Gabriela Ciuprina, Mihai Popescu,
Sorin Lup, Ruxandra Bărbulescu

Laborator de Metode Numerice - anul II

L0 - Programare în Matlab - recapitulare

24 septembrie 2019

Introducere

În cadrul laboratorului de metode numerice veţi implementa şi analiza o parte din algoritmii prezentaţi în cadrul cursului.

Implementările se vor face în Matlab. În Matlab se pot face implementări eficiente, bazate pe operaţii cu vectori şi matrice, care evită structurile repetitive. În multe din exerciţiile propuse se va cere implementarea explicită, corespunzătoare pseudocodului descris. O astfel de implementare, chiar dacă nu este cea mai eficientă pentru acest mediu de programare, va permite analiza experimentală a complexităţii algoritmilor.

Stilul de lucru care va fi promovat va fi următorul. Algoritmii se vor implementa în funcţii, iar pentru testarea lor se vor scrie programe principale în fişiere de tip script. Sintaxa se va modifica până când verificarea dată de `mlint` nu întoarce erori (de preferat ca pătrăţelul care există în mediul de lucru să fie verde, nu galben). Pentru fiecare algoritm implementat studenţii vor fi solicitaţi să creeze un exemplu simplu pentru care să poată verifica funcţionarea programului. Dacă programul nu calculează corect atunci se va folosi debuggerul din Matlab pentru depistarea erorilor codului.

Programele se vor scrie ordonat, cu comentarii. Primul comentariu conţine numele studentului, grupa şi data. Al doilea comentariu descrie ce face codul din fişierul respectiv.

Pe parcursul fiecărui laborator se va solicita completarea unui quiz pe moodle. Necompletarea unui astfel de quiz va însemna că aţi absentat.

În prima şedinţă de laborator se va recapitula folosirea Matlab ca limbaj de programare.

Cuprins

Capitolul 0

Programare în Matlab - recapitulare

Aceast capitol are ca scop familiarizarea cu mediul de lucru la laboratorul de Metode Numerice. Pentru fiecare din exerciţiile propuse, trebuie să completaţi un răspuns în chestionarul de pe Moodle. De aceea, simultan cu lectura acestui document trebuie să aveţi deschis şi chestionarul de pe moodle şi, bineînţeles, `Matlab`.

0.1 Variabile şi constante.

În rezolvarea cerinţelor de la laboratorul de Metode Numerice veţi folosi în mare măsură matrice cu elemente reale. Un număr poate fi considerat o matrice cu un singur element.

• *Dimensiunea unei matrice* nu trebuie declarată explicit. De fapt, nu există declaraţii de tip în Matlab. Cu toate acestea, trebuie să ştiţi foarte bine semnificaţia variabilelor folosite, atât pentru a scrie corect instrucţiunile cât şi pentru a putea efectua analiza complexităţii din punct de vedere al necesarului de memorie.

Exerciţiul 0.1:
Care este efectul comenzii:

```
>> a(10,5) = 1
```

• *Introducerea unei matrice* se poate face natural astfel:

```
>> A = [  1  2  3
          4  5  6
          7  8  9  ]
```

Într-o scriere compactă, liniile matricei pot fi separate prin ";" astfel:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

Pentru separarea elementelor unei linii se poate folosi caracterul blank (ca mai sus) sau virgula:

```
>> A = [1,2,3;4,5,6;7,8,9]
```

Exerciţiul 0.2:
Ştiind că vectorii sunt un caz particular de matrice, care sunt comenzile cu care se va introduce un vector linie, respectiv coloană cu elementele 1, 2, 3?

Exerciţiul 0.3:
Ce reprezintă e în următoarea instrucţiune?

```
>> u = 12.4e-3
```

Observaţie: variabilele utilizate într-o sesiune de lucru ocupă memoria sistemului pe măsură ce sunt definite. Pentru a vizualiza lista variabilelor existente la un moment dat şi memoria disponibilă se foloseşte comanda `who`. Pentru a vizualiza câtă memorie ocupă variabilele existente la un moment dat, se foloseşte comanda `whos`. Dacă se doreşte eliberarea memoriei de toate variabilele generate se foloseşte comanda `clear`.

Exerciţiul 0.4:
Executaţi instrucţiunea `clear`. Acum nu mai există nicio variabilă în mediul de lucru (verificaţi cu `who`). Executaţi totuşi, următoarele instrucţiuni. Comentaţi rezultatul lor.

```
>> 1i
>>i
>> 1j
>>j
>> pi
>> eps
```

0.2 Atribuiri şi expresii

Instrucţiunea de atribuire are sintaxa:

```
variabila = expresie
```

sau simplu:

```
expresie
```

în care `variabila` este numele unei variabile, iar `expresie` este un şir de operatori şi operanzi care respectă anumite reguli sintactice. În a doua formă, după evaluarea expresiei, rezultatul este atribuit variabilei predefinite `ans`.

- *Operatorii aritmetici*[1] recunoscuţi de ***Matlab*** sunt:

`+` adunare;
`-` scădere;
`*` înmulţire;
`/` împărţire la dreapta;
`\` împărţire la stânga;
`^` ridicare la putere.

Pentru transpunerea unei matrice se foloseşte operatorul "apostrof" ca în exemplul:

```
>> B = A'
```

în care matricea `B` se calculează ca transpusa matricei `A` dacă aceasta are elemente reale, sau ca transpusa şi conjugata dacă aceasta are elemente complexe.

Observaţii:

1. Adunarea şi scăderea pot fi efectuate:
   - între două matrice cu aceleaşi dimensiuni;
   - între o matrice şi un număr (caz în care numărul este adunat, respectiv scăzut din fiecare din elementele matricei).

2. Înmulţirea poate fi efectuată:
   - între două matrice dacă lungimea liniei primei matrice este egală cu lungimea coloanei celei de a doua matrice;
   - între un număr şi o matrice (caz în care numărul este înmulţit cu fiecare din elementele matricei);
   - între două matrice cu aceleaşi dimensiuni (element cu element), caz în care operatorul `*` este precedat de un punct, ca în exemplul:

   ```
   --> C = A .* B
   ```

3. Împărţirea matricelor poate fi făcută în mai multe feluri:
   - la dreapta (pentru matrice pătrate şi nesingulare):

[1] Operatorii aritmetici se aplică unor operanzi aritmetici, iar rezultatul este aritmetic.

```
--> X = B / A
```

echivalent cu:

```
--> X = B * inv(A)
```

sau cu:

```
--> X = B * A^(-1)
```

- la stânga:

```
--> X = A \ B
```

echivalent cu:

```
--> X = inv(A) * B
```

sau cu:

```
--> X = A^(-1) * B
```

Dacă A este o matrice dreptunghiulară de dimensiuni $m \times n$, iar b este un vector coloană cu m elemente, atunci împărţirea la stânga $x = A \setminus b$ calculează soluţia ecuaţiei $Ax = b$ în sensul celor mai mici pătrate.
- împărţirea unei matrice la un număr (să îl notăm cu `u`):

```
--> Y = A / u
```

- împărţirea element cu element a matricelor de dimensiuni egale:

```
--> C = A ./ B
```

sau

```
--> C = A .\ B
```

4. Ridicarea la putere `A^p` se face astfel [2]:
   - dacă `p` este un întreg pozitiv: dacă matricea `A` este pătrată atunci `A` se înmulţeşte cu ea însăşi de `p` ori; dacă matricea `A` este dreptunghiulară atunci se ridică la puterea `p` fiecare element din matricea `A`
   - dacă `p` este un întreg negativ: dacă matricea `A` este pătrată atunci inversa ei se înmulţeşte cu ea însăşi de $-p$ ori; dacă matricea `A` este dreptunghiulară atunci se ridică la puterea `p` fiecare element din matricea `A`
   - dacă `p` este un număr real (dar nu întreg) pozitiv: dacă matricea `A` este pătrată atunci calculul se face cu ajutorul vectorilor şi valorilor proprii ale matricei; dacă matricea `A` este dreptunghiulară, atunci se ridică la puterea `p` fiecare element din matricea A.
   - dacă `p` este un număr real (dar nu întreg) negativ: dacă matricea `A` este pătrată atunci calculul se face cu ajutorul vectorilor şi valorilor proprii ale inversei matricei; dacă matricea `A` este dreptunghiulară, atunci se ridică la puterea `p` fiecare element din matricea A.

- *Operatorii de relaţie*[3] recunoscuţi de ***Matlab*** sunt:

`<` mai mic decât;
`<=` mai mic sau egal cu;
`>` mai mare decât;
`>=` mai mare sau egal cu;
`==` egal cu;
`~=` diferit de.

Aceştia permit testarea unor condiţii, rezultatul având valoarea de adevăr `Fals` (0) sau `Adevărat` (1). Dacă operanzii sunt matrice de dimensiuni egale, atunci operaţiile logice se fac între elementele de pe aceleaşi poziţii, iar rezultatul este o matrice cu elementele 0 şi 1.

- *Operatori logici*[4] recunoscuţi de ***Matlab*** sunt:

`&` conjuncţia logică;
`|` disjuncţia logică;
`~` negaţia logică.

Dacă operanzii sunt matrice (logice) cu aceleaşi dimensiuni, atunci operaţia se face element cu element. Dacă unul din operanzi este o valoare logică, atunci acesta se combină

[2] Nu sunt descrise toate situaţiile posibile.
[3] Operatorii de relaţie se aplică unor operanzi aritmetici iar rezultatul este logic.
[4] Operatorii logici se aplică unor operanzi logici iar rezultatul este logic.

cu fiecare din elementele celuilalt operand. Alte situaţii nu sunt permise.

• *Funcţii elementare.* Operanzii unor expresii pot fi şi apeluri de funcţii elementare (de exemplu trigonometrice) sau alte funcţii cunoscute. Aceste funcţii aplicate unei matrice acţionează asupra fiecărui element în mod independent.

Exerciţiul 0.5:

Fie $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $b = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$ şi $v = \begin{bmatrix} 4 & 3 \end{bmatrix}$. Care sunt comenzile ***Matlab*** pentru rezolvarea ecuaţiei $A\left(v^T + x\right) = b$.

0.3 Generarea vectorilor şi matricelor

• *Vectorii ai căror elemente formează o progresie aritmetică* pot fi generaţi cu construcţia:

valIn : pas : valFin

Exerciţiul 0.6:

Comentaţi rezultatele următoarelor instrucţiuni:

```
>> x = 1:10
>> y = 1:2:10
>> z = 1:3:10
>> t = 1:-3:10
>> w = -1:-3:-10
>> u = -1:-10
>> v = -10:-1
>> a = 10:-2:-3
```

• *Vectorii ai căror elemente formează o progresie geometrică* pot fi generaţi cu construcţia:

```
logspace(d1, d2, n)
```

Exerciţiul 0.7:

a) Care este semnificaţia mărimilor $d1, d2, n$ din comanda `logspace` ?

b) Ce generează comanda `linspace` ?

• *Descrierea vectorilor şi matricelor pe blocuri.* Vectorii şi matricele pot fi descrise pe blocuri, folosind notaţii de forma:

```
A = [X Y; U V];
```

cu semnificaţia $A = \begin{bmatrix} X & Y \\ U & V \end{bmatrix}$, în care X, Y, U, V sunt matrice sau vectori.

Exerciţiul 0.8:

Care este rezultatul comenzii:

```
>> A = [1:3 ; 1:2:7]
```

• *Referirea la elementele unei matrice.* Pentru a obţine valoarea unui element, se folosesc construcţii de forma $a(1,1), a(1,2)$. Se pot obţine valorile mai multor elemente prin construcţii de forma $a(u,v)$ unde u şi v sunt vectori. De exemplu a(2,1:3) reprezintă primele trei elemente din linia a doua a matricei a. Pentru a obţine toate elementele liniei 2 se scrie $a(2,:)$.

Exerciţiul 0.9:

Fie $A = \begin{bmatrix} 1 & 10 & 100 & 1000 \\ 2 & 20 & 200 & 2000 \\ 3 & 30 & 300 & 3000 \end{bmatrix}$.

Notaţi rezultatele şi comentaţi următoarele comenzi:

```
>> A(0,1)
>> A(2,3)
>> A(:,3)
>> A(:,:)
>> A(3,:)
>> A(2,2:4)
>> A(2:3,2:4)
>> A(2:end,2:end)
>> A(:)
```

• *Generarea unor matrice particulare utile* se poate face cu ajutorul funcţiilor:

`eye` matrice cu elementele unitare pe diagonală şi nule în rest;

`zeros` matrice nulă;

`ones` matrice cu toate elementele unitare;

`rand` matrice cu elemente aleatoare în intervalul (0,1);

`diag` construieşte o matrice cu o anumită diagonală, sau extrage diagonala dintr-o matrice.

Exerciţiul 0.10:

Comentaţi următoarele comenzi (unde `A` este matricea de la exerciţiul 0.9 iar $v = [\ 1 \quad 2 \quad 3 \quad 4 \quad 5\ ]$):

```
>> eye(3)
```

```
>> eye(3,3)
>> eye(3,4)
>> diag(A)
>> diag(v)
```

Exerciţiul 0.11:
Comentaţi următoarele comenzi:

```
>> A = diag(1:3)
>> B = [A, eye(size(A)); ones(size(A)) zeros(size(A))]
>> C = diag(B)
>> D = C'*C
>> E = (D == 14)
```

• *Dimensiunile matricelor (vectorilor)* pot fi modificate în timpul execuţiei unui program. Pentru a obţine dimensiunea unei matrice X se foloseşte instrucţiunea:

```
[m, n] = size(X)
```

în care m reprezintă numărul de linii şi n numărul de coloane. Dimensiunea unui vector v se obţine cu:

```
length(v)
```

care are semnificaţia `max(size(v))`.

Exerciţiul 0.12:
Definiţi o matrice oarecare B (de exemplu cu 3 linii şi 4 coloane). Executaţi şi comentaţi următoarele instrucţiuni:

```
>> [m,n] = size(B)
>> B = [B; zeros(1, n)]
>> B = [B zeros(m+1,1)]
```

• *Matricea vidă.* ***Matlab*** operează şi cu conceptul de matrice vidă, notată cu [ ] şi care este o matrice de dimensiune nulă, fără elemente. Aceasta se dovedeşte utilă în eliminarea unor linii sau coloane dintr-o matrice dată. De exemplu, instrucţiunea

```
>> B(:, [2 4]) = []
```

are ca efect eliminarea coloanelor 2 şi 4 din matricea B. În acest fel, dimensiunea unei matrice poate să şi scadă în timpul execuţiei unui program, nu numai să crească prin adăugarea de noi elemente.

0.4 Instrucţiuni grafice

Funcţia principală pentru reprezentări grafice este:

```
plot
```

Ea are diferite variante, printre care:

```
plot(x,y)
```

în care x este vectorul variabilei independente, iar y este vectorul variabilei dependente. Instrucţiunea:

```
plot(A)
```

în care A este o matrice are ca efect reprezentarea grafică a variaţiei elementelor coloanelor matricei A în funcţie de indexul lor. Numărul de grafice este egal cu numărul de coloane.

Funcţiile auxiliare ca `title` şi `grid` permit completarea graficului cu un titlu şi respectiv adăugarea unui rastru. Completarea graficului poate fi făcută şi cu ajutorul interfeţei grafice, apăsând *Show plot tools*.

Exerciţiul 0.13:
Realizaţi un grafic ca în figura 1. El reprezintă funcţia $y(t) = 10\sin(t)$ pentru $t \in [0, 4\pi]$. Puneţi etichete axelor, rastrul şi legenda ca în figură.

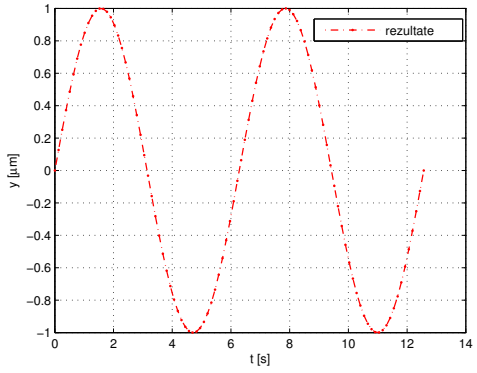

Figura 1: Acest grafic trebuie obţinut la exerciţiul 0.13.

0.5 Programare în *Matlab*

Matlab permite utilizarea unor structuri de control (decizii, cicluri, definiri de funcţii) ca orice limbaj de programare de nivel înalt. Se pot scrie programe în ***Matlab***, ca în orice limbaj de programare.

0.5.1 Editarea programelor

Până acum, aţi lucrat la consola ***Matlab***. Comenzile introduse pot fi scrise într-un fişier (numit script) şi apoi executate fie prin tastarea numelui fişierului script în consolă, fie prin apăsarea butonului *Run*.

Un script în ***Matlab*** are următoarea structură posibilă:

% comentarii

.....

instrucţiune; % fară afişarea rezultatului

instrucţiune % cu afişarea rezultatului

Exerciţiul 0.14:

Scrieţi comenzile cu care aţi rezolvat exerciţiul 0.13, într-un fişier numit `mytest.m` şi apoi executaţi-l fie cu comanda:

```
>> mytest
```

fie apăsând butonul *Run* din editor. Observaţi ce se întâmplă dacă la sfârşitul fiecărei instrucţiuni adăugaţi caracterul final ";".

IMPORTANT: Comenzile necesare rezolvării temelor ce vor urma vor fi scrise în fişiere.

0.5.2 Operaţii de intrare/ieşire

- *Introducerea datelor.*

Cea mai simplă metodă constă în utilizarea instrucţiunii de atribuire, ca în exemplul:

$a = 5$

În cazul unui program scris într-un fişier, este mai convenabil să se folosească funcţia `input`. Funcţia input se utilizează în atribuiri de forma:

variabila = `input`('text')

în care "variabila" este numele variabilei a cărei valoare va fi citită de la consolă, iar "text" este un şir de caractere ce va fi afişat, ajutând utilizatorul la identificarea informaţiei ce trebuie introdusă. De exemplu:

```
a = input('Introduceti valoarea variabilei a. a = ');
```

- *Inspectarea şi afişarea rezultatelor*

Pentru inspectarea valorilor variabilelor este suficient să fie invocat numele lor:

```
>> a
```

pentru ca interpretorul să afişeze valoarea lor.

Dacă se doreşte eliminarea afişării numelui variabilelor din faţa valorii sale, atunci se foloseşte funcţia `disp`:

```
>> disp(a)
```

Această funcţie poate fi folosită şi pentru afişarea textelor, de exemplu:

```
disp('Acest program calculeaza ceva ');
```

Formatul în care sunt afişate valorile numerice poate fi modificat de utilizator cu ajutorul instrucţiunii:

```
format
```

Operaţia de ieşire se poate realiza şi prin apelul funcţiei `fprintf` în instrucţiuni de forma:

`fprintf`('format',variabile)

în care "variabile" sunt variabilele care vor fi scrise în formatul corespunzător instrucţiunii, iar "format" este un şir de caractere ce descrie formatul de afişare. Sunt recunoscute următoarele construcţii, similare celor din limbajul C:

`%f` scrierea numărului în format cu virgulă fixă;
`%e` scrierea numărului în format cu exponent;
`%g` scrierea numărului în formatul cel mai potrivit (%f sau %e).

Celelalte caractere întâlnite în şirul "format" sunt afişate ca atare, de exemplu:

```
fprintf(' Rezultatul este a = %g', a);
```

Afişarea rezultatelor se poate face şi grafic (vezi paragraful 0.4).

Exerciţiul 0.15:
Scrieţi într-un fişier un program prietenos care va permite introducerea de la consola ***Matlab*** a două numere reale, va calcula suma lor, şi va afişa rezultatul în formatul cu exponent.

0.5.3 Structuri de control

- *Decizii*

Decizia simplă:

Pseudolimbaj	***Matlab***	Observaţii
dacă condiţie **atunci** instrucţiuni	`if` condiţie instrucţiuni `end`	"condiţie" este o expresie care este evaluată, iar dacă rezultatul este adevărat (1), atunci se execută "instrucţiuni", altfel se execută prima instrucţiune ce urmează după `end`.

Decizia cu alternativă:

Pseudolimbaj	***Matlab***	Observaţii

dacă condiţie **atunci**
instrucţiuni1
altfel
instrucţiuni2
	`if` condiţie
	instrucţiuni1
	`else`
	instrucţiuni2
	`end`
		"condiţie" este o expresie care este evaluată, iar dacă rezultatul este adevărat (1), atunci se execută "instrucţiuni1", iar dacă rezultatul este fals (0), se execută "instrucţiuni2".

Decizia de tip selecţie:

Pseudolimbaj	***Matlab***	Observaţii

dacă condiţie1 **atunci**
instrucţiuni1
altfel dacă condiţie2
instrucţiuni2
altfel
instrucţiuni3
	`if` condiţie1
	instrucţiuni1
	`elseif` condiţie2
	instrucţiuni2
	`else`
	instrucţiuni3
	`end`
		Pot exista oricâte alternative de selecţie.

	`switch` expresie, `case` expresie1 instrucţiuni1, `case` expresie2 instrucţiuni2, `otherwise` instrucţiuni, `end`	Pot exista oricâte cazuri. "instrucţiuni1" sunt executate dacă expresie == expresie1, etc.

- *Cicluri*

Ciclul cu test iniţial:

Pseudolimbaj	***Matlab***	Observaţii
cât timp condiţie instrucţiuni	`while` condiţie instrucţiuni `end`	Se repetă corpul ciclului, adică "instrucţiuni", cât timp "condiţie" este adevărată. S-ar putea ca "instrucţiuni" să nu fie executate niciodată în timpul rulării programului.

Ciclul cu contor:

Ciclul cu contor are două forme, din care a doua este cea generală. Dacă "expresie" este o matrice, atunci "variabila" ia succesiv valorile coloanelor matricei. "instrucţiuni" nu sunt executate niciodată dacă vectorul "valin:pas:valfin" este incorect definit (vid) sau dacă "expresie" este matricea vidă.

Pseudolimbaj	***Matlab***	Observaţii
pentru contor = valin, valfin, pas instrucţiuni	`for` contor = valin : pas : valfin, instrucţiuni `end`	Forma a doua este cea generală.
	`for` variabila = expresie, instrucţiuni `end`	

Ieşirile forţate din cicluri se pot face cu instrucţiunea `break`.

Exerciţiul 0.16:
Scrieţi un program care să determine cel mai mare număr întreg n pentru care 10^n poate fi reprezentat în ***Matlab***. Indicaţie: folosiţi un ciclu cu test, în care condiţia de intrare în ciclu testează egalitatea dintre 10^n şi constanta `Inf`.

0.5.4 Funcţii

Funcţiile sunt rutine ***Matlab*** care acceptă parametri de intrare şi întorc parametri de ieşire.

Este bine ca fiecare funcţie să fie definită într-un fişier separat, care are acelaşi nume ca numele funcţiei şi extensia *.m. Un fişier conţinând o funcţie trebuie să înceapă astfel:

`function`$[\ y_1, \ldots, y_n\ ] =$ nume_functie $(\ x_1, \ldots, x_m\ )$

unde y_i sunt variabilele de ieşire calculate în funcţie de variabilele de intrare x_j şi, eventual, de alte variabile existente în ***Matlab*** în momentul execuţiei funcţiei. Se recomandă ca acest fişier să se numească `nume_functie.m`.

Exerciţiul 0.17:
Editaţi un fişier numit "combin.m" cu următorul conţinut:

```
function [x,y] = combin(a,b)
x = a + b;
y = a - b;
```

şi un fişier numit "main_combin.m" cu următorul conţinut:

```
clear all;
a = input('Introduceti a. a = ');
b = input('Introduceti b. b = ');
[c,d] = combin(a,b);
fprintf(' Suma numerelor  a = %g si b = %g este a + b = %g',a,b,c);
fprintf(' Diferenta numerelor a = %g si b = %g este a - b  = %g',a,b,d);
```

Executaţi în ***Matlab*** comenzile din "main_combin.m":

```
>> main_combin;
```

a) Explicaţi comanda `fprintf(....)`;
b) Comentaţi necesitatea instrucţiunii `clear all`.
c) Rulaţi programul pas cu pas şi urmăriţi fereastra *Workspace*
d) Adaugaţi în funcţia `combin`, instrucţiunea inutilă $z = 7$. Rulaţi programul pas cu pas şi urmăriţi fereastra *Workspace*. Comentaţi.
e) Pe exemplul de la punctul c), daţi în consola Matlab comanda

```
>> mlint main_combin
```

Comentaţi rezultatul ei, după ce vă informaţi asupra comenzii `mlint`. Observaţi în permanenţă micul pătrat colorat (roşu, galben sau verde) din dreapta ecranului.

IMPORTANT: Folosirea comenzii `mlint` trebuie să fie o practică obişnuită în cazul folosirii limbajului Matlab. În versiunile noi de Matlab, rezultatul acestei comenzi este sugerat de un pătrăţel de culoare verde dacă nu sunt erori şi nici avertizări, galben dacă sunt doar avertizări şi roşu dacă există erori.

0.6 Implementări eficiente ale operaţiilor cu matrice

Unul din avantajele lucrului în Matlab este acela că el permite implementarea operaţiilor cu matrice. Aceasta nu numai că simplifică scrierea programelor, dar conduce la implementări mai eficiente deoarece Matlab are proceduri optimizate pentru aceste operaţii. Următoarele exerciţii ilustrează acest lucru.

Exerciţiul 0.18:
Scrieţi o funcţie `ps_v1` care să calculeze produsul scalar a doi vectori a şi b de dimensiune $1 \times n$ prin implementarea formulei $\sum_{i=1}^{n} a_i b_i$ şi o altă funcţie `ps_v2` care să implementeze calculul produsului scalar folosind operaţiile cu matrice $a * b'$. Verificaţi corectitudinea funcţiilor scrise cu ajutorul unui script în care să comparaţi rezultatele.

Exerciţiul 0.19:
Comentaţi conţinutul scriptului de mai jos. Executaţi-l şi comparaţi rezultatul cu cel din figura 2.

```
clear all;
nn = linspace(1e6,1e7,10);
N = length(nn);
t1 = zeros(1,N);
t2 = zeros(1,N);
```

```
for i = 1:length(nn);
    n = floor(nn(i));
    a = rand(1,n);
    b = rand(1,n);
    tic;
    rez1 = ps_v1(n,a,b);
    t1(i) = toc;
    tic;
    rez2 = ps_v2(a,b);
    t2(i) = toc;
end

plot(nn,t1,'bo-');
hold on;
plot(nn,t2,'r*-');
leg{1} = 'implementare cu for';
leg{2} = 'implementare a*b^\prime';
legend(leg);
xlabel('n');
ylabel('t [s]');
title('Timp de calcul al produsului scalar');
```

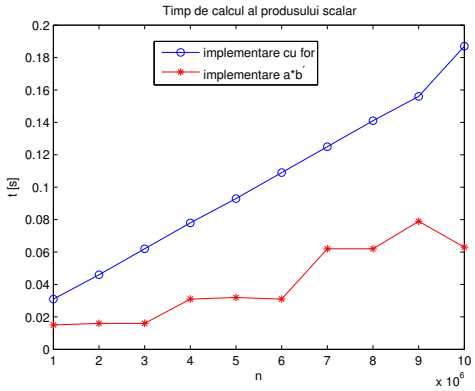

Figura 2: Timpul de calcul al produsului scalar în funcţie de dimensiunea vectorilor. Comparaţie între cele două implementări propuse la exerciţiul 0.19.

În concluzie, pentru a scrie programe eficiente, în Matlab trebuie folosit calculul matriceal ori de câte ori este posibil.

0.7 Lectură recomandată

- Documentaţia Matlab - disponibilă online la
  http://www.mathworks.com/access/helpdesk/help/helpdesk.html
- Clever Moler - Numerical Computing with Matlab, SIAM, 2004, disponibilă online la http://www.mathworks.com/moler/
- Pascal Getreuer - Writing fast Matlab code, 2009
  `http://www.math.ucla.edu/~getreuer/matopt.pdf`