

# 1. INTRODUCERE ÎN MEDIUL DE PROGRAMARE MATLAB

## 1.1 Organizarea calculatorului

Aceasta cuprinde toate comenzile și funcțiile unui sistem de calcul, precum și structura de comunicații necesară interacționării, la care se adaugă software-ul de sistem necesar funcționării.

Procesorul este partea centrală a oricărui sistem de calcul. El determină în cea mai mare parte reprezentarea informației, respectiv codificarea datelor, instrucțiunilor și formatele acestora pentru transfer și memorare. Funcțiile procesorului determină performanțele utilizabile pentru programarea de sistem și de aplicații.

Cresterea performanțelor sistemului de calcul se poate face prin posibilitatea configurării procesorului cu unitați de memorie și echipamente de intrare/ieșire împreună cu căile de transfer care le conectează. Sistemele de mare capacitate sunt realizate în mod frecvent ca sisteme multiprocesor sau ansambluri de calculatoare în rețele de calculatoare.

Exista o mare varietate de softuri de sistem necesare sistemelor de calcul, numite sisteme de operare pentru operații diverse ale calculatoarelor.

## 1.2 Reprezentarea informației

În orice sistem de calcul prelucrarea informației se face prin intermediul datelor, reprezentarea lor fiind exclusiv sub formă binară.

Cea mai mica unitate informațională este bitul (BInary DigiT - cifra binară), ce poate avea două valori (stări) notate cu 0 și 1. Din punct de vedere tehnic, acestea se reprezintă în mod diferit (nivel de tensiune, încărcare de condensator, direcție de magnetizare).

Codificarea datelor presupune unirea biților în cuvinte de cod, al căror număr de biți corespunde formatelor de date necesare pentru prelucrare, transfer și memorare.

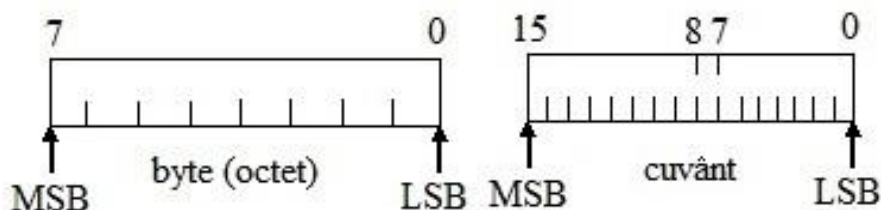
Formatele standard sunt byte (8 biți) sau octet și multipli pari ai acestuia. Noțiunea de cuvânt (16 biți) a fost utilizată inițial pentru dimensiunea elementară de prelucrare și memorare corespunzătoare unui calculator.

Referitor la mărimile datelor, se folosesc următoarele notații:

| Denumire  | Simbol | Mărime binară | Număr de bytes                    | Egal cu     |
|-----------|--------|---------------|-----------------------------------|-------------|
| Kilobyte  | KB     | $2^{10}$      | 1.024                             | 1.024 bytes |
| Megabyte  | MB     | $2^{20}$      | 1.048.576                         | 1.024 KB    |
| Gigabyte  | GB     | $2^{30}$      | 1.073.741.824                     | 1.024 MB    |
| Terabyte  | TB     | $2^{40}$      | 1.099.511.627.776                 | 1.024 GB    |
| Petabyte  | PB     | $2^{50}$      | 1.125.899.906.842.624             | 1.024 TB    |
| Exabyte   | EB     | $2^{60}$      | 1.152.921.504.606.846.976         | 1.024 PB    |
| Zettabyte | ZB     | $2^{70}$      | 1.180.591.620.717.411.303.424     | 1.024 EB    |
| Yottabyte | YB     | $2^{80}$      | 1.208.925.819.614.629.174.706.176 | 1.024 ZB    |

Alte formate de date sunt: bitul individual, semibyte (4 biți), câmpul de biți (are număr variabil de biți).

În reprezentările grafice ale formatelor de date se numerotează biții începând cu zero, de la dreapta la stânga și li se atribuie ponderi crescătoare în vederea reprezentării lor binare.

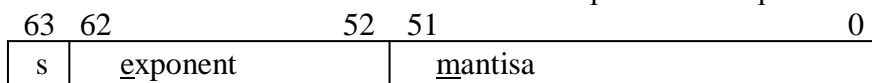


LSB = Least Significant Bit (bitul cel mai puțin semnificativ)

MSB = Most Significant Bit (bitul cel mai semnificativ)

Un calculator tratează secvențele de biți de o lungime dată (8, 16, 32, 64,...) în funcție de abilitatea sa de a trata simultan 8, 16, 32, 64,... biți.

Programul MATLAB folosește codificarea pe 64 de biți pentru reprezentarea tuturor numerelor în virgulă mobilă și dublă precizie (double). Acest cuvânt are o structură ce conține 1 bit pentru semn, 11 biți pentru exponent (notat e) și 52 de biți pentru mantisa m. Codificarea este conform standardului IEEE754 pentru dublă precizie.



Bitul de semn are valoarea 0 pentru numere pozitive și 1 pentru numere negative.

În MATLAB se pot contrui și date în simplă precizie (single), în concordanță cu standardul IEEE754 pentru simplă precizie. Orice valoare memorată în simplă precizie necesită 32 de biți (1 bit pentru semn, 8 biți pentru exponent și 23 de biți pentru mantisă):

|    |          |    |    |         |
|----|----------|----|----|---------|
| 31 | 30       | 23 | 22 | 0       |
| s  | exponent |    |    | mantisă |

Evident, variabilele memorate pe 32 de biți nu sunt atât de precise ca cele memorate pe 64 de biți. Folosirea dublei precizii permite memorarea de variabile mai mari decât aproximativ  $3.4 \times 10^{38}$  și mai mici decât  $-3.4 \times 10^{38}$ . Pentru variabile cuprinse între aceste limite se poate folosi simpla sau dubla precizie.

Crearea unei variabile în dublă precizie nu necesită nici o instrucțiune suplimentară, ci doar definirea ei:

```
>> x=32.764;
```

Folosind funcția „whos” putem vedea cum MATLAB a creat o matrice de tipul (1 x 1) de tip „double” pentru valoarea lui x:

```
>> whos x <Enter>
```

```
Name      Size      Bytes    Class
x          1x1         8      double
```

Comanda „isfloat” permite verificarea faptului că variabila x este un număr în virgulă mobilă și returnează valoarea logică 1 (adevărat dacă variabila este un număr în virgulă mobilă) și 0 (fals) în caz contrar:

```
>> isfloat(x)
```

```
ans=
```

```
1
```

Anumite obiecte frecvent utilizate cu nevoie de codificare. Este vorba de caracterele alfanumerice (cifre, litere, semne de punctuație, caractere speciale) pentru care se cunosc mai multe sisteme de codificare: BCD (Binary Coded Decimal), EBCDIC (Extended Binary Coded Decimal Interchange Code) si ASCII (American Standard Code for Information Interchange).

### 1.3. Codul ASCII


În memoria calculatorului toate obiectele sunt codificate binar. Avem nevoie de minim 26 de litere și 10 cifre, deci 36 de caractere. Dacă dorim să recunoaștem și literele mari și cele mici, mai trebuie încă 26 de caractere. Folosind și caracterele speciale (+, =, \*, %, ...) avem cel puțin 64 de caractere. A fost dezvoltat astfel codul ASCII, un cod pe 7 biți ce permite codificarea a 128 de caractere. El este standardizat ISO. Datorită formatului de octet (byte), un al optulea bit este adăugat, cu valoare prestabilită fie ca bit de paritate, fie ca extensie de cod (folosit pentru detectarea erorilor).

### 1.4. Limbaje de programare

Un limbaj de programare este un limbaj artificial dezvoltat în scopul transmiterii instrucțiunilor calculatorului. Limbajul de programare poate fi folosit pentru a dezvolta programe care controlează funcționarea unui calculator, ele constând în cuvinte, fraze și reguli sintactice.

Începând cu anul 1960 s-a dezvoltat majoritatea limbajelor de programare pe care le folosim și în prezent (FORTRAN, COBOL, C). Primul limbaj destinat programării orientate pe obiecte a fost Simula. Actualmente, există mai multe limbaje de programare utilizate frecvent, fiecare fiind concentrat pe o anumită direcție, destinat rezolvării unor probleme din domenii specifice.

Din punct de vedere al nivelului limbajelor de programare, putem face următoarea clasificare:

1. Limbaje de nivel înalt: 
  - Ada
  - Modula-2
  - Pascal (după matematicianul Blaise Pascal)
  - COBOL (COMMON BUSINESS ORIENTED LANGUAGE)
  - FORTRAN (FORMULA TRANSLATING SYSTEM)
  - BASIC (BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE)
  - LISP (LIST PROCESSING)



aplicații în inginerie. Impactul acestui program a fost simțit în primul rând în fizică, inginerie și matematică. Se folosește cu precădere în cercetare, pentru prototipuri sau interfețe.

**MATHCAD** - prescurtarea de la MATHematical Computer Assisted Design, produs de firma Mathsoft Inc., SUA ([www.mathcad.com](http://www.mathcad.com)). Pune la dispoziția utilizatorilor un pachet bogat specializat în calcule matematice numerice sau simbolice. Are unele funcții predefinite (trigonometrice, exponențiale, logaritmice, hiperbolice, speciale) cu ajutorul cărora se pot executa multe operații matematice. O caracteristică importantă este faptul că în documentele Mathcad se folosesc notațiile matematice obișnuite; prin reprezentarea pe ecran a relațiilor editate, cât și prin tipărirea la imprimantă a documentelor, în forma obținută pe ecran. Este compatibil cu sistemul de operare Windows, maniera de lucru fiind similară cu programele din pachetul Office.

**MAPLE** - include multiple facilități pentru efectuarea calculelor numerice, simbolice, reprezentări grafice și chiar de programare într-un limbaj avansat. Este disponibil sub Windows, DOS, UNIX, Sun, Macintosh. A fost dezvoltat în 1980 de compania Maplesoft.

**DERIVE** - permite atât calculul numeric cât și cel simbolic. A fost dezvoltat de Texas Instruments.

**MATLAB** - MATrix LABoratory - un soft destinat calculului numeric, statistic și reprezentărilor grafice în domeniul ingineresc. Acesta are la bază calculul matricial, matricea fiind elementul cu care MATLAB operează.

Este un mediu ușor de învățat și utilizat, problematica și soluțiile propuse fiind exprimate într-un mod natural, nefiind necesară utilizarea unei scrieri de tip algoritmic specifică limbajelor de programare tradiționale. Alocarea memoriei este dinamică (adică fiecărui vector sau matrici  $i$  se alocă un număr de locații de memorie în RAM corespunzător dimensiunilor efective și tipului de dată utilizată; nu ca și limbajele de programare unde vectorii și matricile au alocate dimensiuni de numere corespunzătoare valorilor maxime –

aceasta în cazul în care nu sunt folosiți pointerii).

Sintaxa MATLAB este asemănătoare cu a limbajelor științifice de programare, având însă următoarele caracteristici:

- mai puține cuvinte cheie (if, else, for, end, ...)
- identificatorii (simboluri alese de programator pentru definirea variabilelor, constantelor, funcțiilor) în MATLAB se definesc de utilizator, doar o parte din ei fiind predefiniți (sau pot fi și ei redefiniți)
- numerele sunt aceleași ca la limbajele științifice, deci în mai multe formate de reprezentare (short, long, hex, short e, long e, rat)
- face distincție între literele mari și cele mici (limbajele de programare nu fac această distincție)
- șirurile de caractere se delimitează cu apostrof (ca în C, C++);
- linia de comandă sau instrucțiunea trebuie terminată cu simbolul punct-și-virgulă (;), astfel rezultatul obținut în urma calculelor nu este afișat.

Pentru scrierea programelor se poate folosi și un editor extern, de regulă NOTEPAD. În MATLAB pe lângă fișierele program (script) mai avem și fișere funcție, similare unit-urilor din Pascal sau header-elor din C. Este un interpretor ce poate executa atât comenzi directe, cât și fișiere program.

## **1.6 Particularitățile mediului de programare MATLAB**

În a doua jumătate a secolului XX, înainte de apariția calculatoarelor personale, calculele de mare complexitate erau efectuate folosind coduri dezvoltate în Fortran. Cu cât numărul subsistemelor dezvoltate pentru probleme specifice de calcul a fost mai mare, acestea au fost transpuse în pachete de programe distribuite liber. MATLAB a fost creat dintr-un astfel de pachet, numit LINPACK, ce conținea un grup de programe pentru lucrul cu matrici și algebră liniară. Dezvoltatorul inițial al programului, profesorul Cleve Moler de la Universitatea din New Mexico, a fondat compania Mathworks Inc., cu scopul de a îmbunătăți și comercializa produsul rezultat.

Astfel, a rezultat un pachet de programe performant, ce integrează calculul numeric, grafica și programarea într-un mediu de

programare prietenos cu utilizatorul.

Acest soft include:

- calcule matematice
- dezvoltarea algoritmilor de programare
- achiziții de date
- analiza datelor, exploatarea rezultatelor și reprezentarea lor grafică
- grafice științifice și ingineresti
- dezvoltare de aplicații, inclusiv interfețe grafice cu utilizatorul (GUI = Graphical User Interface)
- interfețe cu limbajele de programare C, C++ și Fortran.

Sistemul Matlab include cinci părți principale:

1. Mediul de dezvoltare: conține un set de unelte care facilitează folosirea funcțiilor și fișierelor MATLAB, mare parte din acestea fiind interfețe grafice utilizator. Includ fereastra principală MATLAB (MATLAB Desktop), fereastra de comenzi (Command Window), fereastra cu istoricul comenzilor introduse (Command History), un „editor” și un „debugger”, browser-ele de tip Help, Workspace, Files, Search Path.
2. Biblioteca de funcții matematice: conține foarte mulți algoritmi pentru calculul funcțiilor elementare matematice și trigonometrice, funcții în domeniul complex, până la funcții destinate calculului matricial, valorilor proprii și vectorilor proprii, funcții Bessel, transformata Fourier rapidă.
3. Limbajul: toate variabilele sunt considerate vectori sau matrici; include comenzi de programare, controlul buclor de calcul, funcții, structuri de date, comenzi intrare/ieșire, inclusiv posibilitatea programării orientate pe obiecte. Se poate programa „sumar” sau „în detaliu”, rezultând programe mari și complicate.
4. Grafice: se pot reprezenta vectori și matrici în grafice, grafice bi- și tridimensionale, procesare de imagini, animații, posibilitatea creării unor interfețe grafice pentru aplicațiile MATLAB.
5. Interfețe externe: permit dezvoltarea unor programe în C și FORTRAN care să ruleze în MATLAB.



## 1.7 Lansarea și ieșirea din MATLAB

Pornirea MATLAB se poate face în mai multe feluri:

- dublu-clic pe iconița MATLAB de pe Desktop
- din meniul „START” de pe Desktop se face clic pe „MATLAB”
- din meniul „START” se face clic pe „Run”, se tastează „MATLAB” și clic pe „OK”

Când ați lansat MATLAB, următoarea fereastră principală apare pe Desktop, ce conține următoarele interfețe grafice:

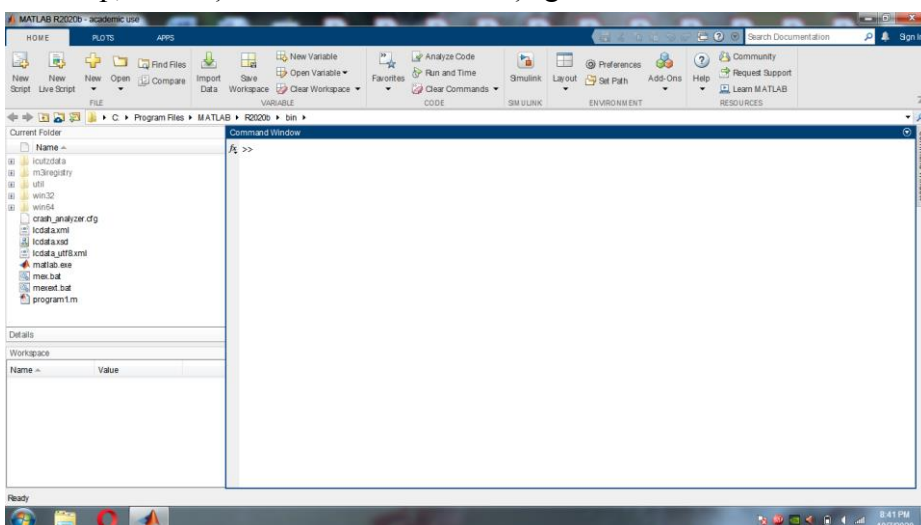


Figura 1.1 Ferestrele programului MATLAB

**„Command Window”**- (spațiul de lucru, zona de comenzi) în care se scriu efectiv comenzile (instrucțiunile) pe care programul le va executa

**„Current Folder”** - permite controlul fișerelor MATLAB și a celor asociate

**„Help”** - activarea ei permite vizualizarea și căutarea în documentația MATLAB

**„Edit”** și **„Debugg”** - activarea lor permite editarea, crearea și

depanarea fișierelor ce conțin funcții MATLAB

„**Desktop**” - activarea ei permite mutarea, minimizarea, redimensionarea sau închiderea diverselor ferestre.

Trebuie menționat că fereastra grafică se deschide numai dacă în zona de comenzi au fost activate comenzi grafice.

Funcții utilizate în zona de comenzi și efectul lor:

- **home** - mută cursorul în prima linie-prima coloană;
- **MORE** - controlează vizualizarea paginii de lucru în zona de comenzi.
  - **MORE OFF** - dezactivează paginarea în zona de comenzi
  - **MORE ON** - activează paginarea în zona de comenzi
  - **MORE(N)** - specifică mărimea paginii vizualizate în zona de comenzi de mărime N linii
- **clc** – are ca efect ștergerea paginii în zona de comenzi
- **clear (clear all)** - șterge toate variabilele definite din memorie
- **format loose** - are ca efect afișarea liniilor în mod distanțat (cu linii libere)
- **format compact** - are ca efect eliminarea liniilor libere, afișând datele într-un format strâns. Se pot vizualiza mai multe informații pe ecran.

Trecerea de la un format la altul se poate face și selectând în baza de meniuri: „**File>Preferences>Command Window>Numeric display: compact (loose)**”.

De asemenea se poate selecta și tipul de reprezentare numerică a datelor folosite. Astfel, în MATLAB sunt posibile reprezentări în următoarele formate:

```
>>b=3/26;
```

```
>>format short, b
```

```
b=
```

```
0.1154 % 4 cifre dupa virgula
```

```
>>format long, b
```

```
b=
```

```
0.115384615384615 % 15 cifre dupa virgula in dubla precizie sau
```

```
>>% 7 cifre in simpla precizie
```

```

>>format short e, b
b=
    1.1538e-001 % 4 cifre dupa virgula plus exponentul litera „e”
>>% reprezinta puterile lui 10, practic „e-001” reprezinta  $10^{-1}$ 
>>format long e, b
b=
    1.153846153846154e-001 % 15 cifre dupa virgula plus
>>% exponentul in dubla precizie si 7 cifre plus
>>% exponentul in simpla precizie
>>format short g, b
b=
    0.11538 % 5 cifre dupa virgula
>>format long g, b
b=
    0.115384615384615 % 15 cifre dupa virgula in dubla precizie sau
>>% 7 cifre in simpla precizie
>> format short eng, b
b=
    115.3846e-003 % 4 cifre dupa virgula plus exponentul care este
>>% multiplu de 3
>>format long eng, b
b=
    115.384615384615e-003 % 12 cifre dupa virgula plus exponentul
>>% care este multiplu de 3
>>format hex, b
b=
    3fbd89d89d89d89e % format hexazecimal
>> format rat, b
b=
    3/26 % reprezentare sub forma de fractie
>>format +, b
b=
    + % semnul „+” pentru elementul pozitiv
>>format bank, b
b=
    0.12 % 2 cifre dupa virgula

```

## 1.8 Constante și variabile speciale în MATLAB

Acestea au un caracter global în orice fișier MATLAB, ele nu pot fi declarate. Variabile implicite:

- ans – este o variabilă creată automat și conține rezultatul unui calcul generat de evaluarea unei expresii, când expresia nu are asociat un nume
- eps – cel mai mic număr care adăugat lui 1 generează un număr mai mare ca acesta. Valoarea sa implicită este  $\text{eps} = 2.220446049250313\text{e-}016$ . Se mai poate defini și ca fiind eroarea relativă atunci când se efectuează calcule în virgulă mobilă. Dacă este nevoie de o precizie mai redusă de calcul, se poate modifica valoarea implicită, se efectuează calcule și la final se revine la valoarea implicită. Exemplu: o eroare relativă de  $2 \cdot 10^{-5}$

```
>>eps = 2.e-05
```

```
eps=
```

```
2.000000000000000e-005
```

```
>> % instructiuni de calcul
```

```
>> % revenire la valoarea implicita
```

```
>>clear eps
```

- pi – raportul între circumferința cercului și diametrul său. Are valoarea  $\text{pi}=3.141592653589793$ .
- i,j –  $i=j=\sqrt{-1}$ , sunt unitățile imaginare folosite în declararea numerelor complexe de forma:  $z=2+3 \cdot i$  sau  $w=-2+7 \cdot j$ . Se poate folosi ca unitate imaginară orice altă unitate dacă este declarată în prealabil, de exemplu:  $t=\sqrt{-1}$  și putem scrie  $z=2+3 \cdot t$
- inf – variabilă pentru a reprezenta rezultatul împărțirii lui  $1/0$ , adică infinit ( $\infty$ );
- NaN – „Not a Number” atunci când avem  $0/0$  sau  $\text{inf}/\text{inf}$ ;
- realmax – cea mai mare valoare pozitivă ce poate fi folosită în calcule:  $\text{realmax}=1.7976931348623163\text{e}+308$
- realmin – cea mai mică valoare pozitivă ce poate fi folosită în calcule:  $\text{realmin}=2.22504385850721\text{e-}308$
- NARGIN – în interiorul unei funcții definite de utilizator,

NARGIN returnează numărul argumentelor de intrare folosite la apelarea funcției

- NARGOUT – în interiorul unei funcții definite de utilizator, NARGOUT returnează numărul argumentelor de ieșire folosite la apelarea funcției
- VARARGOUT – listă de argumente de ieșire cu lungime variabilă; permite folosirea oricărui număr de argumente de ieșire la o funcție. VARARGOUT nu este inițializată la apelarea funcției
- flops – folosirea ei are ca efect afișarea numărului de operații în virgulă mobilă efectuate de calculator. Începând cu MATLAB 7.0 această comandă nu mai este utilizată
- computer – tipul calculatorului;
- version – versiunea programului MATLAB;

**Observație:** orice variabilă poate fi ștearsă din memoria calculatorului folosind comanda „clear”. În cazul variabilelor speciale, utilizarea altor valori pentru ele în diverse programe sau părți de program implică faptul că atunci când acestea sunt șterse se revine la valorile implicite.

În cazul variabilelor definite într-un fișier MATLAB, ștergerea lor este permanentă. De exemplu:

```
>>a=5; b=2; % 2 variabile definite
>>c=a+b
c=
    7
>>clear a b c % șterge cele 3 variabile
>> % tastand a si ENTER
>>a
??? Undefined function or variable 'a'
```

Se poate folosi varianta „clear” în mai multe feluri:

- clear a % șterge variabila a
- clear a b % șterge variabilele a și b
- clear a b c % șterge toate variabilele.

Câteva reguli în definirea variabilelor:

- variabilele pot fi definite cu litere mari sau mici. De exemplu a și

A sunt două variabile diferite (în limbajele de programare a și A sunt același lucru)

- Variabilele pot conține până la 19 caractere (dacă sunt definite folosind mai mult de 19 caractere doar primele 19 sunt memorate);
- Variabilele se definesc obligatoriu cu o literă, care poate fi urmată de alte litere și/sau cifre
- Dacă unei operații i se atribuie numele unei variabile deja existente, acea variabilă ia noua valoare.

### **1.9 Importarea și exportarea datelor în MATLAB**

Cele mai multe date externe sunt în format ASCII. Dacă datele sunt în format numeric și memorate în coloane de aceeași dimensiune, datele pot fi citite folosind:

```
>> load datafile.txt – ascii
```

Dacă datele sunt în format MATLAB se folosește:

```
>>load datafile
```

sau

```
>>load datafile.mat
```

Pentru a salva o variabilă în format MATLAB:

```
>>save % salveaza in format binar toate datele din spatiul de lucru in
```

```
>>% fisierul matlab.mat
```

```
>>save x % toate datele din spatiul de lucru sunt salvate in format
```

```
>>% binar in fisierul x.mat
```

```
>> save x M N P % variabilele M, N, si P din spatiul de lucru sunt
```

```
>>% salvate in format binar in fiserul x.mat
```

```
>>save x.ext M N P – ascii % salveaza in format ASCII pe 8 biti in
```

```
>>% fisierul x.ext variabilele M, N, si P din spatiul de lucru
```

```
>>save x.ext M N P – ascii double % salveaza in format ASCII pe 16
```

```
>>% biti in fisierul x.ext variabilele M, N, si P din spatiul de lucru
```

## 1.10 Reprezentarea numerelor întregi în MATLAB

În MATLAB sunt definite 4 clase pentru numerele întregi cu semn și 4 clase pentru numerele întregi fără semn. Tipurile de clase cu semn permit lucrul cu valori întregi pozitive sau negative, dar nu pot reprezenta un domeniu larg de valori ca în cazul claselor fără semn, deoarece un bit este folosit pentru a arăta dacă un număr este pozitiv sau negativ. Clasele fără semn pot reprezenta domenii largi de valori, dar aceste numere pot fi doar zero sau pozitive.

Valorile întregi se pot memora în MATLAB pe 1, 2, 4 sau 8 biți. Rezultă astfel că se poate salva timp și memorie dacă se folosește cel mai mic tip de reprezentare a numerelor întregi pentru o valoare. **De exemplu:** nu aveți nevoie de o reprezentare întreagă pe 32 de biți pentru a memora valoarea 100.

În tabelul următor se prezintă cele opt clase de numere întregi, valorile pe care le puteți memora cu ele și funcțiile de conversie pentru a crea în MATLAB tipul respectiv de dată:

| Clasa                       | Domeniul de valori           | Funcția de conversie |
|-----------------------------|------------------------------|----------------------|
| întreg cu semn pe 8 biți    | $-2^7$ până la $2^7-1$       | int8                 |
| întreg cu semn pe 16 biți   | $-2^{15}$ până la $2^{15}-1$ | int16                |
| întreg cu semn pe 32 biți   | $-2^{31}$ până la $2^{31}-1$ | int32                |
| întreg cu semn pe 64 biți   | $-2^{63}$ până la $2^{63}-1$ | int64                |
| întreg fără semn pe 8 biți  | 0 până la $2^8-1$            | uint8                |
| întreg fără semn pe 16 biți | 0 până la $2^{16}-1$         | uint16               |
| întreg fără semn pe 32 biți | 0 până la $2^{32}-1$         | uint32               |
| întreg fără semn pe 64 biți | 0 până la $2^{64}-1$         | uint64               |

Datele întregi sunt memorate în MATLAB ca variabile în virgulă mobilă în dublă precizie (**double**). Pentru a memora o valoare ca un întreg, ea trebuie convertită din **double** în tipul dorit de tip întreg.

De exemplu, pentru a memora 428 ca un întreg pe 16 biți o valoare x, trebuie introdus:

```
>>x=int16(428);
```

Dacă numărul care se transformă în întreg are o parte

fracționară, MATLAB îl rotunjește la cea mai apropiată valoare întreagă. Dacă partea fracționară este exact 0.5 atunci MATLAB alege între cele 2 valori succesive întregi pe cea care este mai mare în valoare absolută:

```
>>x=428.499;  
>>int16(x)  
ans=  
    428  
>>x=x+0.001;  
>>int16(x)  
ans=  
    429
```

În cazul în care un număr trebuie rotunjit folosind o altă metodă decât cea implicită, următoarele funcții sunt disponibile în MATLAB: „*round*”, „*fix*”, „*floor*” și „*ceil*”.

**Exemplu:** folosind funcția „*fix*” vă permite să suprascrieți peste valoarea implicită și rotunjită care tinde la 0 când partea fracționară este exact 0.5:

```
>>x=428.5;  
>>int16(fix(x))  
ans=  
    428
```

Operațiile aritmetice ce implică valori întregi și valori în virgulă mobilă vor rezulta întotdeauna în date de tip întreg. Acolo unde este necesar, rezultatul este rotunjit, în concordanță cu algoritmul utilizat.

**De exemplu:** valoarea exactă a produsului  $\text{int16}(428)*2.35$  este 1005.8, dar MATLAB o rotunjește la cel mai apropiat întreg:

```
>>int16(428)*2.35  
ans=  
    1006
```

Funcțiile de conversie sunt utile, spre exemplu, atunci când șiruri sunt convertite în întregi:



```
>>str='student';
```

```
>>int8(str)
```

```
ans=
```

```
115 116 117 100 101 110 116
```

Se observă că fiecare caracter al șirului are un corespondent întreg (s→115; t→116; ... ).

### 1.11 Operații aritmetice cu date de tip întreg

Se pot efectua următoarele operații:

1. numere întregi sau matrici cu numere întregi de același tip:

```
x = uint32([311 210 416]).*unit32(25);
```

2. numere întregi sau matrici cu numere întregi și scalari în virgulă mobilă în dublă precizie: se obține un rezultat ce este de același tip cu tipul datei întregi:

```
x = uint32([311 210 416]).*2.35;
```

Pentru toate operațiile binare în care o variabilă este de tip întreg sau matrice cu numere întregi și cealaltă variabilă este un scalar în dublă precizie, MATLAB efectuează calculul în dublă precizie și rezultatul obținut este convertit la tipul original al variabilei întregi.

Convertirea datelor numerice, caractere sau șiruri, a valorilor logice în variabile în dublă precizie se face folosind funcția **double**. De exemplu, transformarea unui întreg negativ într-o variabilă în dublă precizie este:

```
>>y=int64(-782543290763424829); % creaza un intreg pe 64 biti
```

```
>>x=double(y) % convertește în dubla precizie
```

```
x=
```

```
-7.8254e+017
```

Dacă se dorește ca o variabilă să fie memorată în simplă precizie, se folosește funcția **single** în acest scop:

```
>>x=single(42.415);
```

Simpla precizie se poate utiliza și în conversia datelor numerice, caractere sau șiruri, a valorilor logice. Tot funcția **single** este folosită, de exemplu, pentru a converti o valoare de tip întreg într-o valoare în virgulă mobilă în simplă precizie:

```
>>y=int64(-782543290763424829) %creaza intreg pe 64 de biti
```

```
>>x=single(y) %convertește în simpla precizie
```

x=  
-7.8254e+017

### 1.12 Operații algebrice în MATLAB

Calculare cu scalari: folosesc operatori aritmetici de bază utilizați și în limbajele de programare. În MATLAB există o particularitate legată de împărțirea *la dreapta* sau *la stânga*.

Operatorii aritmetici sunt:

- + adunare
- scădere
- \* înmulțire
- / împărțire la dreapta (a / b înseamnă a : b)
- \ împărțire la stânga (a \ b înseamnă b : a)
- ^ ridicare la putere

Când linia de instrucțiuni conține mai mult de un operator aritmetic, ordinea de efectuare a calculelor este următoarea:

- 1 - parantezele
- 2 - ridicarea la putere, de la stânga la dreapta
- 3 - înmulțirea și împărțirea, de la stânga la dreapta
- 4 - adunarea și scăderea, de la stânga la dreapta.

Aceste reguli se aplică scalarilor în mod diferit. Câteva **exemple**:

```
>> 3*4
ans =
    12
>> 4/5
ans =
    0.8000
>> 4\5
ans =
    1.2500
>> x=pi/2; y=sin(x)
y =
    1
```

```
>> z=0; w=exp(4*z)/5
```

```
w =
```

```
0.2000
```

În acest ultim calcul, deși nu este necesar, se recomandă ca expresia lui  $w$  să fie scrisă sub forma:  $w=(\exp(4*z))/5$ , care dă același rezultat; deoarece în unele cazuri cu multe operații aritmetice se pot face confuzii.

Am folosit în exemplele de mai sus două funcții „sin” și „exp”; prima reprezintă funcția sinus pentru un exponent  $x$  în radiani, iar a doua reprezintă funcția exponențială  $\exp(x)=e^x$ .

Ajungem astfel la prezentarea unor funcții elementare existente în MATLAB. Pentru a vedea lista cu aceste funcții, MATLAB oferă posibilitatea aflării rapide a acestora, precum și a modului lor de utilizare. De exemplu:

```
>>help elfun % listeaza functiile elementare MATLAB
```

```
>>help cos % da informatii despre functia cosinus
```

```
>>% si modul de utilizare
```

Lista funcțiilor elementare este dată în tabelul următor.

| <b>TRIGONOMETRICE</b> |                             | csch                | cosecanta hiperbolică           |
|-----------------------|-----------------------------|---------------------|---------------------------------|
| sin                   | sinus, unghi în radiani     | acsc                | inversa cosecantei, în radiani  |
| sind                  | sinus, unghi în grade       | acscd               | inversa cosecantei, în grade    |
| sinh                  | sinus hiperbolic            | acsch               | inversa cosecantei hiperbolice  |
| asin                  | inversa sinus, în radiani   | cot                 | cotangenta, unghi în radiani    |
| asind                 | inversa sinus, în grade     | cotd                | cotangenta, unghi în grade      |
| asinh                 | inversa sinus hiperbolic    | coth                | cotangenta hiperbolică          |
| cos                   | cosinus, unghi în radiani   | acot                | inversa cotangentei, în radiani |
| cosd                  | cos, unghi în grade         | acotd               | inversa cotangentei, în grade   |
| cosh                  | cosinus hiperbolic          | acoth               | inversa cotangentei hiperbolice |
| acos                  | inversa cosinus, în radiani | <b>EXPONENȚIALE</b> |                                 |
| acosd                 | inversa cosinus, în grade   | exp                 | exponent                        |
| acosh                 | inversa cosinus hiperbolic  | log                 | logaritmul natural              |
| tan                   | tangenta, unghi în radiani  | log10               | logaritmul comun                |
| tand                  | tangenta, unghi în grade    | sqrt                | radical                         |

|       |                                     |                 |  |
|-------|-------------------------------------|-----------------|--|
| tanh  | tangenta hiperbolică                | <b>COMPLEXE</b> |  |
| atan  | inversa tangentei, în radiani       | abs             | valoarea absolută                        |
| atand | inversa tangentei, în grade         | angle           | unghiul de fază                          |
| atan2 | inversa tangentei, în<br>cadranul 4 | conj            | conjugatul complex                       |
| atanh | inversa tangentei<br>hiperbolice    | imag            | partea complexă                          |
| sec   | secanta, unghi în radiani           | real            | partea reală                             |
| secd  | secanta, unghi în grade             | <b>NUMERICE</b> |  |
| sech  | secanta hiperbolică                 | fix             | rotunjește la zero                       |
| asec  | inversa secantei, în radiani        | floor           | rotunjește la minus infinit              |
| asecd | inversa secantei, în grade          | ceil            | rotunjește la plus infinit               |
| asech | inversa secantei hiperbolice        | round           | rotunjește la cel mai<br>apropiat întreg |
| csc   | cosecanta, unghi în radiani         | rem             | restul după împărțire                    |
| cscd  | cosecanta, unghi în grade           | sign            | funcția signum                           |