CAPITOLUL 1

Introducere în MATLAB

Cuprins

MATLAB[1] este un sistem interactiv destinat calculelor numerice. Prima versiune MATLAB a fost scrisă în anii ’70 de Cleve Moler. MATLAB uşurează sarcina utilizatorului de

[1]MATLAB® este o marcă înregistrată a Mathworks Inc., Natick MA

a rezolva problemele numerice. Aceasta permite concentrarea asupra părţii creatoare a rezolvării problemei şi încurajează experimentele. MATLAB utilizează algoritmi cunoscuţi şi testaţi, în care utilizatorul poate avea încredere. Operaţiile puternice se pot realiza uşor cu un număr mic de comenzi (de multe ori una sau două). Vă puteţi programa propriul set de funcţii pentru aplicaţia dumneavoastră. De asemenea, sunt disponibile facilităţi grafice excelente, iar imaginile pot fi inserate în documente LaTeX sau Word. Pentru o introducere mai detaliată în MATLAB a se vedea [30, 44, 39].

1.1. Lansarea MATLAB şi sistemul de help

Sub sistemul de operare Windows, MATLAB se lansează dând un click dublu pe iconul corespunzător sau selectând programul din meniul de start. Prompterul din fereastra de comandă este indicat prin `>>`. MATLAB poate fi utilizat în mai multe moduri: ca un calculator avansat (când comenzile sunt introduse în linia de comandă de la tastatură), ca un limbaj de programare de nivel înalt şi sub formă de rutine apelate dintr-un limbaj de programare, de exemplu C.

Informaţiile de help pot fi obţinute în mai multe moduri:

- din linia de comandă utilizând comanda '`help subiect`';
- dintr-o fereastră de help separată, deschisă prin meniul Help;
- utilizând MATLAB helpdesk memorat pe disc sau CD.

Comanda `help help` da o scurtă descriere a sistemului de help, iar `help` fară nici un parametru dă o listă a subiectelor de help. Primele linii arată astfel

```
HELP topics:

matlab\general - General purpose commands.
matlab\ops     - Operators and special characters.
matlab\lang    - Programming language constructs.
matlab\elmat   - Elementary matrices and matrix manipulation.
matlab\elfun   - Elementary math functions.
matlab\specfun - Specialized math functions.
matlab\matfun  - Matrix functions - numerical linear algebra.
```

Pentru a obţine informaţii de help despre funcţiile elementare se tastează

```
>> help elfun
```

Pentru a obţine doar un ecran la un moment dat se poate introduce întâi comanda `more on`, adică

```
>> more on
>> help elfun
```

Pentru a trece la următoarea pagină se poate apăsa orice tastă.

O altă facilitate utilă este utilizarea unei comenzi de forma `lookfor cuvant-cheie`, care caută în fişierele help un cuvânt cheie. Propunem cititorului să testeze `lookfor` `factorization`, care dă informaţii despre rutinele de factorizare a matricelor, deosebit de utile în algebra liniară.

Pentru începători şi cei care predau MATLAB demonstraţiile sunt foarte utile. Un set cuprinzător se poate lansa prin comanda

```
>> demo
```

Atenţie, ea şterge toate variabilele!

Înafară de facilitatea de help on-line, există un sistem bazat pe hipertext, care dă detalii asupra asupra celor mai multe comenzi şi exemple. El este disponibil prin comanda `doc`.

1.2. Modul calculator

Operaţiile aritmetice de bază sunt `+ - * /` şi ridicarea la putere `^`. Ordinea implicită a operaţiilor se poate schimba cu ajutorul parantezelor.

MATLAB recunoşte mai multe tipuri de numere:

- întregi, cum are fi 1362 sau -217897;
- reale, de exemplu 1.234, -10.76;
- complexe, cum ar fi $3.21 - 4.3i$, unde $i = \sqrt{-1}$;
- Inf, desemnează infinitul;
- NaN, Not a Number, care se obţine ca rezultat al unei operaţii ilegale sau al unei nedeterminări din analiza matematică (0/0, ∞/∞, $\infty - \infty$, etc.).

Notaţia cu exponent este de asemenea utilizată:

$$\begin{aligned} -1.3412e+03 &= -1.3412 \times 10^3 = -1341.2 \\ -1.3412e-01 &= -1.3412 \times 10^{-1} = -0.13412 \end{aligned}$$

Toate calculele se realizează în virgulă flotantă. Formatul în care MATLAB *afişează* numerele este controlat de comanda `format`. Tastaţi `help format` pentru o listă completă. Tabela următoare dă câteva exemple.

Comanda	Exemple de ieşiri
`format short`	`31.4162`(4 zecimale)
`format short e`	`31.416e+01`
`format long e`	`3.141592653589793e+000`
`format short g`	`31.4162`(4 zecimale)
`format bank`	`31.42`(2 zecimale)

Comanda `format compact` elimină liniile goale de la ieşire şi permite să se afişeze mai multă informaţie.

Numele de variabile în MATLAB sunt formate din secvenţe de litere şi cifre, prima fiind o literă. Exemple: `x`, `y`, `z525`, `TotalGeneral`. Se face distincţie între literele mari şi cele mici. Există şi nume speciale, a căror folosire trebuie evitată, cum ar fi:

- `eps = 2.2204e-16` = 2^{-54} este epsilon-ul maşinii (vezi capitolul 3) care reprezintă cel mai mare număr cu proprietatea că `1+eps` nu poate fi distins de 1;

- `pi` $= \pi$.

Dacă se fac calcule cu numere complexe folosirea variabilelor `i` şi `j` este contraindicată, deoarece ele desemnează unitatea imaginară. Dăm câteva exemple:

```
>>x = 3-2^4
x =
   -13
>>y = x*5
y =
   -65
>>eps
ans =
  2.2204e-016
```

Variabila specială `ans` păstrează valoarea ultimei expresii evaluate. Ea poate fi utilizată în expresii, la fel ca orice altă variabilă.

```
>>3-2^4
ans =
   -13
>>ans*5
ans =
   -65
```

Dacă dorim să suprimăm afişarea ultimei expresii evaluate, vom pune caracterul „;" la sfârşitul expresiei. Pe o linie de comandă se pot introduce mai multe expresii. Ele pot fi separate prin virgulă, caz în care valoarea expresiei terminată cu virgulă va fi afişată, sau cu „;", caz în care valoarea expresiei nu va fi afişată.

```
>> x=-13; y = 5*x, z = x^2+y, z2 = x^2-y;
y =
   -65
z =
   104
```

Dacă dorim să salvăm variabile, o putem face cu comanda

```
>>save nume-fisier lista-variabile
```

unde variabilele din `lista-variabile` sunt separate prin blanc. Se pot folosi în numele de variabile construcţii de tip wildcard, desemnate prin *. Rezultatul salvării se păstrează în fişierul `nume-fisier` de tip .mat, în format binar, specific MATLAB. Variabilele salvate pot fi încărcate prin

```
>>load nume-fisier
```

Se pot face salvări şi încărcări şi în format ascii, în dublă precizie sau prin adăugare la un fişier existent. Pentru detalii a se vedea `help save` şi `help load`.

Lista variabilelor utilizate în sesiunea curentă se poate vizualiza cu `whos`:

cos, sin, tan, csc, sec, cot	Funcţii trigonometrice
acos, asin, atan, atan2, asec, acsc, acot	Funcţii trigonometrice inverse
cosh, sinh, tanh, sech, csch, coth	Funcţii hiperbolice
acosh, asinh, atanh, asech, acsch, acoth	Funcţii hiperbolice inverse
log, log2, log10, exp, pow2, nextpow2	Funcţii exponenţiale
ceil, fix, floor, round	Rotunjiri
abs, angle, conj, imag, real	Complexe
mod, rem, sign	Rest, semn
airy, bessel*, beta*, erf*, expint, gamma*, legendre	Funcţii matematice
factor, gcd, isprime, lcm, primes, nchoosek, perms, rat, rats	Funcţii din teoria numerelor
cart2sph, cart2pol, pol2cart, sph2cart	Transformări de coordonate

Tabela 1.1: Funcţii elementare şi funcţii matematice speciale (`"fun*"` indică existenţa mai multor funcţii al căror nume începe cu ”`fun`”

```
>>whos
  Name      Size         Bytes  Class
  ans       1x1              8  double array
  i         1x1              8  double array
  v         1x3             24  double array
  x         1x1              8  double array
  y         1x1              8  double array
  z         1x1              8  double array
  z2        1x1              8  double array
Grand total is 7 elements using 72 bytes
```

Comanda

```
>>diary nume-fisier
```

salvează toate comenzile şi rezultatele afişate pe ecran (cu excepţia celor ale comenzilor grafice) în fişierul `nume-fisier`. Acest proces de „jurnalizare” se termină prin

```
>>diary off
```

1.3. Matrice

Matricele sunt tipuri de date fundamentale în MATLAB. Ele sunt de fapt tablouri multidimensionale în dublă precizie. Cele mai folosite sunt matricele bidimensionale, care sunt tablouri bidimensionale cu m linii şi n coloane. Vectorii linie ($m = 1$) şi coloană ($n = 1$) sunt cazuri particulare de matrice bidimensionale.

zeros	Matricea nulă
ones	Matrice formată din elemente 1
eye	Matricea identică
repmat	Replicarea şi pavarea tablourilor
rand	Numere aleatoare distribuite uniform
randn	Numere aleatoare distribuite normal
linspace	Vector de elemente echidistante
logspace	Vector de elemente spaţiate logaritmic

Tabela 1.2: Funcţii pentru generarea de matrice

1.3.1. Generarea matricelor

Există mai multe moduri de a genera matrice. Unul dintre ele este cel explicit, care utilizează parantezele pătrate. Ca separatori între elemente se folosesc blancul sau virgula în interiorul unei linii şi punctul şi virgula sau „newline" pentru a separa liniile:

```
>> A = [5 7 9
1 -3 -7]
A =
     5     7     9
     1    -3    -7
>> B = [-1 2 5; 9 0 5]
B =
    -1     2     5
     9     0     5
>> C = [0, 1; 3, -2; 4, 2]
C =
     0     1
     3    -2
     4     2
```

Dimensiunea unei matrice se poate obţine cu comanda `size`:

```
>> v = size(A)
v =
     2     3
>> [r, c] = size(A)
r =
     2
c =
     3
```

Prima formă returnează un vector cu două elemente ce conţine numărul de linii şi respectiv de coloane. A doua pune dimensiunile în variabile separate.

MATLAB are un set util de funcţii pentru construirea unor matrice speciale, vezi tabela 1.2. Matricele de zerouri, de elemente 1 şi matricele identice se obţin cu funcţiile `zeros`, `ones` şi respectiv `eye`. Toate au aceeaşi sintaxă. De exemplu, `zeros(m,n)` sau `zeros([m,n])` produce o matrice $m \times n$ de zerouri, în timp ce `zeros(n)` produce o matrice $n \times n$. Exemple:

```
>> zeros(2)
ans =
     0     0
     0     0
>> ones(2,3)
ans =
     1     1     1
     1     1     1
>> eye(3,2)
ans =
     1     0
     0     1
     0     0
```

O situaţie comună se întâlneşte atunci când se doreşte construirea unei matrice identice sau nule având o dimensiune egală cu a unei matrice date `A`. Aceasta se poate face cu `eye(size(A))`. O funcţie înrudită cu `size` este funcţia `length`: `length(A)` este cea mai mare dintre dimensiunile lui `A`. Astfel, pentru un vector $n \times 1$ sau $1 \times n$, `x`, `length(x)` returnează `n`.

Funcţiile `rand` şi `randn` generează matrice de numere (pseudo-)aleatoare, utilizând aceeaşi sintaxă ca şi `eye`. Funcţia `rand` produce o matrice de numere aleatoare având distribuţia uniformă pe intervalul [0,1]. Funcţia `randn` generează o matrice de numere aleatoare având distribuţia normală standard. Apelate fără argumente, ambele funcţii produc un singur număr aleator.

```
>> rand
ans =
    0.4057
>> rand(3)
ans =
    0.9355    0.8936    0.8132
    0.9169    0.0579    0.0099
    0.4103    0.3529    0.1389
```

În simulările şi experimentele cu numere aleatoare este important ca secvenţele de numere aleatoare să fie reproductibile. Numerele produse de `rand` depind de starea generatorului. Starea se poate seta prin comanda `rand('state',j)`. Pentru `j=0` generatorul `rand` este setat în starea iniţială (starea de la lansarea MATLAB). Pentru întregi `j` nenuli, generatorul este setat pe a `j`-a stare. Starea lui `randn` se setează în acelaşi mod. Perioadele lui `rand` şi `randn`, adică numărul de termeni generaţi înainte ca secvenţele să înceapă să se repete este mai mare decât $2^{1492} \approx 10^{449}$.

Matricele se pot construi şi în formă de bloc. Din matricea `B`, definită prin `B=[1 2; 3 4]`, putem crea

```
>> C=[B, zeros(2); ones(2), eye(2)]
C =
     1     2     0     0
     3     4     0     0
     1     1     1     0
     1     1     0     1
```

Matricele diagonale pe blocuri se pot defini utilizând funcţia `blkdiag`, care este mai uşor de utilizat decât notaţia cu paranteze pătrate. Exemplu:

```
>> A=blkdiag(2*eye(2),ones(2))
A =
     2     0     0     0
     0     2     0     0
     0     0     1     1
     0     0     1     1
```

Funcţia `repmat` permite construirea de matrice prin repetarea de subblocuri: `repmat(A,m,n)` crează o matrice de `m` pe `n` blocuri în care fiecare bloc este o copie a lui `A`. Dacă `n` lipseşte, valoarea sa implicită este `m`. Exemplu:

```
>> A=repmat(eye(2),2)
A =
     1     0     1     0
     0     1     0     1
     1     0     1     0
     0     1     0     1
```

Sunt disponibile şi comenzi pentru manipularea matricelor; vezi tabela 1.3.

`reshape`	Schimbarea dimensiunii
`diag`	Matrice diagonale şi diagonale ale matricelor
`blkdiag`	Matrice diagonală pe blocuri
`tril`	Extragerea părţii triunghiulare inferior
`triu`	Extragerea părţii triunghiulare inferior
`fliplr`	Rotire matrice în jurul axei de simetrie verticale
`flipud`	Rotire matrice în jurul axei de simetrie orizontale
`rot90`	Rotaţia unei matrice cu 90 de grade

Tabela 1.3: Funcţii de manipulare a matricelor

Funcţia `reshape` schimbă dimensiunile unei matrice: `reshape(A,m,n)` produce o matrice `m` pe `n` ale cărei elemente sunt luate coloană cu coloană din `A`. De exemplu:

```
>>A=[1 4 9; 16 25 36], B=reshape(A,3,2)
A =
     1     4     9
    16    25    36
B =
     1    25
    16     9
     4    36
```

Funcţia `diag` lucrează cu diagonalele unei matrice şi poate avea ca argument o matrice sau un vector. Pentru un vector `x`, `diag(x)` este matricea cu diagonala principală `x`:

```
>>diag([1,2,3])
```

```
ans =
     1     0     0
     0     2     0
     0     0     3
```

Mai general, `diag(x,k)` pune `x` pe diagonala cu numărul `k`, unde $\mathtt{k} = 0$ înseamnă diagonala principală, $\mathtt{k} > 0$ specifică diagonale situate deasupra diagonalei principale, iar $\mathtt{k} < 0$ diagonale dedesubtul diagonalei principale:

```
>> diag([1,2],1)
ans =
     0     1     0
     0     0     2
     0     0     0
>> diag([3 4],-2)
ans =
     0     0     0     0
     0     0     0     0
     3     0     0     0
     0     4     0     0
```

Pentru o matrice `A`, `diag(A)` este vectorul coloană format din elementele de pe diagonala principală a lui `A`. Pentru a produce o matrice diagonală având aceaşi diagonală ca `A` se va utiliza `diag(diag(A))`. Analog cazului vectorial, `diag(A,k)` produce un vector coloană construit din a `k`-a diagonală a lui `A`. Astfel dacă

```
A =
     2     3     5
     7    11    13
    17    19    23
```

atunci

```
>> diag(A)
ans =
     2
    11
    23
>> diag(A,-1)
ans =
     7
    19
```

`tril(A)` obţine partea triunghiulară inferior a lui `A` (elementele situate pe diagonală principală şi dedesubtul ei şi în rest zero). Analog lucrează `triu(A)` pentru partea triunghiulară superior. Mai general, `tril(A,k)` dă elementele situate pe diagonala a `k`-a a lui `A` şi dedesubtul ei, în timp ce `triu(A,k)` dă elementele situate pe a `k`-a diagonală a lui `A` şi deasupra ei. Pentru `A` ca mai sus:

```
>>tril(A)
ans =
```

`compan`	matrice companion
`gallery`	colecţie de matrice de test
`hadamard`	matrice Hadamard
`hankel`	matrice Hankel
`hilb`	matrice Hilbert
`invhilb`	inversa matricei Hilbert
`magic`	pătrat magic
`pascal`	matricea Pascal (coeficienţi binomiali)
`rosser`	matrice simetrică pentru testarea valorilor proprii
`toeplitz`	matrice Toeplitz
`vander`	matrice Vandermonde
`wilkinson`	matricea lui Wilkinson pentru testarea valorilor proprii

Tabela 1.4: Matrice speciale

```
     2     0     0
     7    11     0
    17    19    23
>>triu(A,1)
ans =
     0     3     5
     0     0    13
     0     0     0
>>triu(A,-1)
ans =
     2     3     5
     7    11    13
     0    19    23
```

MATLAB posedă un set de funcţii pentru generarea unor matrice speciale. Aceste matrice au proprietăţi interesante care le fac utile pentru construirea de exemple şi testarea algoritmilor. Ele sunt date în tabela 1.4. Vom exemplifica funcţiile `hilb` şi `vander` în secţiunea 4.2. Funcţia `gallery` asigură accesul la o colecţie bogată de matrice de test creată de Nicholas J. Higham [32]. Pentru detalii vezi `help gallery`.

1.3.2. Indexarea şi notaţia „:”

Pentru a permite accesul şi atribuirea la nivel de *submatrice*, MATLAB are o notaţie puternică bazată pe caracterul „:”. Ea este utilizată pentru a defini vectori care acţionează ca indici. Pentru scalarii `i` şi `j`, `i:j` desemnează vectorul linie cu elementele `i`, `i+1`, `...` `j` (pasul este 1). Un pas diferit, `s`, se specifică prin `i:s:j`. Exemple:

```
>> 1:5
ans =
    1     2     3     4     5
>> 4:-1:-2
ans =
```

```
    4     3     2     1     0    -1    -2
>> 0:.75:3
ans =
    0    0.7500    1.5000    2.2500    3.0000
```

Elementele individuale ale unei matrice se accesează prin `A(i,j)`, unde $i \geq 1$ şi $j \geq 1$ (indicii zero sau negativi nu sunt admişi în MATLAB). Notaţia `A(p:q,r:s)` desemnează submatricea constând din intersecţia liniilor de la `p` la `q` şi coloanelor de la `r` la `s` a lui `A`. Ca un caz special, caracterul „:" singur, ca specificator de linie şi coloană, desemnează toate elementele din acea linie sau coloană: `A(:,j)` este a `j`-a coloană a lui `A`, iar `A(i,:)` este a `i`-a linie. Cuvântul cheie `end` utilizat în acest context desemnează ultimul indice în dimensiunea specificată; astfel `A(end,:)` selectează ultima linie a lui `A`. În fine, o submatrice arbitrară poate fi selectată specificând indicii de linie şi coloană individuali. De exemplu, `A([i,j,k],[p,q])` produce submatricea dată de intersecţia liniilor `i`, `j` şi `k` şi coloanelor `p` şi `q`. Iată câteva exemple ce utilizează matricea

```
>> A = [
     2     3     5
     7    11    13
    17    19    23
]
```

a primelor nouă numere prime:

```
>> A(2,1)
ans =
     7
>> A(2:3,2:3)
ans =
    11    13
    19    23
>> A(:,1)
ans =
     2
     7
    17
>> A(2,:)
ans =
     7    11    13
>> A([1 3],[2 3])
ans =
     3     5
    19    23
```

Un caz mai special este `A(:)` care desemnează un vector coloană ce conţine toate elementele lui `A`, aşezate coloană după coloană, de la prima la ultima

```
>> B=A(:)
B =
     2
     7
```

```
    17
     3
    11
    19
     5
    13
    23
```

Când apare în partea stângă a unei atriburi, `A(:)` completează `A`, păstrându-i forma. Cu astfel de notaţie, matricea de numere prime 3×3, `A`, se poate defini prin

```
>> A=zeros(3); A(:)=primes(23); A=A'
A =
     2     3     5
     7    11    13
    17    19    23
```

Funcţia `primes` returnează un vector de numere prime mai mici sau egale cu argumentul ei. Transpunerea `A = A'` (vezi secţiunea 1.3.3) este necesară pentru a ordona numerele prime după linii nu după coloane.

Legată de notaţia „:" este funcţia `linspace`, care în loc de increment acceptă număr de puncte: `linspace(a,b,n)` generează `n` puncte echidistante între `a` şi `b`. Dacă `n` lipseşte, valoarea sa implicită este 100. Exemplu:

```
>> linspace(-1,1,9)
ans =
  Columns 1 through 7
  -1.0000   -0.7500   -0.5000   -0.2500   0   0.2500   0.5000
  Columns 8 through 9
   0.7500    1.0000
```

Notaţia `[]` înseamnă matricea vidă, 0×0. Atribuirea lui `[]` unei linii sau unei matrice este un mod de a şterge o linie sau o coloană a matricei:

```
>>A(2,:)=[]
A =
     2     3     5
    17    19    23
```

Acelaşi efect se obţine cu `A = A([1,3],:)`. Matricea vidă este de asemenea utilă ca indicator de poziţie într-o listă de argumente, aşa cum se va vedea în §1.3.4.

1.3.3. Operaţii în sens matricial şi în sens tablou

Pentru scalarii `a` şi `b`, operaţiile `+`, `-`, `/` and `^` produc rezultate evidente. Pe lângă operatorul uzual de împărţire, cu semnificaţia $\frac{a}{b}$, MATLAB are operatorul de împărţire stângă (backslash \), cu semnificaţia $\frac{b}{a}$. Pentru matrice, toate aceste operaţii pot fi realizate în sens matricial (în conformitate cu regulile algebrei matriciale) sau în sens tablou (element cu element). Tabela 1.5 dă lista lor.

Operaţia	Sens matricial	Sens tablou
Adunare	+	+
Scădere	-	-
Înmulţire	*	.*
Împărţire stângă	\	.\
Împărţire uzuală	/	./
Ridicare la putere	ˆ	.ˆ

Tabela 1.5: Operaţii pe matrice şi tablouri

Operaţiile de adunare şi scădere sunt identice atât în sens matricial cât şi în sens tablou. Produsul `A*B` este înmulţirea matricială uzuală. Operatorii de împărţire `/` şi \ definesc soluţii ale sistemelor liniare: `A\B` este soluţia `X` a lui `A*X = B`, în timp ce `A/B` este soluţia lui `X*B = A`. Exemple:

```
>> A=[2,3,5; 7,11,13; 17,19,23]
A =
     2     3     5
     7    11    13
    17    19    23
>> A=[1 2; 3 4], B=ones(2)
A =
     1     2
     3     4
B =
     1     1
     1     1
>> A+B
ans =
     2     3
     4     5
>> A*B
ans =
     3     3
     7     7
>> A\B
ans =
    -1    -1
     1     1
```

Înmulţirea şi împărţirea în sens tablou, sau pe elemente, se specifică precedând operatorul cu un punct. Dacă `A` şi `B` sunt matrice de aceeaşi dimensiune, atunci `C = A.*B` însemnă `C(i,j)=A(i,j)*B(i,j)` iar `C = A.\B` înseamnă `C(i,j)=B(i,j)/A(i,j)`. Pentru `A` şi `B` ca în exemplul precedent:

```
>> A.*B
ans =
     1     2
     3     4
```

```
>> B./A
ans =
    1.0000    0.5000
    0.3333    0.2500
```

Inversa unei matrice pătratice nesingulare se obţine cu funcţia `inv`, iar determinantul unei matrice pătratice cu `det`.

Ridicarea la putere ^ este definită ca putere a unei matrice, dar forma cu punct face ridicarea la putere element cu element. Astfel, dacă `A` este o matrice pătratică, atunci `A^2` este `A*A`, dar `A.^2` se obţine ridicând la pătrat fiecare element al lui `A`:

```
>> A^2, A.^2
ans =
     7    10
    15    22
ans =
     1     4
     9    16
```

Operaţia .^ permite ca exponentul să fie un tablou când dimensiunile bazei şi exponentului coincid, sau când baza este un scalar:

```
>> x=[1 2 3]; y=[2 3 4]; Z=[1 2; 3 4];
>> x.^y
ans =
     1     8    81
>> 2.^x
ans =
     2     4     8
>> 2.^Z
ans =
     2     4
     8    16
```

Ridicarea la putere a matricelor este definită pentru toate puterile, nu numai pentru întregi pozitivi. Dacă `n<0` este un întreg, atunci `A^n` este definit prin `inv(A)^(-n)`. Pentru `p` neîntreg, `A^p` este evaluată utilizând valorile proprii ale lui `A`; rezultatul poate fi incorect sau imprecis dacă `A` nu este diagonalizabilă sau când `A` este prost condiţionată din punct de vedere al valorilor proprii.

Transpusa conjugată a matricei `A` se obţine cu `A'`. Dacă `A` este reală, atunci aceasta este transpusa obişnuită. Transpusa fără conjugare se obţine cu `A.'`. Alternativele funcţionale `ctranspose(A)` şi `transpose(A)` sunt uneori mai convenabile.

În cazul particular al vectorilor coloană `x` şi `y`, `x'*y` este produsul scalar, care se poate obţine şi cu `dot(x,y)`. Produsul vectorial a doi vectori se poate obţine cu `cross`. Exemplu:

```
>> x=[-1 0 1]'; y=[3 4 5]';
>> x'*y
ans =
     2
>> dot(x,y)
```

```
ans =
     2
>> cross(x,y)
ans =
    -4
     8
    -4
```

La adunarea dintre un scalar şi o matrice, MATLAB va expanda scalarul într-o matrice cu toate elementele egale cu acel scalar. De exemplu

```
>> [4,3;2,1]+4
ans =
     8     7
     6     5
>> A=[1 -1]-6
A =
    -5    -7
```

Totuşi, dacă atribuirea are sens fără expandare, atunci va fi interpretată în acest mod. Astfel, dacă comanda precedentă este urmată de `A=1`, `A` va deveni scalarul 1, nu `ones(1,2)`. Dacă o matrice este înmulţită sau împărţită cu un scalar, operaţia se realizează element cu element:

```
>> [3 4 5; 4 5 6]/12
ans =
    0.2500    0.3333    0.4167
    0.3333    0.4167    0.5000
```

Funcţiile de matrice în sensul algebrei liniare au numele terminat în `m`: `expm`, `funm`, `logm`, `sqrtm`. De exemplu, pentru `A = [2 2; 0 2]`,

```
>> sqrt(A)
ans =
    1.4142    1.4142
         0    1.4142
>> sqrtm(A)
ans =
    1.4142    0.7071
         0    1.4142
>> ans*ans
ans =
    2.0000    2.0000
         0    2.0000
```

1.3.4. Analiza datelor

Tabela 1.6 dă funcţiile de bază pentru analiza datelor. Cel mai simplu mod de utilizare a lor este să fie aplicate unui vector, ca în exemplele

`max`	Maximul
`min`	Minimul
`mean`	Media
`median`	Mediana
`std`	Abaterea medie pătratică
`var`	Dispersia
`sort`	Sortare în ordine crescătoare
`sum`	Suma elementelor
`prod`	Produsul elementelor
`cumsum`	Suma cumulată
`cumprod`	Produsul cumulat
`diff`	Diferenţa elementelor

Tabela 1.6: Funcţii de bază pentru analiza datelor

```
>> x=[4 -8 -2 1 0]
x =
     4    -8    -2     1     0
>> [min(x) max(x)]
ans =
    -8     4
>>sort(x)
ans =
    -8    -2     0     1     4
>>sum(x)
ans =
    -5
```

Funcţia `sort` sortează crescător. Pentru un vector real `x`, se poate face sortarea descrescătoare cu `-sort(-x)`. Pentru vectori complecşi, `sort` sortează după valorile absolute.

Dacă argumentele sunt matrice, aceste funcţii acţionează pe coloane. Astfel, `max` şi `min` returnează un vector ce conţine elementul maxim şi respectiv cel minim al fiecărei coloane, `sum` returnează un vector ce conţine sumele coloanelor, iar `sort` sortează elementele din fiecare coloană a unei matrice în ordine crescătoare. Funcţiile `min` şi `max` pot returna un al doilea argument care specifică în care componente sunt situate elementul minim şi cel maxim. De exemplu, dacă

```
A =
     0    -1     2
     1     2    -4
     5    -3    -4
```

atunci

```
>>max(A)
ans =
     5     2     2
```

```
>>[m,i]=min(A)
m =
     0    -3    -4
i =
     1     3     2
```

Aşa cum ne arată acest exemplu, dacă există două sau mai multe elemente minimale într-o coloană, se returnează numai indicele primului. Cel mai mic element dintr-o matrice se poate obţine aplicând `min` de două ori succesiv:

```
>>min(min(A))
ans =
    -4
```

sau utilizând

```
>> min(A(:))
ans =
    -4
```

Funcţiile `max` şi `min` pot fi făcute să acţioneze pe linie printr-un al treilea argument:

```
>>max (A,[],2)
ans =
     2
     2
     5
```

Argumentul 2 din `max(A,[],2)` specifică maximul după a doua dimensiune, adică după indicele de coloană. Al doilea argument vid `[]` este necesar, deoarece `max` sau `min` cu două argumente returnează maximul sau minimul celor două argumente:

```
>>max(A,0)
ans =
     0     0     2
     1     2     0
     5     0     0
```

Funcţiile `sort` şi `sum` pot fi şi ele făcute să acţioneze pe linii, printr-un al doilea argument. Pentru detalii a se vedea `help sort` sau `doc sort`.

Funcţia `diff` calculează diferenţe. Aplicată unui vector `x` de lungime `n` produce vectorul `[x(2)-x(1) x(3)-x(2) ... x(n)-x(n-1)]` de lungime `n-1`. Exemplu

```
>>x=(1:8).^2
x =
     1     4     9    16    25    36    49    64
>>y=diff(x)
y =
     3     5     7     9    11    13    15
>>z=diff(y)
z =
     2     2     2     2     2     2
```

ischar	Testează dacă argumentul este şir de caractere(string)
isempty	Testează dacă argumentul este vid
isequal	Testează dacă tablourile sunt identice
isfinite	Testează dacă elementele unui tablou sunt finite
isieee	Testeză dacă maşina utilizează aritmetica IEEE
isinf	Testează dacă elementele unui tablou sunt inf
islogical	Testează dacă argumentul este un tablou logic
isnan	Test de NaN
isnumeric	Testează dacă argumentul este numeric
isreal	Testează dacă argumentul este tablou real
issparse	Testează dacă argumentul este tablou rar

Tabela 1.7: O selecţie de funcţii logice is*

1.3.5. Operatori relaţionali şi logici

Operatorii relaţionali în MATLAB sunt: == (egal), ~=(diferit), < (mai mic), > (mai mare), <= (mai mic sau egal) şi >= (mai mare sau egal). De notat că un singur egal = înseamnă atribuire.

Comparaţia între scalari produce 1 dacă relaţia este adevărată şi 0 în caz contrar. Comparaţiile sunt definite între matrice de aceeaşi dimensiune şi între o matrice şi un scalar, rezultatul fiind în ambele cazuri o matrice de 0 şi 1. La comparaţia matrice-matrice se compară perechile corespunzătoare de elemente, pe când la comparaţia matrice-scalar se compară scalarul cu fiecare element. De exemplu:

```
>> A=[1 2; 3 4]; B = 2*ones(2);
>> A == B
ans =
     0     1
     0     0
>>A > 2
ans =
     0     0
     1     1
```

Pentru a testa dacă dacă matricele A şi B sunt identice, se poate utiliza expresia isequal(A,B):

```
>> isequal(A,B)
ans =
     0
```

Mai există şi alte funcţii logice înrudite cu isequal şi al căror nume începe cu is. O selecţie a lor apare în tabela 1.7; pentru o listă completă a se tasta doc is. Funcţia isnan este utilă deoarece testul x == NaN produce întotdeauna 0 (false), chiar dacă x este NaN! (Un NaN este prin definiţie diferit de orice şi nu are o relaţie de ordine cu nimic, vezi secţiunea 3.4.)

Operatorii logici în MATLAB sunt: & (şi), | (sau), ~ (not), xor (sau exclusiv), all (adevărat dacă toate elementele unui vector sunt nenule), any (adevărat dacă cel puţin un element al unui vector este nenul). Dăm câteva exemple:

Nivel de precedență	Operator
1 (cea mai mare)	transpusa (`.'`), putere(`.^`), transpusa conjugată complexă(`'`), putere matricială(`^`)
2	plus unar (+), minus unar (-), negație (~)
3	înmulțire (`.*`), împărțire dreaptă (`./`), împărțire stângă (`.\`), înmulțire matricială (`*`), împărțire dreaptă matricială (`/`), împărțire stângă matricială (`\`)
4	adunare (+), scădere (-)
5	două puncte (:)
6	mai mic (<), mai mic sau egal (<=), mai mare (>), mai mare sau egal (>=) , egal (==) , diferit (~=)
7	și logic (&)
8 (cea mai mică)	sau logic (|)

Tabela 1.8: Precedența operatorilor

```
>> x = [-1 1 1]; y = [1 2 -3];
>> x>0 & y>0
ans =
     0     1     0
>> x>0 | y>0
ans =
     1     1     1
>> xor(x>0,y>0)
ans =
     1     0     1
>> any(x>0)
ans =
     1
>>all(x>0)
ans =
     0
```

De notat că `xor` trebuie apelat ca o funcție: `xor(a,b)`. Operatorii logici `and`, `or`, `not` și cei relaționali pot fi apelați și în formă funcțională: `and(a,b)`, ..., `eq(a,b)`, ...(vezi `help ops`).

Precedența operatorilor este rezumată în tabela 1.8 (vezi `help precedence`). MATLAB evaluează operatorii de precedență egală de la stânga la dreapta. Precedența se poate modifica cu ajutorul parantezelor.

De notat că versiunile MATLAB anterioare lui MATLAB 6 aveau aceeași precedență pentru `and` și `or` (spre deosebire de majoritatea limbajelor de programare). MathWorks recomandă folosirea parantezelor pentru a garanta obținerea rezultatelor identice în toate versiunile MATLAB.

Pentru matrice `all` returnează un vector linie ce conține rezultatul lui `all` aplicat fiecărei coloane. De aceea `all(all(A=B))` este un alt mod de a testa egalitatea matricelor `A` și `B`. Funcția `any` lucrează analog; de exemplu, `any(any(A==B))` returnează 1 dacă `A` și `B` au cel puţin un element egal şi 0 în caz contrar.

Comanda `find` returnează indicii corespunzători elementelor nenule ale unui vector. De exemplu,

```
>> x = [-3 1 0 -inf 0];
>> f = find(x)
f =
     1     2     4
```

Rezultatul lui `find` poate fi apoi utilizat pentru a selecta doar acele elemente ale vectorului:

```
>> x(f)
ans =
    -3     1    -Inf
```

Cu `x` ca în exemplul de mai sus, putem utiliza `find` pentru a obţine elementele finite ale lui `x`,

```
>> x(find(isfinite(x)))
ans =
    -3     1     0     0
```

şi să înlocuim componentele negative ale lui `x` cu zero:

```
>> x(find(x<0))=0
x =
     0     1     0     0     0
```

Când `find` se aplică matricei `A`, vectorul de indici corespunde lui `A` privită ca un vector coloană obţinut din aşezarea coloanelor una peste alta (adică `A(:)`), şi acest vector poate fi utilizat pentru a indexa `A`. În exemplul următor se utilizează `find` pentru a face zero toate elementele lui `A` care sunt mai mici decât elementele corespunzătoare ale lui `B`:

```
>> A = [4 2 16; 12 4 3], B = [12 3 1; 10 -1 7]
A =
     4     2    16
    12     4     3
B =
    12     3     1
    10    -1     7
>> f = find(A<B)
f =
     1
     3
     6
>> A(f) = 0
A =
     0     0    16
    12     4     0
```

În cazul matricelor, putem utiliza `find` sub forma `[i,j] = find(A)`, care returnează vectorii `i` şi `j` ce conţin indicii de linie şi coloană ale elementelor nenule.

Rezultatele operatorilor logici şi ale funcţiilor logice din MATLAB sunt tablouri de elemente 0 şi 1, care sunt exemple de tablouri logice. Astfel de tablouri pot fi create şi prin aplicarea funcţiei `logical` unui tablou numeric. Tablourile logice pot fi utilizate la indexare. Fie exemplul

```
>> clear
>> y = [1 2 0 -3 0]
y =
     1     2     0    -3     0
>> i1 = logical(y)
Warning: Values other than 0 or 1 converted to logical 1(Type
"warning off MATLAB:conversionToLogical" to suppress
this warning.)
>> i1 =
     1     1     0     1     0
>> i2 = ( y~=0 )
i2 =
     1     1     0     1     0
>> i3 = [1 1 0 1 0]
i3 =
     1     1     0     1     0
>> whos
  Name      Size            Bytes  Class
  i1        1x5                 5  logical array
  i2        1x5                 5  logical array
  i3        1x5                40  double array
  y         1x5                40  double array
Grand total is 20 elements using 90 bytes
>> y(i1)
ans =
     1     2    -3
>> y(i2)
ans =
     1     2    -3
>> isequal(i2,i3)
ans =
     1
>> y(i3)
??? Subscript indices must either be real positive
integers or logicals.
```

Acest exemplu ilustrează regula că `A(M)`, unde `M` este un tablou logic de aceeaşi dimensiune ca şi `A`, extrage elementele lui `A` corespunzând elementelor lui `M` cu partea reală nenulă. Chiar dacă `i2` are aceleaşi elemente ca `i3` (şi la comparaţie ele ies egale), doar tabloul logic `i2` poate fi utilizat la indexare.

Un apel la `find` poate fi uneori evitat dacă argumentul său este un tablou logic. În exemplul precedent, `x(find(isfinite(x)))` poate fi înlocuit cu `x(isfinite(x))`. Se recomandă utilizarea lui `find` pentru claritate.