ij} = 1, i = j$ si $a_{ij} = 0$, altfel
`eye(n)`	Matricea identitate $\mathbf{I} = (\delta_{ij}) \in \mathbf{R}^{n\times n}$

OBSERVATIE: Notatia [ ] inseamna matrice de tip 0×0 nedefinita. In consecinta, atribuirea [ ] unei linii sau coloane ale unei matrice este o modalitate eleganta de a sterge/elimina linia sau coloana respectivei matrice. In acest caz, dimensiunile matricei se modifica!

8. OPERATII ARITMETICE CU TABLOURI

Operatorii aritmetici pot fi aplicati matricelor si vectorilor in doua moduri distincte:

a) ca **operatori in sens matricial** – sunt respectate regulile uzuale ale algebrei liniare si aceste operatii sunt obtinute prin utilizarea operatorilor +, -, *, / si ^;
b) ca **operatori cu tablouri** – sunt operatii definite intre tablouri de aceleasi dimensiuni, sunt element cu element si se obtin folosind operatorii +, -, *, / si ^ precedati de simbolul punct (.)

- **Adunarea si scaderea matricelor** se fac in cazul matricelor de acelasi tip, i.e. acelasi numar de linii si de coloane.
- **Inmultirea matricelor** `A` si `B` (`A * B`) se face in concordanta cu regulile algebrei liniare, i.e. numarul de linii ale matricei `A` trebuie sa fie egal cu numarul de coloane ale matricei `B`.
- **Ridicarea la putere a matricelor** se face doar pentru matrice patrate.

Rezolvarea ecuatiilor liniare in MATLAB

a) Consideram sistemul de ecuatii algebrice liniare:

$$\underset{n\times n}{A} \quad \underset{n\times 1}{X} \quad = \quad \underset{n\times 1}{B}$$

Solutia $X = A^{-1}B$ poate fi calculata in MATLAB in doua moduri:

`X = inv(A)*B` (este calculata mai intai A^{-1}, apoi este calculata $A^{-1}B$);
`X = A\B` (solutia este obtinuta numeric prin metoda de eliminare a lui Gauss).

Impartirea la stanga \ este recomandata pentru rezolvarea ecuatiei $AX = B$ intrucat calculul matricei inverse A^{-1} poate fi mai inacurata decat metoda de eliminare a lui Gauss in cazul matricelor de dimensiuni mari.

b) Consideram sistemul de ecuatii algebrice liniare:

$$\underset{1\times n}{Y}\ \underset{n\times n}{C} = \underset{1\times n}{D}$$

Solutia $Y = DC^{-1}$ poate fi calculata in MATLAB in doua moduri:

`Y = D*inv(C)` (este calculata mai intai C^{-1}, apoi este calculata DC^{-1});
`X = D/C` (solutia este obtinuta folosind operatorul **impartire la dreapta** /).

Operatie	Matrice	Tablouri
Adunare	+	+
Scadere	-	-
Inmultire	*	.*
Impartire la stanga	\	.\
Impartire la dreapta	/	./
Ridicare la putere	^	.^

9. TABLOURI CA ARGUMENTE IN FUNCTII MATLAB PREDEFINITE

```
>>x = [0:pi/6:pi]
>>y = cos(x)              %size(y) = size(x)

>>A = [1 4 9; 16 25 36; 49 64 81]
>>det(A)
>>inv(A)
>>E = sqrt(A)             %size€ = size(A)

>>a = [5 9 2 4 11 6 7 11 0 1]
>>b = min(a)              %b = min{x(i)|i=1,…,size(x)}
>>c = max(a)              %c = max{x(i)|i=1,…,size(x)}
>>[d1, n] = max(a)        %d1 = min{x(i)|i=1,…,size(x)} & x(n) = d1
>>[d2, n] = max(a)        %d2 = max{x(i)|i=1,…,size(x)} & x(n) = d2
>>sum(a)
>>sort(a)                 %rearanjare a x(i) in ordine crescatoare
```

10.FISIERE DE TIP M

Fisierele de tip M sunt echivalente cu programele, functiile,subrutinele si procedurile din alte limbaje de programare (FORTRAN, C, C++). Fisierele de tip M sunt fisiere text, cu extensia '.m' si contin comenzi MATLAB. Sunt doua tipuri de astfel de **fisiere de tip M**:

1. **Fisiere script** (**fisiere de comanda**) – nu au argumente de intrare sau de iesire si pot opera pe variabile in spatiul de lucru (workspace).
2. **Fisiere functie** – contin cuvantul **`function`** in linia de definitie (prima linie a fisierului), pot accepta argumente de intrare si returna argumentele de iesire, iar variabilele interne sunt variabile locale pentru functie daca acestea nu au fost declarate, in prealabil, ca variabile globale (folosind comanda MATLAB **`global`**).

FISIERE SCRIPT

- Un fisier script este o insiruire de comenzi MATLAB, fiind numit si program.
- Cand un fisier script este rulat/executat, MATLAB executa comenzile in ordinea in care acestea sunt scrise, ca si cum ar fi scrise in fereastra de comenzi (Command Window).
- Daca un fisier script contine o comanda ce genereaza un output, atunci outputul este afisat in fereastra de comenzi (Command Window).

- Utilizarea unui fisier script file este convenabila deoarece acesta poate fi editat (corectat si/sau schimbat) si rulat/executat de multe ori.
- Fisierele script pot fi editate in orice editor de text ai apoi copiate in editorul din MATLAB.
- Script trebuie sa aiba extensia .m cand sunt salvate.
- Un fisier script poate fi executat tastand numele acestuia in fereastra de comenzi (Command Window) si apoi apasand tasta **ENTER** sau, direct din fereastra de editare (Editor Window), dand click pe iconul **RUN**.

```
%marks.m
%Acest program executa calcule statistice pe o multime de note
exmark = [12 0 5 28 87 3 56];
exsort = sort(exmark)
exmean = mean(exmark)
exstd = std(exmark)
```

Variabile globale

Variabilele globale sunt variabilele care odata create intr-o parte a MATLAB-ului, sunt recunoscute si in alte parti ale MATLAB-ului. Aceasta se intampla si in cazul variabilelor din fereastra de comanda (Command Window) si fisierele script deoarece ambele opereazat asupra variabilelor in spatial de lucru (Workspace). Cand o variabila este definita in fereastra de comanda (Command Window), ea este, de asemenea, recunoscuta si poate fi folosita intr-un fisier script. In mod similar, daca o variabila este definita intr-un fisier script, atunci ea este, de asemenea, recunoscuta si poate fi folosita in fereastra de comanda (Command Window). Cu alte cuvinte, odata ce o variabial este create, ea exista, poate fi folosita si ii poate fi reatribuita o noua valoare atat in fereastra de comanda (Command Window), cat si intr-un fisier script.

Inputul unui fisier script

a) Variabila este definita si ii este alocata o valoare in fisierul script
b) Variabila este definita si ii este alocata o valoare in fereastra de comanda (Command Window)
c) Variabila este in fisierul script, dar o anumita valoare ii este alocata in fereastra de comanda (Command Window), atunci cand fisierul script este rulat/executat

Comenzi pentru output

MATLAB genereaza in mod automat un afisaz atunci cand unele comenzi sunt executate. De exemplu, cand unei variable ii este atribuita o valoare sau numele variabilei careia i s-a atribuit o valoare anterior este scris si apoi este apasata tasta **ENTER**, MATLAB afiseaza variabila si valoarea acesteia. Acest tip de output nu este afisat daca caracterul punct si virgule (;) este tiparit la sfarsitul comenzii. In plus fata de acest afisaz automat, MATLAB poseda diverse comenzi ce pot fi folosite pentru generarea afisajelor. Afisajele pot fi mesaje care furnizeaza o informatie, date numerice sau grafice.

Una dintre comenzile folosite frecvent pentru generarea outputului este comanda MATLAB `disp`. Comanda `disp` este utilizata pentru afisarea elementelor unei variabile fara a afisa numele variabilei respective si pentru afisarea unui text.

```
>> disp(nume_variabila)
>> disp('text ca sir de cractere')
```

FUNCTII SI FISIERE FUNCTIE

In mod frecvent, in cadrul unui program este necesar calculul valorilor unor functii care nu sunt predefinite in MATLAB. Cand o functie este simpla si este necesar calcul acesteia doar o singura data, aceasta poate fi tiparita ca parte a unui program. Cu toate acestea, cand o functie trebuie

evaluate de mai multe ori pentru diferite valori ale argumentelor ei, atunci este convenabila crearea unui fisier functie definit de user pentru functia respectiva. Odata ce aceasta noua functie a fost creata si salvata cu extensia .m, aceasta se poate folosi in mod similar cu functiile predefinite ale limbajului.

O functie definita de user este un program MATLAB creat de user, salvat ca **fisier functie** si care apoi poate fi folosit ca orice alta functie predefinita din MATLAB. Functia poate fi o expresie matematica simpla sau complicata ce implica o serie de calcule. Principala caracteristica a unui fisier functie este ca poseda un input si un output. Atat inputul, cat si outputul poate fi o variabila sau mai multe variabile, fiecare dintre acestea putand fi un scalar, un vector sau un tablou de orice dimensiune. Schematic, un fisier function file poate fi ilustrat astfel:

Crearea unui fisier functie
Functiile sunt create si editate, la fel ca fisierele script, in fereastra de editare (Editor/Debugger Window). Prima linie executabila a fisierului functie trebuie sa fie linia de definitie a functiei, in caz contrar fisierul fiind considerat drept un fisier script. Linia de definitie a functiei:

- Defineste fisierul respectiv drept un fisier functie
- Defineste numele functiei
- Defineste numarul si ordinea argumenetelor de intrare (inputuri) si de iesire (outputuri)

Forma liniei de definitie a functiei este urmatoarea:

`function[`*`output arguments`*`] = `*`function_name`*`(`*`input arguments`*`)`

- Cuvantul **`function`** trebuie sa fie primul cuvant din prima linie a fisierului functie si trebuie tiparit cu litere mici.
- Lista argumentelor de iesire (***`output arguments`***) intre paranteze patrate [ ].
- Un fisier functie se salveaza cu un nume identic cu cel dat functiei (**`function_name`**) asa cum apare in linia de definitie a functiei. In acest mod, functia este apelata (folosita) prin utilizarea numelui functiei (***`function_name`***).
- Lista argumentelor de intrare (***`input arguments`***) este tiparita intre paranteze rotunde ( ).

```
%Fisierul functie se salveaza cu numele “fun1.m”
function y = fun1(x)
y = 1/(1 + x^2);
end

%Apelarea functiei in fereastra de comanda(Command Window)
>> fun1(1.0)                  %Evaluarea functiei fun1 in x = 1.0
```

Comparatie intre fisierele script si fisierele functie

- Atat fisierele script, cat si fisierele functie sunt salvate cu extensia .m
- Prima line intr-un fisier functie este linia de definitie.
- Variablele dintr-un fisier functie sunt **variabile locale**. Variablele dintr-un fisier script file sunt **recunoscute si in fereastra de comanda** (**Command Window**).
- Fisierele script pot folosi variabile definite in spatial de lucru (Workspace).
- Fisierele script contin o insiruire de comenzi (declaratii) MATLAB.

- Fisierele functioe pot accepta date prin intermediul argumentelor de intrare (inputuri) si pot returna date prin intermediul argumentelor de iesire (outputuri).
- Cand un fisier functie este salvat (**`function_name.m`**), numele fisierului functie trebuie sa coincida cu numele functiei (**`function_name`**).

Exercitiu:
Se considera functia de gradul doi $f:\mathbf{R}\to\mathbf{R}$, $f(x)=ax^2+bx+c$, unde $a,b,c\in\mathbf{R}$ si $a\neq 0$. Sa se scrie o functie MATLAB care calculeaza minimul sau maximul local al functiei definite mai sus pe un interval dat $[\alpha,\beta]\subset\mathbf{R}$. Folositi numele functiei si argumentele urmatoare `[fmin,fmax] = minmax(a,b,c,alpha,beta).`

FUNCTII INLINE
Fisierele functie pot fi folosite pentru functii matematice simple, functii matematice complicate ce necesita calcule si programare laborioase, precum si in cazul subprogramelor din cadrul unor programe de calcul foarte mari.

In cazul in care trebuie determinata, de mai multe ori in cadrul unui program, valoarea unei functii matematice relativ simple, MATLAB furnizeaza optiunea folosirii **functiilor inline**. O functie inline este definite in interiorul unui program de calcul (si nu ca un fisier separate, cum este cazul fisielrelor functie) si apoi este folosita in programul de calcul. Functiile inline pot fi definite in orice parte MATLAB-ului.

Functiile inline sunt create cu ajutorul comenzii MATLAB `inline`, in conformitate cu urmatorul format:

fun2 **= inline(*'expr', 'arg1', 'arg2',..., 'argn'*)**

fun2 - numele atribuit functiei
expr - sir de caractere ce contine expresia functiei
arg1 - sir de caractere ce contine argumentul 1 al functiei
arg2 - sir de caractere ce contine argumentul 2 al functiei
……
argn - sir de caractere ce contine argumentul *n* al functiei

- Expresia matematica poate avea una sau mai multe variabile independente.
- Orice litera cu exceptia literelor `i` si `j` poate fi folosita pentru variabilele independente in expresia matematica
- Expresia matematica poate include orice functii predefinite sau definite de user
- Expresia matematica trebuie scrisa in conformitate cu dimensiunea argumentului (fiecare element in parte sau folosind calcule algebrice)
- Expresia matematica nu poate include variable predefinite
- Once the function is defined it can be used by typing its name and a value for the argument (or arguments) in parenthesis.
- Functia `inline` poate fi folosita si ca argument pentru alte functii

```
>> fun2 = inline('1/(1 + x^2)', 'x');
>> fun2(1.0)                  %Evaluarea functiei fun2 in x = 1.0
```

FUNCTIA ANONIMA (HANDLE FUNCTION) @

fun3 = @(***arg1,arg2,...,argn***)[***expr***]

fun3 - numele atribuit functiei
arg1 - sir de caractere ce contine argumentul 1 al functiei
arg2 - sir de caractere ce contine argumentul 2 al functiei
……
argn - sir de caractere ce contine argumentul n al functiei
expr - expresia functiei

```
>> fun3 = @(x)[1/(1 + x^2)];
>> fun3(1.0)                  %Evaluarea functiei fun3 in x = 1.0
```

Comanda `feval`
Comanda feval (functia evaluare) evalueaza valoarea unei functii pentru valori date (valoare data) ale argumentelor (a argumentului). Formatul comenzii:

```
[var1,var2,...,varm] = feval('function name',arg1,arg2,...,argn)
```

var1 - numele variabilei 1 ce defineste argumentul de iesire (outputul) 1
var2 - numele variabilei 2 ce defineste argumentul de iesire (outputul) 1
……
varm - numele variabilei *m* ce defineste argumentul de iesire (outputul) *m*
function name - sir de caractere ce contine numele functiei
arg1 - argumentul 1 al functiei
arg2 - argumentul 2 al functiei
……
argn - argumentul *n* al functiei

```
%Fisierul functie salvat cu numele "fun4.m"
function [y1, y2] = fun4(x1, x2, x3)
y1 = 1/(1 + x1^2 + x2^2 + x3^2);
y2 = (x1*x2*x3)/(1 + x1^2 + x2^2 + x3^2);
end

%Apelarea functiei in fereastra de comanda(Command Window)
>> [var1, var2] = feval('fun4',1,2,3)
>> var1             %Evaluarea functiei fun4 in (1.0, 2.0, 3.0)
>> var2             %Evaluarea functiei fun4 in (1.0, 2.0, 3.0)
```

11. OPERATORI RELATIONALI

Un operator relational compara doua numere (e.g. `5 < 8`), determinand veridicitatea declaratiei de comparare si atribuindu-i valoarile logice 1 daca declaratia este adevarata, respective valoarea logica 0 daca declaratia este falsa. is true or false.

Operatori relationali:

= =	egal
~ =	diferit de
<	mai mic strict
>	mai mare strict
<=	mai mic sau egal
>=	mai mare sau egal

- Daca doua numere sunt comparate, atunci rezultatul este un scalar 1 sau 0

- Daca sunt comparate doua tablouri (in mod necesar trebuie sa fie de acelasi tip), atunci compararea se face element cu element, iar rezultatul este un tablou logic de aceeasi dimensiune cu cele doua tablouri avand elementele 1 si 0 in conformitate cu rezultatele compararii element cu element.
- Daca un scalar este comparat cu un tablou, atunci scalarul este comparat cu fiecare element al tabloului respective, rezultatul este un tablou logic de aceeasi dimensiune cu tabloul respectiv avand elementele 1 si 0 in conformitate cu rezultatele compararii element cu element.

Rezultatul unei operatii relationale cu vectori, care este un vector cu componentele 1 sau 0, se numeste vector logic. Un vector logic poate fi folosit pentru a apela elementele unui vector, in sensul in care se extrag din acel vector elementele aflate pe pozitiile pe care vectorul logic are valoarea 1.

```
>>r = [8 12 9 4 23 19 10]
>>s = r<=10                    %s = [1 0 1 1 0 0 1]
>>t = r(s)                     %t = [r(1) r(3) r(4) r(7)]
>>w = r(r<=10)                 %similar cu linia de mai sus
```

Vectorii numerici si tablourile numerice avand ca elemente numerele 0 sau 1 sunt entitati distincte de vectorii logici si tablourile logice cu aceleasi dimensiuni. Vectorii numerici si tablourile numerice nu pot fi folosite la apelarea altor vectori si tablouri. Vectorii logici si tablourile logice pot fi folosite, insa, in operatii aritmetice. Cu prima ocazie cu care un vector sau tablou logic este folosit intr-o operatie aritmetica, vectorul sau tabloul logic devine vector sau tablou numeric.

Prioritatea operatiilor: Intr-o expresie matematica ce include operatii relationale si aritmetice, opratiile aritmetice (+, -, *, /, \) sunt prioritare fata de cele relationale. Operatorii relationali au prioritati egale, fiind executati de la stanga la dreapta. Intotdeauna pot fi folosite parantezele rotunde pentru a schimba ordinea de executare a operatiilor relationale si/sau aritmetice.

```
>>3 + 4 < 16/2          %se executa 3 + 4 si 16/2;
                        %se compara 7 cu 8 si se obtine 1 logic
>>3 + (4<16)/2          %se executa (4< 16) si se obtine 0 logic;
                        %se executa 0/2 si se obtine 0 numeric;
                        %se executa 3 + 0
```

12. OPERATORI LOGICI

&	SI logic
|	SAU logic
~	NEGARE logica

- Operatorii logici opereaza asupra numerelor. Un numar nenul este adevarat, in vreme ce zero este fals
- Operatorii logici (ca si operatorii relationali) pot fi folositi cu scalari si tablouri
- Operatorii logici SI si SAU pot opera atat asupra a doi scalari sau doua tablouri, cat si asupra unui scalar si a unui tablou. Daca ambii operanzii sunt scalari, atunci rezultatul este scalarul 0 sau 1. Daca ambii operanzi sunt tablouri (in mod necesar de aceeasi dimensiune), atunci operatia logica se face element cu element, iar rezultatul este un tablou de aceeasi dimensiune cu cea a tablourilor respective, avand elementele 0 sau 1 in functie de rezultatul operatiei logice executate element cu element. Daca un operand este scalar si celalalt operand este un tablou, atunci operatia logica se face intre scalar si fiecare element al tabloului, iar rezultatul este un

tablou de aceeasi dimensiune cu cea a tabloului operand, avand elementele 0 sau 1 in functie de rezultatul operatiei logice executate element cu element.

- Operatia de NEGARE logica are un singur operand. Daca este folosita cu un scalar, atunci rezultatul este 0 sau 1. Daca este folosita cu un tablou, atunci rezultatul este un tablot de aceeasi dimensiune cu cea a tabloului operand si avand elementele 1 in pozitiile in care tabloul initial are elemente nule, respectiv 0 in caz contrar.

```
>>3&7
>>a = 5|0
>>~25
>>t = 25*((12&0) + (~0) + (0|5))
>>u = [9 3 0 11 0 15], v=[2 0 13 -11 0 4]
>>u&v
>>w = u|v
>>~(u + v)
```

Prioritatea operatiilor: Operatorii aritmetici, relationali si logici pot fi combinati in expresii matematice. Daca o expresie contine o astfel de combinatie, atunci rezultatul ei depinde de ordinea de executare a operatiilor.

```
>>help precedence        %Informatii - prioritatea operatiilor
```

Prioritate	**Operatie**
1 (maxima)	Parentezele rotunde ( )
2	Ridicarea la putere (^)
3	Negare logica (~)
4	Inmultirea (*), impartirea (/ , \)
5	Adunarea (+), scaderea (-)
6	Operatorii relationali (>, <, >=, <=, ==, ~=)
7	SI logic (&)
8 (minima)	SAU logic (|)

```
>>a = -2; b = 5;
>>-5 < a < -1
>>-5<a & a<-1
>>~(b<7)
>>~b<7
>>~((b>=8)|(a<-1))
>>~(b>=8)|(a<-1)
>>x=[-1 1 1]
>>y=[1 2 -3]
>>x>0 & y>0
>>x>0 | y>0
```

FUNCTII LOGICE PREDEFINITE

Function	Description	Example
`xor(a,b)`	SAU exclusiv Returneaza 1 (adevarat) daca cel putin unul dintre operanzi este adevarat Returneaza 0 (fals) daca ambii operanzi sunt falsi.	`>>xor(7,0)` `>>xor(7,-5)`

`all(A)`	Returneaza 1 (adevarat) daca toate elemetele vectorului A sunt nenule (adevarat).
	Returneaza 0 (fals) daca cel putin un element este nul (fals).
	Daca A este o matrice, coloanele lui A sunt considerate drept vectori si rezultatul este un vector cu elementele 1 si 0.
		`>>A=[6 2 15 9 7 11]`
		`>>all(A)`
		`>>B=[6 2 15 9 0 11]`
		`>>all(B)`
		`>>C=A'*B`
		`>>all(C)`

`any(A)`	Returneaza 1 (adevarat) daca exista cel putin un element al lui A nenul (adevarat).
	Returneaza 0 (false) daca toate elementele lui A sunt nule (fals).
	Daca A este o matrice, coloanele lui A sunt considerate drept vectori si rezultatul este un vector cu elementele 1 si 0.
		`>>A=[6 0 15 0 0 11]`
		`>>any(A)`
		`>>B=[0 0 0 0 0 0]`
		`>>any(B)`

`find(A)`
 	
`find(A>d)`
	Daca A este un vector, atunci returneaza indicii elementelor nenule ale lui A.
	Daca A este un vector, atunci returneaza indicii elementelor nenule ale lui A mai mari ca *d* (se poate folosi orice operator relational).
		`>>A=[0 9 4 3 7 0 0 1]`
		`>>find(A)`
		`>>find(A>4)`

13. STRUCTURI DE CONTROL IN MATLAB

Exista patru structuri de control in MATLAB: declaratiile conditionale **`if`** si **`switch`**, respectiv ciclurile **`for`** si **`while`**.

DECLARATII CONDITIONALE

`if` conditional

```
if <conditia>
     <declaratii>
end
```

`<conditia>` expresie logica in MATLAB a carei valoare logica (1 sau 0) se calculeaza;
`<declaratii>` seturi de declaratii in MATLAB (atribuiri, operatii, etc) care se executa doar daca valoarea expresiei logice `<conditia>` este 1 (adevarat).

`if-else` conditional

```
if <conditia 1>
     <declaratii 1>
else
     <declaratii 2>
end
```

<conditia> expresie logica in MATLAB a carei valoare logica (1 sau 0) se calculeaza;
<declaratii 1> seturi de declaratii in MATLAB care se executa doar daca <conditia 1> este 1 (adevarat);
<declaratii 2> seturi de declaratii in MATLAB care se executa doar daca <conditia 1> este 0 (fals).

`if-elseif-else` conditional

```
if <conditia 1>
     <declaratii 1>
elseif <conditia 2>
     <declaratii 2>
else
     <declaratii 3>
end
```

<conditia 1>, <conditia 2> expresii logice in MATLAB ale caror valori logice (1 sau 0) se calculeaza;
<declaratii 1> seturi de declaratii in MATLAB care se executa doar daca <conditia 1> este 1 (adevarat);
<declaratii 2> seturi de declaratii in MATLAB care se executa doar daca <conditia 1> este 0 (fals) si <conditia 2> este 1 (adevarat);
<declaratii 3> seturi de declaratii in MATLAB care se executa doar daca <conditia 1> este 0 (fals) si <conditia 2> este 0 (fals).

CICLURI
Ciclul `for`

```
for var = expr
     <declaratii>
end
```

```
>> f(1) = 0;
>> f(2) = 1;
>> for k = 3:10
     f(k) = f(k-1) + f(k-2)
>> end
```

Se repeta setul de declaratii <declaratii> din ciclul **`for`** pentru valorile specificate ale indicelui de iteratie (`k`) a ciclului ce satisface relatia `k = 3:10` (`var = expr`).

Ciclul `while`

```
while <conditia>
     <declaratii>
end
```

```
>> f(1) = 0;
>> f(2) = 1;
>> k = 3;
>> while k <= 10
     f(k) = f(k-1) + f(k-2)
>> end
```

Se repeta setul de declaratii `<declaratii>` din ciclul **`while`** pentru valorile indicelui de iteratie (`k`) a ciclului ce respecta conditia logica `<conditia>` specificata (`k <= 10`).

Exercitiu:
Se considera functia de gradul doi $f:\mathbf{R}\to\mathbf{R}$, $f(x)=ax^2+bx+c$, unde $a,b,c\in\mathbf{R}$.
Sa se scrie o functie MATLAB care calculeaza solutiile ecuatiei de gradul doi $f(x)=0$ intr-un interval dat. Folositi numele functiei si argumentele urmatoare `[x1,x2] = sol(a,b,c).`

Exercitiu:
Sa se calculeze, folosind ciclul `for`, produsul primelor 10 numere naturale consecutive nenule.

Exercitiu:
Sa se calculeze, folosind ciclul `while`, suma maxima posibila a primelor numere naturale consecutive care este mai mica decat 1000.

14. REPREZENTARE GRAFICA

MATLAB are posibilitati grafice remarcabile si versatile, putand fi generate diferite tipuri de grafice cu relative ususrinta. In MATLAB, graficele pot fi generate fie in fereastra de comenzi (Command Window), fie in fisierele de tip M, in vreme ce figurile deja existente pot fi modificate si adnotate interactive folosind editorul de figure (Plot Editor).

REPREZENTARE GRAFICA IN 2D

Comanda `plot`

`plot(x,y,'LineSpecifiers', 'PropertyName',PropertyValue)`

`x` – vector de dimensiune *n*, **`x`**=[$x_1,x_2,\ldots,x_n$], continand argumentul functiei *f* in intervalul de interes

`y` – vector de aceeasi dimensiune *n* cu **`x`**, **`y`**=[$y_1,y_2,\ldots,y_n$], continand valoarea functiei *f*, i.e. $y_i=f(x_i)$

```
>> x = [1 2 3 5 7 7.5 8 10]
>> y = [2 6.5 7 7 5.5 4 6 8]
>> plot(x,y)
```

`Line Specifiers` – specificatori de linie, i.e. stilul si culoarea liniei, precum si (optional) tipul markerului

Stilul liniei	Specificator
Linie continua	-
Linie intrerupta	--
Linie punctata	:
Linie mixta	-.

Culoarea liniei	Specificator
Rosu	r
Verde	g
Albastru	b

Cyan	c
Magenta	m
Galben	y
Negru	b
Alb	w

Tipul markerului	Specificator
Semnul plus	+
Cerc	o
Asterisc	*
Punct	.
Patrat	s
Romb	d
Stea cu 5 colturi	p
Stea cu 6 colturi	h

```
>> plot(x,y,'r')
>> figure(1)
>> plot(x,y,'--y')
>> plot(x,y,'g:d')
```

`Property Name & Property Values` – numele si valoarea proprietatilor folosite pentru definirea grosimii liniei, marimii si culorii markerului (optional)

Numele proprietatii	Descriere	Valorile proprietatii
`LineWidth` (`linewidth`)	Specifica grosimea liniei	Un numar in unitati de punct (0.5 = valoarea de default)
`MarkerSize` (`markersize`)	Specifica marimea markerului	Un numar in unitati de punct (0.5 = valoarea de default)
`MarkerEdgeColor` (`markeredgecolor`)	Specifica culoarea markerului sau culoarea muchiilor markerului cand interiorul markerului este colorat	Specificatori de culoare introdusi ca sir
`MarkerfaceColor` (`markerfacecolor`)	Specifica culoarea fetelor markerelor cand interiorul markerului este colorat	Specificatori de culoare introdusi ca sir

```
>> figure(2)
>> plot(x,y,'--r*','linewidth',2,'markersize',12)
>> xlabel('x')
>> ylabel('y')
```

Pentru **reprezentarea mai multor grafice pe o aceeaşi pagină**, se foloseşte comanda **`subplot(m,n,p)`** care imparte fereastra de reprezentare grafica in $m \times n$ subgrafice, aranjate ca elementele unei matrici cu *m* linii şi *n* coloane, *p* fiind subgraficul curent.

```
>> x = [1 2 3 5 7 7.5 8 10]
>> y1 = [2 6.5 7 7 5.5 4 6 8]
>> y2 = [-2 0.5 0.5 2 2.5 1 4 5]
>> y3 = [1 2 3 5 7 7.5 8 10]
>> y4 = (x - 4).^2
>> subplot(2,2,1)
>> plot(x,y1)
```

```
>> subplot(2,2,2)
>> plot(x,y2)
>> subplot(2,2,3)
>> plot(x,y3)
>> subplot(2,2,4)
>> plot(x,y4)
```

Reprezentarea a doua sau mai multe functii/date pe acelasi grafic se realizeaza prin gruparea perechilor corespunzatoare de vectori in cadrul aceleiasi comenzi `plot`, i.e.

`plot(`*`x1`*`,`*`y1`*`,'LineSpecifiers1', 'PropertyName1',PropertyValue1,`
`  `*`x2`*`,`*`y2`*`,'LineSpecifiers2', 'PropertyName2',PropertyValue2, …,)`

```
>> plot(x,y1,'-r*',x,y2,'--go',x,y3,':bd')
```

Exercitiu:
Reprezentaţi grafic, pe aceeasi pagina si, respectiv, pe acelasi grafic, funcţia $f_1(x) = x^2 + 4\sin(2x) - 1, \quad x \in [-\pi, \pi]$, si derivatele de ordinul intai si al doilea ale acesteia.

Exercitiu:
Reprezentaţi grafic funcţia $f_2(x) = \dfrac{x^2 - x + 1}{x^2 + x + 1}, \quad x \in [-10,10]$.

Exercitiu:
Reprezentaţi grafic, pe aceeasi pagina si, respectiv, pe acelasi grafic, functia $f_3(x) = 3x\sin(x) - 2x$, $x \in [0,2\pi]$, si derivatele de ordinul intai si al doilea ale acesteia.

Exercitiu:
Reprezentaţi grafic, pe aceeasi pagina si, respectiv, pe acelasi grafic, funcţiile definite prin $f_1(x) = x^2 + 4\sin(2x) - 1$ si $f_2(x) = \dfrac{x^2 - x + 1}{x^2 + x + 1}, x \in [-5,5]$.

Exercitiu:
Sa se defineasca, folosind structura **`if-elseif-else`**, urmatoarea functie:

$$f(x) = \begin{cases} x^2, & x \in (-2,0) \\ x^3, & x \in [0,3) \\ x^2 + 18, & x \in [3,4) \\ 0, & \text{altfel} \end{cases}$$

Sa se construiasca graficul acestei functii pe intervalul [-5,5].

Comanda `fplot`

`fplot('`*`fun`*`', limits, 'LineSpecifiers')`

`fun` - numele atribuit functiei ce se vrea reprezentata graphic; aceasta poate fi introdusa direct in comanda `fplot` ca un sir de caractere

limits - vector cu doua elemente care specifica domeniul lui x `[xmin, xmax]` sau vector cu 4 elemente care specifica domeniul lui x si limitele axei y `[xmin, xmax, ymin, ymax]`

LineSpecifiers – specificatori de linie, i.e. stilul si culoarea liniei si (optional) tipul markerului

```
>> fplot('x^2+4*sin(2*x)-1',[-3,3])
>> figure
>> fplot('x^2+4*sin(2*x)-1',[-3,3,-10,10],'--go')
```

Comenzile `xlabel` si `ylabel`

```
xlabel('text axa Ox',PropertyName,PropertyValue)
ylabel('text axa Oy',PropertyName,PropertyValue)
```

Comanda `title`

```
title('text titlul graficului',PropertyName,PropertyValue)
```

Comanda `text`

```
text(x,y,'text',PropertyName,PropertyValue)
gtext('text',PropertyName,PropertyValue)
```

(x,y) – coordonatele pozitiei primului caracter din textul inserat in figura
text – textul inserat in figura ca sir de caractere

GRAFICE 2D ANIMATE

```
>> t = 0:.0001:1;
>> y = sin(4*pi*t);
>> comet(t,y)
```

REPREZENTARE GRAFICA IN 3D

15. APLICATII IN ANALIZA NUMERICA

ZEROURILE UNEI FUNCTII

```
x = fzero('fun',x0)
```

'fun' – functia considerata scrisa ca un sir de caractere
***x* –** zeroul functiei **'fun'**
***x0* –** valoare initiala a zeroului functiei **'fun'**

```
x = fzero('fun',[a,b])
```

'fun' – functia considerata scrisa ca un sir de caractere
***x* –** zeroul functiei **'fun'**
[*a*,*b*] – interval in care functia **'fun'** admite o radacina unica

PUNCTE DE EXTREM LOCAL ALE UNEI FUNCTII

```
x = fminbnd('fun',x1,x2)
```

'fun' - functia considerata scrisa ca un sir de caractere sau nume de fisier functie sau functie inline
[x1,x2] – intervalul in care se cauta minimul local al functiei **'fun'**
x - punctul de minim local al functiei **'fun'**

```
[x,fval] = fminbnd('fun',x1,x2)
```

'fun' – functia considerata scrisa ca un sir de caractere sau nume de fisier functie sau functie inline
[x1,x2] – intervalul in care se cauta minimul local al functiei **'fun'**
x – punctul de minim local al functiei **'fun'**
fval – valoarea functiei **'fun'** in punctul de minim local

INTEGRAREA NUMERICA A UNEI FUNCTII
Metoda adaptiva de integrare a lui Simpson

```
q = quad('fun',a,b)
```

'fun' - functia considerata scrisa ca un sir de caractere sau nume de fisier functie sau functie inline
q – valoarea numerica a integralei
[a,b] – intervalul pe care se integreaza functia **'fun'**

Metoda adaptiva de integrare a lui Lobatto

```
q = quadl('fun',a,b)
```

'fun' - functia considerata scrisa ca un sir de caractere sau nume de fisier functie sau functie inline
q – valoarea numerica a integralei
[a,b] – intervalul pe care se integreaza functia **'fun'**

Metoda trapezelor

```
q = trapz(x,y)
```

x,y - vectori continand coordonatele (x, y=f(x)) ale functiei f(x) considerate
q – valoarea numerica a integralei

Exercitiu:
Fie functia $f(x) = x^3 - 7x^2 + 2x + 4$.
1. Definiti fisierul functie **f0** care descrie functia de mai sus.
2. Reprezentati grafic functia *f* definita pe intervalul [-3, 7] apeland functia definita la 1.
3. Aflati radacinile ecuatiei $f(x) = 0$ folosind cele doua definitii ale comenzii MATLAB **fzero** date mai sus.
4. Determinati punctele de extrem ale functiei *f* folosind comanda MATLAB **fminbnd.**
5. Calculati $\int_{-3}^{7} f(x)dx$ folosind cele trei metode numerice de integrare de mai sus.

REZOLVAREA NUMERICA A PROBLEMEI CAUCHY PENTRU ECUATII DIFERENTIALE ORDINARE

Integrarea numerica a unei ecuatii diferentiale ordinare de ordinul I

Cosideram urmatoarea problema de tip Cauchy pt ecuatia diferentiala ordinara de ordinul I:

$$\begin{cases} y'(t) \equiv \dfrac{dy}{dt} = f(t,y), & t \in [t_0, t_f] \\ y(t_0) = y_0 \end{cases}.$$

MATLAB dispune de mai multe subrutine care permit rezolvarea numerica a problemelor de tipul celei de mai sus. Sintaxa generala pentru apelarea unei astfle de subrutine este urmatoarea:

```
[t,y] = solver_name('ODEfun',tspan,Y0,Tol)
[t,y] = solver_name(@ODEfun,tspan,Y0,Tol)
```

solver_name	–	numele solverului (metodei numerice) utilizat, e.g. **ode45** sau **ode23** (algoritm de tip Runge-Kutta de ordinul 4)
'ODEfun'	-	functie data, scrisa ca un sir de caractere sau nume de fisier functie care calculeaza `f(t,y)` pentru valori date ale lui `t` si `y`
tspan	–	vector ce specifica intervalul solutiei `[t0, tf]`
Y0	–	conditia initiala pentru `y`
[t,y]	-	outputul: `t,y` sunt vectori coloana ce contin solutia problemei
Tol	-	toleranta metodei numerice (eroarea)

Exercitiu:

Fie functia $f(t,y) = t - 2y$.

1. Definiti fisierul functie **fun** care descrie functia de mai sus.
2. Calculati solutia problemei $y'(t) = f(t,y)$, $t \in [0,1]$, cu datele initiale $y(0) = 1$, folosind subrutina **ode45.**
3. Construiti graficul solutiei numerice obtinute.
4. Calculati erorile obtinute prin folosirea subrutinei **ode45**. Solutia exacta a problemei Cauchy este $y(t) = \frac{1}{4}\left(2t - 1 + 5e^{-2t}\right)$.

Exercitiu:

Fie functia $f(t,y) = (t^2 - 2t + 3)/y^2$.

1. Definiti fisierul functie **fun** care descrie functia de mai sus.
2. Calculati solutia problemei $y'(t) = f(t,y)$, $t \in [0.5,3]$, cu datele initiale $y(0.5) = 2$, folosind subrutina **ode45.**
3. Construiti graficul solutiei numerice obtinute.
4. Calculati erorile obtinute prin folosirea subrutinei **ode45**. Solutia exacta a problemei Cauchy este $y(t) = (t^3 - 3t^2 + 9t + 33/8)^{1/3}$.

Integrarea numerica a unui sistem de ecuatii diferentiale ordinare de ordinul I
Cosideram urmatoarea problema de tip Cauchy pt ecuatia diferentiala ordinara:

$$\begin{cases} y_1'(t) \equiv \dfrac{dy_1}{dt} = f(t, y_1, y_2), & t \in [t_0, t_f] \\ y_2'(t) \equiv \dfrac{dy_2}{dt} = f(t, y_1, y_2), & t \in [t_0, t_f]. \\ y_1(t_0) = y_1^{(0)} \\ y_2(t_0) = y_2^{(0)} \end{cases}$$

MATLAB dispune de mai multe subrutine care permit rezolvarea numerica a problemelor de tipul celei de mai sus. Sintaxa generala pentru apelarea unei astfle de subrutine este urmatoarea:

```
[t,y] = solver_name('ODEsys',tspan, ,[Y1;Y2],Tol)
```

solver_name – numele solverului (metodei numerice) utilizat, e.g. **ode45** sau **ode23** (algoritm de tip Runge-Kutta de ordinul 4)
'ODEsys' - nume de fisier functie care calculeaza ($f_1(t, y_1, y_2)$, $f_2(t, y_1, y_2)$) pentru valori date ale lui t, y_1 si y_2
tspan – vector ce specifica intervalul solutiei $[t_0, t_f]$
Y0 – conditia initiala pentru (y_1, y_2)
[t,y] - outputul: ***t*** si **y=[y(:,1); y(:,2)]** sunt vectori coloana ce contin solutia problemei
Tol - toleranta metodei numerice (eroarea)

Exercitiu:
Fie functiile $f_1(t, y_1, y_2) = y_2$ si $f_2(t, y_1, y_2) = 4y_1$.

1. Definiti fisierul functie **sys** care descrie functia de mai sus.
2. Calculati solutia problemei $y_1'(t) = f_1(t, y_1, y_2)$, $y_2'(t) = f_2(t, y_1, y_2)$, $t \in [0,1]$, cu datele initiale $y_1(0) = 3$ si $y_2(0) = -2$, folosind subrutina **ode45**.
3. Construiti graficul solutiei numerice obtinute.
4. Determinati solutia exacta a sistemului de ecuatii diferentiale de ordinul I cu date initiale. Calculati erorile obtinute prin folosirea subrutinei **ode45.**

BIBLIOGRAFIE

[1] *Using MATLAB v6*, The MathWorks Inc., 2002.
[2] Desmond J. Higham, Nicholas J. Higham, *Matlab Guide*, Philadelphia: Society for Industrial and Applied Mathematics, 2005.
[3] Timothy A. Davis, Kermit Sigmon, *Matlab Primer*, London; Boca Raton, Fl.: Chapman & Hall/CRC, 2005.
[4] Amos Gilat, *MATLAB: An Introduction with Applications*, Hoboken, NJ: Wiley, 2004.
[5] Gerald W. Recktenwald, *Numerical Methods with MATLAB: Implementations and Applications*, New Jersey: Prentice Hall, 2000.