E. SCHEIBER

Laborator de ANALIZĂ NUMERICĂ

SCILAB

Braşov

Tema de LABORATOR

Pentru a fi primit în examen trebuie efectuată şi predată cadrului didactic coordonator al activităţii de laborator *Lucrarea de laborator de Analiză numerică/Calcul numeric.*

Lucrarea constă din 2 teme:

- Rezolvarea a câte unei probleme din toate temele cuprinse în Culegerea de probleme.
- Programarea unor metode de calcul numeric.

Fiecare temă primeşte o notă iar media notelor reprezintă 50% din nota de examen.

Rezolvarea problemelor din Culegerea de probleme

Pe o foaie **A5** se vor scrie

1. Titlul capitolului;
2. Enunţul temei şi al problemei;
3. Rezultatele obţinute.

Pe prima foaie se trece

1. Numele şi prenumele;
2. Elemente de identificare a formaţiei de studiu;
3. Produsul informatic cu care s-au rezolvat problemele;
4. Numărul de ordine din catalog sau cel primit de la coordonatorul activităţii de laborator. În continuare ne referim la acest număr prin notaţia ID.

La o temă, un student rezolvă problema având numărul de ordine ID. Dacă ID este mai mare decât numărul problemelor atunci problema ce trebuie rezolvată se obţine cu formula

$$ID \mod^{1} NumarulProblemelor + 1.$$

Se va folosi următorul produs informatic

- Scilab - produs gratuit descărcabil din internet;

Îndrumarul de laborator de Analiză numerică/Calcul numeric prezintă modul de rezolvare a problemelor.

[1] Restul împărţirii.

Tema de programare

Programarea unor metode de calcul numeric

Fiecare student va prezenta 2 aplicaţii programate în Scilab.
Cele două aplicaţii se aleg dintre

1. Rezolvarea unui sistem algebric printr-o metodă finită;
2. Rezolvarea unui sistem algebric printr-o metodă iterativă;
3. Formulă de derivare numerică;
4. Formulă de integrare numerică.

Documentaţia realizată pe foi **A4** cuprinde:

1. Datele de identificare.
   (a) Autorul (nume, prenume, specializarea, grupa);
   (b) Data predării proiectului;
   (c) Tema proiectului.
2. Formulele de calcul utilizate.
3. Problema de test cu rezolvarea matematică şi calculele auxiliare pentru depanare.
4. Reprezentarea algoritmilor în pseudocod (l. română).

Exemplu de aplicaţie Scilab

Metoda tangentei pentru rezolvarea ecuaţiei algebrice $f(x) = 0,\ f : \mathbb{R} \to \mathbb{R}$ constă în construirea şirului $(x_k)_{k\in\mathbb{N}}$ definit prin $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.

Documentaţia cuprinde:

1. Formula de calcul.

$$\begin{aligned} x0 &\in \mathbb{R} \\ x_{k+1} &= x_k - \frac{f(x_k)}{f'(x_k)}, \quad k \in \mathbb{N} \end{aligned}$$

2. Problema de test.

   Să se rezolve ecuaţie $2^x - x^2 = 0, x0 = 1$

   S-au calculat $x_1 = 2.6294457, x_2 = 1.8807153$.

3. Algoritmul metodei este dat pe pagina următoare.

Algorithm 1 Pseudocodul metodei

procedure METODA TANGENTEI
    generarea aproximaţiei iniţiale xv
    $iter \leftarrow 0$
    do
        $iter \leftarrow iter + 1$
        $x \leftarrow xv - \frac{f(xv)}{f'(xv)}$
        $nrm \leftarrow |x - xv|$
        $xv \leftarrow x$
    while $(nrm \geq eps)$ şi $iter < nmi)$
    if $nrm < eps$ **then**
        $er \leftarrow 0$
    else
        $er \leftarrow 1$
    end if
    return x
end procedure

4. Textele sursă.

   Proiectul este alcătuit din 3 programe *Scilab*:

   (a) Datele problemei de test:

```
function [f,df,x0,nmi,tol]=datas()
    deff('y=f(x)','y=2.^x-x.*x');
    deff('y=df(x)','y=2.^x.*log(2)-2*x');
    x0=-0.5;
    nmi=50;
    tol=1e-5;
endfunction
```

(b) Rezolvarea problemei – nucleul proiectului:

```
function [x,er]=mtangentei(f,df,x0,nmi,tol)
    xv=x0;
    sw=%t;
    iter=0;
    while sw
        iter=iter+1;
        x=xv-f(xv)./df(xv);
        nrm=max(abs(x-xv));
        xv=x;
        printf("iter=%d x=%f\n",iter,x);
        if((iter>=nmi)|(nrm<=tol))
            sw=%f;
        end
        if(nrm<=tol)
            er=0;
        else
            er=1;
        end
    end
endfunction
```

(c) Apelarea aplicaţiei:

```
function aplicatia(path)
    exec(path+'\datas.sci',-1);
    exec(path+'\mtangentei.sci',-1);
    [f,df,x0,nmi,tol]=datas();
    [x,er]=mtangentei(f,df,x0,nmi,tol)
    printf('error_code = %d\n',er)
    printf('computed_solution = %f\n',x)
endfunction
```

În timp, la elaborarea acestui *Îndrumar de laborator* au contribuit:

- Silviu Dumitrescu
- Cristina Luca
- Vlad Monescu

Cuprins

I Scilab **10**

1 Elemente de programare în SCILAB **11**

1.1 Obiecte Scilab 11

1.2 Elemente de programare in Scilab 20

1.3 Funcţii(subprograme) în Scilab 23

1.4 Salvarea şi restaurarea datelor 26

1.5 Grafică în *Scilab* 27

1.5.1 Grafică 2D 27

1.5.2 Grafică 3D 28

2 Algebră liniară numerică **32**

2.1 Factorizarea unei matrice 32

2.2 Rezolvarea sistemelor algebrice de ecuaţii liniare 34

3 Rezolvarea sistemelor şi ecuaţiilor algebrice **37**

3.1 Rezolvarea sistemelor algebrice de ecuaţii neliniare 37

3.2 Rezolvarea ecuaţiilor algebrice 39

3.3 Rezolvarea ecuaţiilor polinomiale 40

4 Rezolvarea problemelor de interpolare **42**

4.1 Interpolare polinomială 42

4.2 Interpolare cu funcţii spline cubice 45

5 Derivare numerică **48**

5.1 Derivarea funcţiilor de o variabilă reală 48

5.2 Cazul funcţiilor de mai multe variabile 49

6 Metoda celor mai mici pătrate **51**

7 Integrare numerică **57**
7.1 Integrarea funcţiilor de o variabilă reală 57
7.2 Calculul numeric al integralelor duble 59

II CULEGERE DE PROBLEME **63**

8 Metode numerice în algebra liniară **64**

9 Sisteme şi ecuaţii algebrice **68**

10 Probleme de interpolare **72**

11 Derivare numerică **75**

12 Metoda celor mai mici pătrate **77**

13 Integrare numerică **78**

Bibliografie **80**

Part I

Scilab

Capitolul 1

Elemente de programare în SCILAB

⟶
Cap. 2

Prezentăm pe scurt elemente ale limbajului de programare utilizat în produsul *Scilab*. Elementele prezentate în secţiunile următoare reprezintă doar *ghid de utilizare* rapidă. Se presupune că utilizatorul posedă cunoştinţe de programare, având experienţă de lucru în cel puţin un limbaj de programare procedurală.

Scilab este un produs de calcul numeric produs de INRIA – Franţa – disponibil în mediile Windows şi Linux. Produsul este distribuit gratuit. Produsul este însoţit de o documentaţie cuprinzătoare.

Ca produs informatic *Scilab* are trăsături comune cu produsul comercial *Matlab*.

1.1 Obiecte Scilab

- **Constante**

Tip	Mnemonic	Valoare
real	%pi	π
real	%e	e
real	%eps	$\approx 2.210^{-16}$
	%inf	∞
complex	%i	i
boolean	%t	true
boolean	%f	false
polinom	%s	s
	%nan	*Not A Number*

```
%inf / %inf
   Nan
```

- **Literali**
  - **Identificatori** - denumiri date de utilizator diverselor entităţi Scilab (variabile, funcţii, etc).
    - Primul caracter al unui identificator este literă sau unul din caracterele %, _, #, !, $, ?
    - Următoarele caractere pot fi litere, cifre sau unul din caracterele _, #, !, $, ?
    - Se face distincţie între literele mari şi mici.
    - Numărul maxim de caractere este 24.
  - **Literali numerici**

    Exemple:
    1. Numărul real 10,23 se poate scrie:
    10.23=0.1023e+2=0.1023E+2=1023e-2=1023E-2.

    2. Numărul complex $1+5i$ se scrie $1+5*\%i$.
  - **Literali nenumerici**

    Un caracter sau un şir se caractere – string – se defineşte (scrie):

'c'	sau	*"c"*
'string'	sau	*"string"*

- **Funcţii matematice uzuale:**

Mnemonic	Semnificaţie	Mnemonic	Semnificaţie
abs(x)	$\|x\|$	exp(x)	e^x
log(x)	$\ln(x)$	log10(x)	$\lg(x)$
sin(x)	$\sin(x)$	asin(x)	$\arcsin(x)$
sinh(x)	$\mathrm{sh}(x)$	asinh(x)	$\mathrm{arcsh}(x)$
cos(x)	$\cos(x)$	acos(x)	$\arccos(x)$
cosh(x)	$\mathrm{ch}(x)$	acosh(x)	$\mathrm{arcch}(x)$
tan(x)	$\mathrm{tg}(x)$	atan(x)	$\mathrm{arctg}(x)$
tanh(x)	$\mathrm{th}(x)$	atanh(x)	$\mathrm{arcth}(x)$
cotg(x)	$\mathrm{ctg}(x)$	coth(x)	$\mathrm{cth}(x)$
sqrt(x)	$\sqrt{x}$	sinc(x)	$\frac{\sin x}{x}$
floor(x)	$\lfloor x \rfloor$	ceil(x)	$\lceil x \rceil$
erf(x)	$\frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t$	gamma(x)	$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} \mathrm{d}t$
gammah(x)	$\ln \Gamma(x)$	dlgamma(x)	$\frac{\Gamma'(x)}{\Gamma(x)}$
real(z)	$\Re(z)$	imag(z)	$\Im(z)$
int32(x)	conversie $\mathbb{Z}$		

Funcţii de mediu:

- Fixarea formatului de afişare a rezultatelor:
  `format(`*tip, lung*`)`
  unde
  $$tip = \begin{cases} 'v' & \text{reprezentare în virgulă fixă} \\ 'e' & \text{reprezentare cu exponent} \end{cases}$$
  şi $lung$ = numărul caracterelor destinate afişării.
- Tipul unui obiect *Scilab*
  `typeof(`*objScilab*`)`

- **Şiruri de numere**

  Progresia aritmetică introduce definind primul termen (a), raţia (r) şi o marginea superioară sau inferioară (M), după cum raţia este pozitivă sau negativă, prin sintaxa
  $$a : r : M$$
  Dacă parametrul r lipseşte atunci raţia este 1.

```
a=0.2:0.3:1
0.2 0.5. 0.8
```

```
b=1:-0.3:0
1. 0.7 0.4 0.1
```

Funcţia Scilab `linspace`(a,b,n) defineşte progresia $a+i\frac{b-a}{n-1}$, $i=0,1,\dots,n-1$.

Un şir de numere $(a_i)_{1\le i\le n}$ se poate defini prin

1. `i=1:n`
   `a=formula termenului general, functie de i`
2. $a=[a_1,a_2,\dots,a_n]$,
   unde $a_1,a_2,\dots,a_n$ sunt termenii şirului.

Indicele primului termen al unui şir este 1.

```
i=1:4
1 2 3 4
a=i^2
1 4 9 16
b=[1,4,9,16]
a==b
T T T T
```

- **Matrice**

  - **Definirea unei matrice.**
    Exemplu: Matricea
    $$a=\begin{pmatrix}1&2&3\\4&5&6\end{pmatrix}$$
    se defineşte în *Scilab* prin

    ```
    a=[1,2,3;4,5,6]
    ```

    Elementele unei linii se pot separa prin virgulă sau spaţiu. Elementele unei matrice pot fi: constante/variabile numerice, constante booleene, şiruri de caractere, polinoame, funcţii raţionale.
    O progresie aritmetică $prog=a:r:M$ este interpretată ca un vector linie (adică o matrice cu o singură linie).
    Exemplu. Progresiile aritmetice $a=0.2,0.6,1.0,1.4,1.8$ şi $b=0,-1,-2,-3,-4$ se obţin prin

```
a=0.2 : 0.4 : 2
a=
   !  0.2  0.6  1.0  1.4  1.8  !
b=-1 : -1 : -4
b=
   ! -1  -2  -3  -4  !
```

Funcţia *Scilab* `size`(*variabilaMatrice*) returnează numărul liniilor şi numărul coloanelor *variabileiMatrice*.

– **Definirea unei matrice rară** se face cu funcţia *Scilab*

$$a\texttt{=sparse}([l_1, c_1; l_2, c_2; \ldots], [v_1, v_2, \ldots])$$

unde $a_{l_i,c_i} = v_i.\ \forall i$.

Funcţia `full(`a`)` afişează o matrice rară a în formatul obişnuit.

Funcţia `nnz`(a) are ca valoare numărul elementelor nenule din a.

– **Matrice speciale.**

- `zeros`(m, n) defineşte o matrice nulă cu m linii şi n coloane;
- `ones`(m, n) defineşte o matrice cu toate elementele egale cu 1 având m linii şi n coloane;
- `eye`(m,n) defineşte o matrice unitate cu m linii şi n coloane;
- `rand`(m,n) defineşte o matrice cu m linii şi n coloane având ca elemente numere (semi)aleatoare cuprinse între 0 şi 1.

– **Selectarea unui element** se obţine prin sintaxa

variabilaMatrice(numărLinie, numărColoană)

Prima linie şi prima coloană are numărul de ordine 1.

Elementul unei matrice rare se selectează prin intermediul funcţiei `full`:
`full(`*variabilaMatrice(numărLinie, numărColoană*`).`

– **Operatori matriceali.**

Simbol	Semnificaţie
+	adunare
-	scădere
$*$	înmulţire de matrice
$.*$	înmulţire pe componente
ˆ	ridicare la putere prin produs matriceal
.ˆ	ridicarea la putere a componentelor
\	$a\backslash b = a^{-1} \cdot b$
. \	$a.\backslash b = (\frac{b_{i,j}}{a_{i,j}})_{i,j}$
/	$b/a = b \cdot a^{-1}$
. /	$b/a = (\frac{b_{i,j}}{a_{i,j}})_{i,j}$
'	transpunere

– **Funcţii matriceale.**

Mnemonic	Semnificaţia
triu(A)	Matricea superior triunghiulară a lui A
tril(A)	Matricea inferior triunghiulară a lui A
diag(A)	Vectorul cu elementele diagonale ale lui A
size(A)	Şir cu dimensiunile lui A
length(A)	Numărul elementelor lui A
max(A)	Cel mai mare element al lui A
min(A)	Cel mai mic element al lui A
matrix(A, m, n)	Transformă matricea A într-o matrice cu m linii şi n coloane

– **Extinderea unei matrice.** Date fiind o matrice a şi un vector linie v

* adăugarea liniei v ca prima linie a matricei a se obţine prin

$$[v; a]$$

* adăugarea liniei v ca ultimă linie a matricei a se obţine prin

$$[a; v]$$

* adăugarea vectorului v ca prima coloană a matricei a se obţine prin

$$[v', a] \quad \text{sau} \quad [v; a']'$$

* adăugarea vectorului v ca ultimă coloană a matricei a se obţine prin

$$[a, v'] \quad \text{sau} \quad [a'; v]'$$

Trebuie observată diferenţa dintre cazul în care extinderea se face prin linie faţă de cel în care se extinde prin coloană prin utilizarea “;” şi respectiv “,”.

- **Extragerea unei submatrice.** O zonă compactă fixată prin liniile $l_1 : l_2$ şi coloanele $c_1 : c_2$, inclusiv, se extrage din matricea a prin

$$a(l_1 : l_2, c_1 : c_2)$$

Pentru extragerea unei zone necompacte – aflată la intersecţia liniilor $l_1, l_2, \ldots, l_p$ cu coloanele $c_1, c_2, \ldots, c_q$ – dintr-o matrice a având m linii şi n coloane există opţiunile:

∗

$$a([l_1, l_2, \ldots, l_p], [c_1, c_2, \ldots, c_q])$$

∗ se definesc vectori linie

$$lin = (lin_i)_{1 \le i \le m}, \qquad lin_i = \begin{cases} \%t & \text{dacă } i \in \{l_1, \ldots, l_p\} \\ \%f & \text{în caz contrar} \end{cases}$$

$$col = (col_j)_{1 \le j \le n}, \qquad col_j = \begin{cases} \%t & \text{dacă } j \in \{c_1, \ldots, c_q\} \\ \%f & \text{în caz contrar} \end{cases}$$

Extragerea rezultă din

$$a(lin, col)$$

Comenzile $a(:, \$)$ şi $a(\$, :)$ extrag din matricea a ultima coloană, respectiv ultima linie.

- **Polinoame şi expresii raţionale**

Scilab posedă o modalitate deosebită de lucru cu polinoame definite peste R sau C.

Pentru a defini un polinom în nedeterminata (variabila) X se poate defini întâi nedeterminata

```
X=poly(0,'X');
```

iar apoi polinomul se introduce nemijlocit, de exemplu

```
p=X^2+3*X+2
```

Altfel, un polinom se defineşte prin intermediul funcţiei *Scilab* `poly`

$$polinom = \texttt{poly}(vector,\ nedeterminată,\ [string])$$

Dacă $vector = (a_1, a_2, \ldots, a_n)$ şi *string='coeff'* atunci se obţine polinomul de grad $n-1$

$$polinom = a_1 + a_2 X + \ldots + a_n X^{n-1},$$

iar dacă *string='roots'* atunci se obţine polinomul de grad n

$$polinom = (X - a_1)(X - a_2)\ldots(X - a_n)$$

A doua variantă este cea implicită.

```
v=[1,2,3];
p=poly(v,'X')
p=
   -6 + 11X - 6X^2 +X^3
q=poly(v,'X','coeff')
q=
    1 + 2X + 3X^2
```

Dacă A este o matrice pătrată atunci `poly`$(A, 'X')$ generează polinomul caracteristic matricei A, adică $\det(X \cdot I - A)$.

Dacă p şi q sunt două polinoame în nederminata X atunci p/q defineşte o funcţie raţională.

Scilab efectuează operaţiile uzuale cu polinoame şi funcţii raţionale în mod simbolic.

Dacă r este o funcţie raţională atunci r('`num`') sau `numer`(r) furnizează numărătorul şi r('`den`') sau `denom`(r) furnizează numitorul lui r.

Funcţii *Scilab* având polinoame ca argumente:

Mnemonic	Semnificaţia
$v = coeff(p)$	returnează coeficienţii polinomului p
$v = horner(p, x)$	p polinom, $v = p(x)$
$q = derivat(p)$	p, q polinoame, $q = p'$
$[q, U] = gcd([p_1, p_2, \ldots])$	$q = cmmdc(p_1, p_2, \ldots), [p_1, p_2, \ldots] * U = [0, \ldots, 0, q]$ U matrice pătrată de dimensiune $[p_1, p_2, \ldots]$ *greatest common divisor*
$[q, U] = lcm([p_1, p_2, \ldots])$	$q = cmmmc(p_1, p_2, \ldots)$ $[p_1, p_2, \ldots].* U = q * ones([p_1, p_2, \ldots])$ *least common multiple*

```
X=poly(0,'X')
p=(X-1)*(X-2)^2
q=(X-1)*(X-2)*(X-3)
[c,U]=gcd([p.q]);
c
   2-3X+X^2
[p,q]*U
   0    2-3X+X^2
[d,V]=lcm([p,q]);
d
   12-28X+23X^2-8X^3+X^4
[p,q].*V-d*ones([p,q])
   0    0
```

- **Listă.** O listă se defineşte prin

  $$\texttt{list}(e_1, e_2, \ldots, e_n)$$

  unde e_i sunt obiecte *Scilab*.

  Exemplu.

```
l=list(1,%t,'abc',[1,2;3,4])
 l=
    l(1)
  1
    l(2)
  T
    l(3)
  abc
    l(4)
 ! 1.  2.  3.  4. !
```

  Al i-lea element al unei liste l este accesibil cu $l(i)$.

  $l(i)$=`null` şterge al i-lea element din lista l.

Exemplificarea tipurilor obiectelor *Scilab* cu funcţia `typeof`:

```
x=1
typeof(x)
    constant
typeof(int32(%pi))
    int32
typeof('c')
```

```
    string
typeof(exp)
    fptr
typeof(1:5)
    constant
typeof(%t)
    boolean
p=poly(0,'X')
typeof(p)
    polynomial
l=list(1,'c')
typeof(l)
    list
deff('y=f(x)','y=x.^2')
typeof(f)
    function
l=[1,1];v=1;m=sparse(l,v)
typeof(m)
   sparse
```

1.2 Elemente de programare in Scilab

Programarea în *Scilab* se face în cadrul funcţiilor definite în exterior. Pentru editarea acestor funcţii *Scilab* posedă un editor propriu, dar se poate fi utilizat orice editor de fişiere.

O linie de comentariu este // *text - comentariu.*

Fiecare instrucţiune introdusă în dreptul promptului *Scilab* este executată şi rezultatul este afişat. Pe o linie pot fi puse mai multe instrucţiuni separate prin , sau ;. În plus, încheierea unei comenzi cu caracterul ; are ca efect inhibarea afişării rezultatului.

Pentru exprimarea condiţiilor se utilizează

Operatori relaţionali		Operatori logici	
Simbol	Semnificaţie	Simbol	Semnificaţie
==	=	&	şi
~=	$\neq$	|	sau
<=	$\leq$	~	negaţia
<	<		
>=	$\geq$		
>	>		

- **Instrucţiunea de atribuire**
  variabila=expresie

- **Instrucţiuni condiţionate**

1.

`if` *condiţie* $\left\{ \begin{array}{c} , \\ \text{then} \end{array} \right\}$

    instrucţiuni

  `elseif` *condiţie* $\left\{ \begin{array}{c} , \\ \text{then} \end{array} \right\}$

    instrucţiuni

  `else`

    instrucţiuni

`end`

Separatorii , sau `then` sunt obligatorii doar în cazul în care o instrucţiune se scrie pe aceaşi linie cu `if` .

Exemplul 1.1 *Rezolvarea ecuatiei* $x^2 + ax + b = 0$ *în linie de comandă.*

```
a=3;b=2;delta=a^2-4*b;
if delta>=0 then x=(-a+sqrt(delta))/2;y=(-a-sqrt(delta))/2;
        else x=-a/2;y=sqrt(-delta)/2;end
```

2.

`select` *expresie*

  `case` $expresie_1$ `then` *instrucţiuni*

$\vdots$

  `else`

    instrucţiuni

`end`

`case` şi `then` trebuie scrise pe aceeaşi linie.

- **Instrucţiuni de ciclare**

  1.

  `for` $n = n1 : pas : n2 \left\{ \begin{array}{c} , \\ do \end{array} \right\}$ *instrucţiuni* `end`

  Dacă $pas = 1$ atunci acest parametru este opţional.

  Separatorii , sau `do` sunt obligatorii doar în cazul în care o instrucţiune se scrie pe aceaşi linie cu `for` .

Exemplul 1.2 $\sum_{i=1}^{5} i$

```
s=0;
for i=1:5, s=s+i; end
```

Exemplul 1.3 *Împărţirea elementelor unei mulţimi în clase de echivalenţă modulo 3.*

```
a=[1,2,3,4,5,6,7,8,9];
c0=zeros(3,1);c1=zeros(3,1);c2=zeros(3,1);
i0=0;i1=0;i2=0;
for i=1:length(a),
  select a(i)-floor(a(i)/3)*3
    case 0 then i0=i0+1;c0(i0)=a(i);
    case 1 then i1=i1+1;c1(i1)=a(i);
    case 2 then i2=i2+1;c2(i2)=a(i);
  end
end
```

Exemplul 1.4 *Şirul lui Fibonacci generat în linie de comandă.*

```
F=[1,1];
n=15;
for i=3:n do F(i)=F(i-1)+F(i-2); end
```

2.

$$\texttt{while}\ \textit{condiţie} \left\{ \begin{array}{c} , \\ \texttt{do} \\ \texttt{then} \end{array} \right\} \textit{instrucţiuni}\ \texttt{end}$$

`while` şi `do` sau `then` trebuie scrise pe aceaşi linie.

Separatorii , `then` sau `do` sunt obligatorii doar în cazul în care o instrucţiune se scrie pe aceaşi linie cu `while`.

Exemplul 1.5 $\sum_{i=1}^{5} i$

```
s=0;
i=0;
while i<5, i=i+1;s=s+i; end
```

Instrucţiunea `break` realizează un salt necondiţionat la prima instrucţiune aflată după instrucţiunea de ciclare.

Instrucţiunea `continue` realizează un salt necondiţionat la următorul pas al ciclului.

Editarea unei instrucţiuni în linie de comandă se face într- o singură linie de cod, prezenţa virgulei este obligatorie (acolo unde este cazul). Editarea codul în funcţii externe se poate face pe mai multe linii, utilizând indentarea (scrierea decalată). În acest caz se poate omite scrierea virgulelor date în sintaxa instrucţiunilor.

1.3 Funcţii(subprograme) în Scilab

O funcţie *Scilab* este corect definită dacă produce rezultate pentru diferite tipuri de variabile: numere, vectori, matrice. În acest scop, uzual operaţiile de înmulţire, împărţire şi ridicare la putere se folosesc în varianta cu punct, adică operaţiile se fac pe componente.

Funcţiile *Scilab* predefinite satisfac această cerinţă.

Funcţiile se pot defini în

- **Linie de comandă.**

  deff('$[y_1,\ldots,y_n]=f(x_1,\ldots,x_m)$',['$y_1 = expresie_1$',...,'$y_n = expresie_n$'])

  Funcţia $f:\mathbb{R}^2 \to \mathbb{R}^2$ definită prin

  $$f(x,y) = \left(\begin{array}{c} x+y \\ x^2+y^2 \end{array}\right)$$

  se defineşte prin

  ```
  deff('[u,v]=f(x,y)',['u=x+y','v=x^2+y^2'])
  ```

  Variabilele funcţiei pot fi numere, vectori, matrice. În cazul în care variabilele sunt tablouri, operaţiile algebrice se fac între componentele corespunzătoare.

  O funcţie definită în linie de comandă se salvează

  save('*cale**numeFişier*.bin','f')

  Funcţia `save` salvează orice tip de obiect. Dacă a,b sunt variabile *Scilab* atunci sintaxa utilizată va fi save('*cale**numeFişier*.bin','a','b').

  Reîncărcarea se realizează prin

  load('*cale**numeFişier*.bin')

- **Exterior.** Cu un editor de fişiere se crează subprogramul

  function $[y_1,\ldots,y_n]=f(x_1,\ldots,x_m)$
      instructiuni Scilab
  endfunction

  care se salvează într-un fişier cu extensia `sci` sau `sce`.

  Pentru a putea fi utilizată funcţia exterioară trebuie încărcată prin `exec`(’*cale**numeFisier*`.sci`’,-1)

Valoarea -1 pentru al doilea parametru are ca efect lipsa afişărilor legate de operaţia de încărcare a funcţiei.

Este recomandat *numeFisier=numeFuncţie,* restricţie obligatorie în cazul în care subprogramul se include într-o bibliotecă.

Pentru funcţia din exemplul anterior, codul *macro*-ului de definiţie exterioară este

```
function [u,v]=f(x,y)
    u=x+y;
    v=x^2+y^2;
  endfunction
```

Funcţiile definite în exterior pot fi organizate în biblioteci. Astfel funcţiile *Scilab* – deci având extensia `.sci` – aflate într-un catalog localizat prin *cale* sunt reunite într-o bibliotecă prin comanda

`genlib`(’*numeBibliotecă*’,’*cale*’)

Prin această operaţie are loc compilarea funcţiilor şi crearea unui fişier `names` cu cuprinsul bibliotecii.

O bibliotecă odată creată, resursele ei pot fi utilizate după reîncărcarea ei prin

`lib`(’*cale*’)

Exemplul 1.6 *Funcţie pentru rezolvarea ecuaţiei de gradul al doilea.*

```
function [u,v]=eq2(a,b)
  delta=a*a-4*b
  if delta>=0
    u=0.5*(-a+sqrt(delta))
    v=0.5*(-a-sqrt(delta))
    disp('Real roots')
  else
    u=-a/2
    v=sqrt(-delta)
    disp('Complex conjugate roots')
  end
endfunction
```

Funcţia `disp(`*obiectScilab*`)` afişează valoarea argumentului.

Exemplul 1.7 *Funcţie pentru calculul valorii unui polinom.*

```
function r=polyval1(x,c)
  [l,n]=size(c)
  r=c(1)
  for i=2:n
    r=r*x+c(i)
  end
endfunction

function r=polyval2(x,c)
  [l,n]=size(c)
  i=0:n-1
  u=x^i
  r=u*c'
endfunction
```

Exemplul 1.8 *Funcţie pentru calculul primelor* n *termeni ai şirului Fibonacci.*

```
function t=fib(n)
  t=[1,1];
  for i=3:n
    t(i)=t(i-1)+t(i-2)
  end
endfunction
```

Exemplul 1.9 *Funcţie pentru calculul limitei şirului* $(a_n)_{n\in\mathbb{N}}$ *definit prin* $a_0 = \sqrt{2},\ a_{n+1} = \sqrt{a_n+2}$.

```
function [l,error]=lim(tol,nmi)
  v=sqrt(2)
  cont=%t
  ni=1
  while(cont)
    ni=ni+1
    l=sqrt(v+2)
    d=abs(v-l)
    v=l
```

```
    if(d<tol)&(ni>nmi)
      cont=%f
    end
  end
  if(d<tol)
    error=0
  else
    error=1
  end
endfunction
```

O functie Scilab poate avea un număr variabil de parametri formali de intrare. Aceşti parametri se indică prin variabila `varargin`.

Exemplul 1.10 *Procedură pentru generarea nodurilor lui Cebîşev de speţa a doua dintr-un interval* $[l,r]$. *În lipsa parametrilor suplimentari intervalul implicit este* $[-1,1]$.

În intervalul $[-1,1]$, pentru $n\in\mathbb{N}^*$, nodurilor lui Cebîşev de speţa a doua sunt definite ca punctele de extrem ale polinomului lui Cebîşev $T_n(x):\ \cos\frac{k\pi}{n},\ k=0:n$.

Bijecţia între intervalele $[-1,1]$ şi $[l,r]$ este $x\mapsto l+\frac{r-l}{2}(x+1)$.

```
function x=chebpoints(n,varargin)
    l=-1
    r=1
    if length(varargin)>0 then
      if length(varargin)==2 then
        l=varargin(1)
        r=varargin(2)
      else
        error('Wrong number of input parameters')
      end
    end
    k=0:n
    x=l+0.5*(r-l)*(cos(k*%pi/n)+1)
endfunction
```

1.4 Salvarea şi restaurarea datelor

Salvarea şi restaurarea datelor în fişiere text se obţin cu funcţiile *Scilab*:

- `csvWrite(`*M,filename, separator*`)`

- `csvWrite(`*M,filename*`)`

  M este matricea care se salvează iar separatorul implicit este virgula (*csv - Comma separated values*).

- *M*=`csvRead(`*filename*`)`
- *M*=`csvRead(`*filename, separator*`)`

1.5 Grafică în *Scilab*

1.5.1 Grafică 2D

Din mulţimea funcţiilor grafice ale produsului *Scilab* amintim

- `plot2d`. Cea mai simplă formă de utilizare este

$$\texttt{plot2d}(x, y)$$

  unde

$$x = \begin{pmatrix} x_{1,1} & \dots & x_{1,n} \\ \dots & \dots & \dots \\ x_{m,1} & \dots & x_{m,n} \end{pmatrix} \qquad y = \begin{pmatrix} y_{1,1} & \dots & y_{1,n} \\ \dots & \dots & \dots \\ y_{m,1} & \dots & y_{m,n} \end{pmatrix}$$

  sunt două matrice de acelaşi tip. Funcţia construieşte în acelaşi panou (fereastră grafică) graficele a n curbe ce trec respectiv prin punctele

$$(x_{i,j}, y_{i,j})_{1\le i\le m}, \qquad j \in \{1, \dots, n\}.$$

- `fplot2d`. Utilizarea funcţiei este

$$\texttt{fplot2d}(x, f)$$

  unde

  - x este un şir de numere reale;
  - f este o funcţie *Scilab*

  iar rezultatul este graficul funcţiei f construit pe baza valorilor ei în punctele lui x.

- `comet`. Reprezentare grafică animată. Utilizarea este asemănătoare cu cea a fucţiei `plot2d`.

- `paramfplot2d` realizează o animaţie reprezentând evoluţia graficele funcţiilor $f_t(x),\ x \in [a,b]$ când parametrul t parcurge o mulţime de valori. Modul de utilizare

  $$\texttt{paramfplot2d}(f,x,t)$$

  unde

  – $f(x,t)$ este o funcţie *Scilab*.

  – x şi t sunt şiruri.

Exemplul 1.11 *Reprezentarea grafică a funcţiilor* $\sin x$ *şi* $\cos x$ *în* $[-\pi,\pi]$

Reprezentarea grafică se obţine prin

```
t=-%pi:0.1:%pi;
x=[t',t'];
y=[sin(t'),cos(t')];
plot2d(x,y)
```

Va rezulta imaginea din Fig. 1.1

1.5.2 Grafică 3D

- `plot3d(X,Y,Z)`

  $X=(X_i)_{1\le i\le m}$ şi $Y=(Y_{1\le j\le n}$ sunt vectori daţi în ordine crescătoare, iar $Z=(Z_{i,j}),\ i\in\{1,\ldots,m\},\ j\in\{1,\ldots,n\}$ reprezintă valoarea funcţiei calculată în (X_i,Y_j).

  Poziţia observatorului este indicată prin parametrii $theta = longitudine, alpha = 90 - latitudine$. Valorile implicite sunt $theta = 35, alpha = 45$.

  Dacă suprafaţa este definită prin $z=f(x,y),\ x\in[a,b], y\in[c,d]$ atunci se defineşte funcţia *Scilab*

```
function [x,y,z]=suprafata(m,n)
  u=linspace(a,b,m);
  v=linspace(c,d,n);
  for i=1:m
    for j=1:n
      x(i,j)=. . .
```

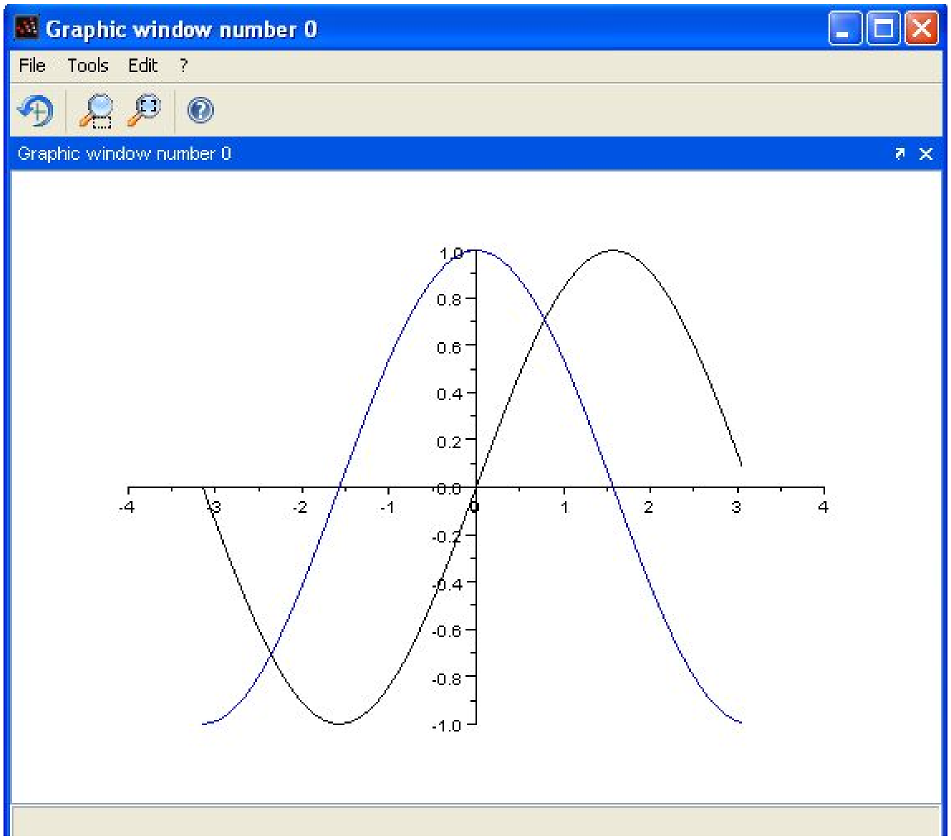

Figure 1.1: Grafică 2D.

```
      y(i,j)=. . .
    end
  end
  z=f(x,y);
endfunction

exec('. . .\suprafata.sce',-1)
[x,y,z]=suprafata(m,n);
plot3d(x,y,z)
```

Se poate proceda mai simplu

```
[x,y]=meshgrid(a:dx:b,c:dy:d);
z=f(x,y);
mesh(x,y,z)
```

- `fplot3d(X,Y,f)`

Exemplul 1.12 *Reprezentarea sferei* $x^2 + y^2 + z^2 = 1$.

Semisfera superioară este dată de $z = \sqrt{1 - x^2 - y^2}$, pentru care construim funcţia *Scilab*

```
function[x,y,z]=sfera(r,m,n)
    x=linspace(-r,r,m)
    y=linspace(-r,r,n)
    for i=1:m
        for j=1:n
            if x(i)^2+y(j)^2<=r^2 then
                z(i,j)=sqrt(r^2-x(i)^2-y(j)^2)
            else
                z(i,j)=%nan
            end
        end
    end
endfunction
```

Imaginea obţinută este dată în Fig. 1.2.
Altfel

```
deff('z=sf(x,y)',['z=sqrt(1-x.^2-y.^2)'])
x=-1:0.05:1;y=x;
fplot3d(x,y,sf)
```

Exerciţii

1. Să se reprezinte grafic funcţiile în intervalele indicate

   a. $\sqrt{|x|}$ $\quad x \in [-4,4]$

   b. e^{-x^2} $\quad x \in [-4,4]$

   c. $\frac{4x\sin x-3}{2+x^2}$ $\quad x \in [0,4]$

   d. $x - \sqrt{x^2-1}$ $\quad x \in [-4,4]$

2. Să se reprezinte evoluţia funcţiilor

   a. x^t $\quad x \in [0,1]$ $\quad t \in \{0,1,2,\dots,500\}$

   b. $\sin tx$ $\quad x \in [0,\pi/2]$ $\quad t \in \{1,2,\dots,100\}$

   c. $\sin tx$ $\quad x \in [0,2\pi]$ $\quad t = 1 - 0.05i,\ i \in \{0,1,\dots,20\}$

3. Să se reprezinte:

   a. $z = \frac{x^2}{a^2} + \frac{y^2}{b^2}$ $\quad a = 4, b = 3$ $\quad x \in [-3,3], y \in [-3,3]$
   Paraboloidul eliptic

   b. $z = \frac{x^2}{a^2} - \frac{y^2}{b^2}$ $\quad a = 4, b = 3$ $\quad x \in [-3,3], y \in [-3,3]$
   Paraboloidul hiperbolic

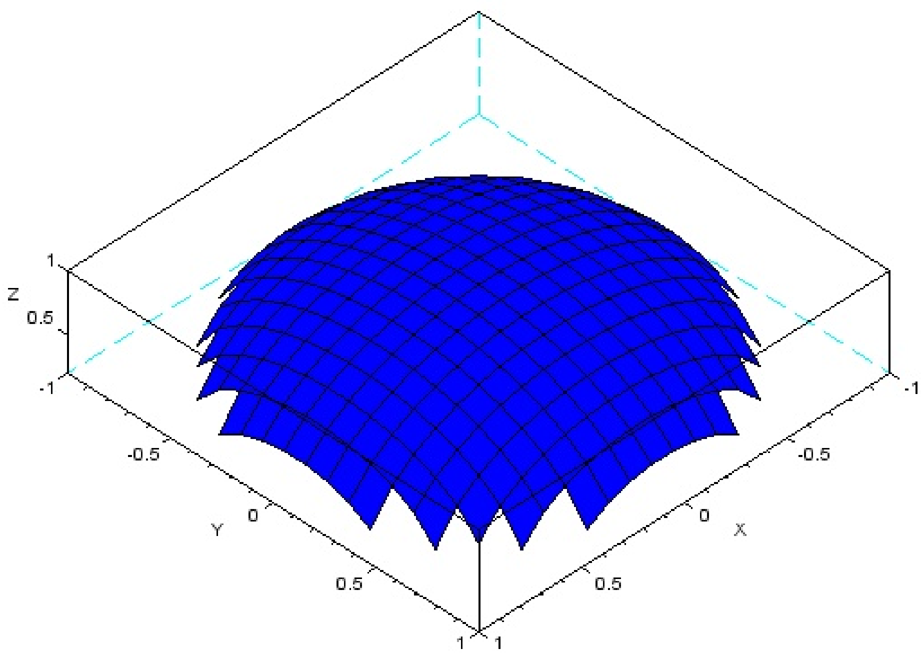

Figure 1.2: Grafică 3D Semisfera superioară

Capitolul 2

Algebră liniară numerică

Cap. 1

Cap. 3

2.1 Factorizarea unei matrice

Funcţiile

- $[L, U, P] = \mathrm{lu}(A)$ calculează factorizarea LU a matricei $A: PA = LU$.
- $[Q, R] = \mathrm{qr}(A)$ calculează factorizarea QR a matricei $A:\ A = QR$

Exemplul 2.1 *Să se calculeze factorizarea LU a matricei*

$$A = \begin{pmatrix} 1 & 2 & -1 & 3 & 2 \\ 2 & 4 & -2 & 5 & 1 \\ -1 & -2 & 1 & -3 & -4 \\ 3 & 6 & 2 & 10 & 7 \\ 1 & 2 & 4 & 0 & 4 \end{pmatrix}.$$

```
A=[1,2,-1,3,2;2,4,-2,5,1;-1,-2,1,-3,-4;3,6,2,10,7;1,2,4,0,4];
[L,U,P]=lu(A)
 P  =
```

```
   0.    0.    0.    1.    0.
   0.    1.    0.    0.    0.
   0.    0.    0.    1.    0.
   1.    0.    0.    0.    0.
   0.    0.    1.    0.    0.
 U  =

   3.    6.    2.            10.          7.
   0.    0.  - 3.3333333  - 1.6666667  - 3.6666667
   0.    0.    3.3333333  - 3.3333333    1.6666667
   0.    0.    0.         - 2.           0.5
   0.    0.    0.           0.         - 2.
 L  =

   1.           0.    0.      0.    0.
   0.6666667    1.    0.      0.    0.
   0.3333333    0.    1.      0.    0.
   0.3333333    0.  - 0.5     1.    0.
 - 0.3333333    0.    0.5   - 1.    1.
```

Exemplul 2.2 *Să se calculeze factorizarea QR a matricei*

$$X = \begin{pmatrix} 6 & 6 & 1 \\ 3 & 6 & 1 \\ 2 & 1 & 1 \end{pmatrix}$$

```
X=[6,6,1;3,6,1;2,1,1];

[Q,R]=qr(X)
 R  =

   7.    8.    1.5714286
   0.    3.    0.1428571
   0.    0.    0.7142857
 Q  =

   0.8571429  - 0.2857143  - 0.4285714
   0.4285714    0.8571429    0.2857143
   0.2857143  - 0.4285714    0.8571429
```

```
Q*R-X

    0.           8.882D-16    4.441D-16
  - 4.441D-16  - 5.329D-15  - 1.554D-15
    0.         - 3.109D-15  - 8.882D-16
```

2.2 Rezolvarea sistemelor algebrice de ecuaţii liniare

Pentru rezolvarea unui sistem algebric de ecuaţii liniare $Ax = b$, în cazul în care

- numărul ecuaţiilor coincide cu numărul necunoscutelor;
- determinantul sistemului este diferit de zero;

se procedează după cum urmează:

1. se fixează matricele A şi b;
2. se calculează $x = A^{-1}b$.

Expresia A^{-1} se poate introduce întocmai `(A^(-1))` sau se poate utiliza funcţia *Scilab* `inv(`A`)`.

Funcţia *Scilab* `det(`A`)` calculează determinantul matricei A.

Exemplul 2.3 *Să se rezolve sistemul algebric de ecuaţii liniare*

$$\left\{\begin{array}{ccccccccc} x_1 & + & 2x_2 & + & 3x_3 & + & 4x_4 & = & 11 \\ 2x_1 & + & 3x_2 & + & 4x_3 & + & x_4 & = & 12 \\ 3x_1 & + & 4x_2 & + & x_3 & + & 2x_4 & = & 13 \\ 4x_1 & + & x_2 & + & 2x_3 & + & 3x_4 & = & 14 \end{array}\right.$$

Rezolvarea constă din

```
A = [1,2,3,4;2,3,4,1;3,4,1,2;4,1,2,3];
b = [11;12;13;14];
det(A)
ans=
    160
x=inv(A)*b
```

```
x=
! 2 !
! 1 !
! 1 !
! 1 !
```

În general, pentru sistemul algebric de ecuaţii liniare $Ax+b=0$, funcţia *Scilab* `[x,k]=linsolve(A,b)` calculează

1. x – o soluţie particulară;
2. k – o bază a subspaţiului liniar $\mathrm{Ker}(A)=\{x|Ax=0\}$.

Soluţia sistemului $Ax+b=0$ este $x+kc$ unde c este un vector oarecare a cărui dimensiune coincide cu numărul coloanelor matricei k.

Dacă sistemul este incompatibil atunci rezultatele sunt egale cu matricea vidă [ ].

Exemplul 2.4 *Să se rezolve sistemul algebric de ecuaţii liniare*

$$\left\{\begin{array}{rcrcrcrcr} x_1 & + & x_2 & + & x_3 & + & x_4 & = & 2 \\ 2x_1 & - & x_2 & + & 2x_3 & - & x_4 & = & 1 \\ x_1 & + & 2x_2 & - & x_3 & + & 2x_4 & = & -1 \\ 2x_1 & + & x_2 & + & 4x_3 & + & x_4 & = & 7 \\ 3x_1 & + & 2x_2 & - & 2x_3 & + & 2x_4 & = & -5 \end{array}\right.$$

având soluţia $x_1=-1,\ x_2=1-x_4,\ x_3=2.$

Scris matriceal, soluţia sistemului este

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 2 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} c = \begin{pmatrix} -1 \\ 1 \\ 2 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{-1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix} c'.$$

Rezolvarea sistemului este

```
A=[1,1,1,1;2,-1,2,-1;1,2,-1,2;2,1,4,1;3,2,-2,2];
b=[2;1;-1;7;-5] ;
[x,k]=linsolve(A,-b)
k  =

  - 5.551D-17
```

```
    0.7071068
    2.776D-16
  - 0.7071068
 x  =

  - 1.
    0.5
    2.
    0.5
```

Observaţie. k aproximează vectorul de normă euclidiană 1

$$\begin{pmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ 0 \\ \frac{-1}{\sqrt{2}} \end{pmatrix}.$$

În cazul unui sistem algebric de ecuaţii liniare incompatibil se poate calcula elementul care minimizează funcţionala $J(x) = \|b - Ax\|_2^2$. Acest element este soluţia sistemului algebric de ecuaţii liniare compatibil $A^T Ax = A^T b$ şi este dat de funcţia *Scilab* `x=lsq(`A, b`)`.

Exemplul 2.5 *Să se rezolve sistemul algebric de ecuaţii liniare*

$$\left\{ \begin{array}{ccccccc} 2x & - & y & + & 3z & = & 7 \\ x & + & y & + & z & = & 4 \\ 3x & - & 3y & + & 5z & = & 8 \end{array} \right.$$

în sensul celor mai mici pătrate.

Rezolvarea este

```
A=[2,-1,3;1,1,1;3,-3,5];
b=[7;4;8];
x=lsq(A,b)
 x  =

    1.4871795
    1.2948718
    1.5512821
```

Funcţia `linsolve(`$A, -b$`)` returnează `[ ]`.

Capitolul 3

Rezolvarea sistemelor şi ecuaţiilor algebrice

⟵ Cap. 2 ⟶ Cap. 4

3.1 Rezolvarea sistemelor algebrice de ecuaţii neliniare

Pentru rezolvarea sistemului algebric de ecuaţii neliniare

$$\left\{\begin{array}{lcl} f_1(x_1,\ldots,x_n) & = & 0 \\ \vdots & & \\ f_n(x_1,\ldots,x_n) & = & 0 \end{array}\right.$$

sau scris sub formă concentrată $f(x)=0$ cu

$$x=\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \qquad f(x)=\begin{pmatrix} f_1(x_1,\ldots,x_n) \\ \vdots \\ f_n(x_1,\ldots,x_n) \end{pmatrix},$$

se utilizează funcţia *Scilab*

```
[x [,y [,info ] ] ]=fsolve(x0, f [,fjac [,tol ] ] )
```

unde semnificaţia parametrilor este:

- $x0$ reprezintă o aproximaţie iniţială a soluţiei sistemului.
- f identificatorul funcţiei *Scilab* care defineţte sistemul neliniar $f(x) = 0$.
- $fjac$ identificatorul funcţiei *Scilab* al jacobianului fincţiei f, adică

$$fjac(x) = \left(\begin{array}{ccc} \frac{\partial f_1}{\partial x_1}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1}(x) & \dots & \frac{\partial f_n}{\partial x_n}(x) \end{array} \right)$$

- tol toleranţă, parametrul utilizat în testele de precizie (scalar real).
- x aproximaţia calculată a soluţiei sistemului algebric.
- y reprezintă valoarea funcţiei f calculată în x; $y = f(x)$.
- $info$ indicatorul de răspuns al programului `fsolve`. În cazul rezolvării cu succes, valoarea indicatorului este 1.

Astfel, rezolvarea constă din

1. definirea funcţei f (şi eventual al jacobianului $fjac$);
2. apelarea funcţiei *Scilab* `fsolve`.

Exemplul 3.1 *Să se rezolve sistemul algebric de ecuaţii neliniare*

$$\left\{ \begin{array}{ccccccccc} 10x_1 & + & x_1^2 & - & 2x_2x_3 & - & 0.1 & = & 0 \\ 10x_2 & - & x_2^2 & + & 3x_1x_3 & + & 0.2 & = & 0 \\ 10x_3 & + & x_3^2 & + & 2x_1x_2 & - & 0.3 & = & 0 \end{array} \right.$$

Rezolvarea constă din:

```
deff('q=f(p)',['x=p(1)','y=p(2)','z=p(3)',
       'q(1)=10*x+x.^2-2*y.*z-0.1',
       'q(2)=10*y-y.^2+3*x.*z+0.2',
       'q(3)=10*z+z.^2+2*x.*y-0.3'])
p0=[0;0;0];
[p,q,info]=fsolve(p0,f)
```

```
info=
   1.
q=
   1.0E-15*
!  -  .2636780  !
!     .2775558  !
!     .9436896  !
p=
!     .0098702  !
!  -  .0200485  !
!     .0299499  !
```

3.2 Rezolvarea ecuaţiilor algebrice

Dacă $n = 1$, adică dimensiunea spaţiului este 1, atunci $f(x) = 0$ cu $f : I \subset R \to R$ reprezintă o ecuaţie algebrică.

Pentru rezolvarea ecuaţiilor algebrice se utilizează de asemenea funcţia *Scilab* `fsolve`.

Exemplul 3.2 *Să se rezolve ecuaţia* $2^x = x^2$.

Rezolvarea ecuaţiei $f(x) = 2^x - x^2 = 0$ este

```
deff('y=f(x)','y=2.^x-x.^2')
[x,y,info]=fsolve(1,f)
info
  1.
y=
  0.
x=
  2.
```

sau cu precizarea derivatei funcţiei $f(x)$

```
deff('y=df(x)','y=(2.^ x).*log(x)-2*x')
[x,y,info]=fsolve(1,f,df)
info=
  1.
y=
```

```
   0
x=
   2.
```

Dacă $x0 = 5$ atunci $x = 4$, iar pentru $x0 = -1$ găsim $x = -0.7666647$.
Graficul funcţiei f este reprezentat în Fig. 3.1.

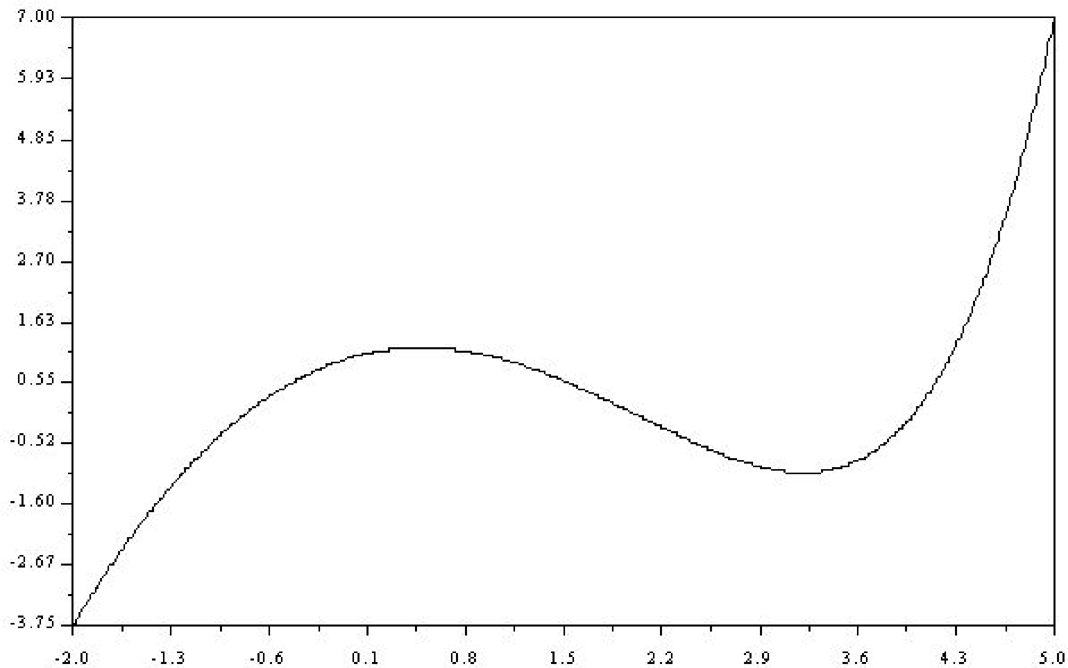

Figure 3.1: $y = 2^x - x^2$

3.3 Rezolvarea ecuaţiilor polinomiale

În cazul particular al ecuaţiilor polinomiale se cere determinarea tuturor rădăcinilor reale sau complexe.

Pentru aflarea rădăcinilor unui polinom se procedează astfel:

1. Se defineşte polinomul *Scilab* p de variabilă x.

2. Rădăcinile polinomului p sunt calcutate ajutorul funcţiei *Scilab*

```
x=roots(p)
```

unde

- p este polinomul *Scilab* cu coeficienţi reali sau complecşi de grad cel mult 100.
- x este un tablou cu rădăcinile polinomului p.

Exemplul 3.3 *Să se determine rădăcinile polinomului* $p = x^4 + 2x^3 + 3x^2 + 2x + 1$.

Rezolvarea constă din:

```
c=[1,2,3,2,1];
p=poly(c,'x','coeff');
x=roots(p)
x=
! - .5 + .8660254i !
! - .5 - .8660254i !
! - .5 + .8660254i !
! - .5 - .8660254i !
```

Capitolul 4

Rezolvarea problemelor de interpolare

Cap. 3 Cap. 5

Fie $\mathscr{F}$ o familie interpolatoare de ordin n pe axa reală. Dându-se nodurile $(x_i)_{1\le i\le n}$ şi numerele $(y_i)_{1\le i\le n}$, dacă $\varphi\in\mathscr{F}$ este funcţia de interpolare care satisface condiţiile $\varphi(x_i)=y_i,\ 1\le i\le n$, se cere să se calculeze $\varphi(z)$, unde z este un punct dat.

4.1 Interpolare polinomială

Pentru $\mathscr{F}=P_{n-1}$ soluţia problemei de interpolare Lagrange este polinomul

$$L(P_{n-1};x_1,\dots,x_n;y_1,\dots,y_n)(z)=\sum_{i=1}^{n} y_i \prod_{\substack{j=1\\ j\neq i}}^{n} \frac{z-x_j}{x_i-x_j}$$

Valoarea acestui polinom este calculat de funcţia *Scilab* (formula baricentrică)

```
function f=lagrange(xd,x,y)
  [mx,nx]=size(x);
  [my,ny]=size(y);
```

```
ierror=0;
if (nx~=ny)|(mx~=1)|(my~=1),
  ierror=1;
  disp('data dimension error')
  abort
end
xx=gsort(x);
for k=1:nx-1,
  if xx(k)==xx(k+1),
    ierror=1;
    break,
  end
end
if ierror~=0,
  disp('data error')
  abort
end
[m,n]=size(xd);
f=zeros(m,n);
p=zeros(m,n);
q=zeros(m,n);
w=ones(1,nx);
for i=1:nx,
  for j=1:nx,
    if i~=j,
      w(i)=w(i)*(x(i)-x(j)),
    end
  end
end
for i=1:m,
  for j=1:n,
    u=find(x==xd(i,j));
    if ~isempty(u),
      f(i,j)=y(u);
    else
      for k=1:nx,
        p(i,j)=p(i,j)+y(k)/(xd(i,j)-x(k))/w(k);
        q(i,j)=q(i,j)+1/(xd(i,j)-x(k))/w(k);
      end
      f(i,j)=p(i,j)/q(i,j);
    end
  end
end
endfunction
```

Semnificaţiile parametrilor formali şi a rezultatului sunt:

- xd este o matrice de numere. În fiecare element al matricei xd se calculează valoarea polinomului de interpolare Lagrange.
- $x = (x_i)_{1\le i\le n}$, şirul nodurilor de interpolare;
- $y = (x_i)_{1\le i\le n}$, şirul valorilor interpolate;
- f este o matrice de aceleaşi dimensiuni ca xd şi are ca elemente valorile

polinomului de interpolare Lagrange calculate în elementele corespunzătoare ale matricei xd.

Programul de mai sus, pentru calculul valorii polinomului de interpolare Lagrange, foloseşte formula baricentrică.

Exemplul 4.1 *Să se calculeze $L(P_n; x_0, \ldots, x_n; f)(z)$ pentru*

$$\begin{array}{lcl} f(x) & = & x^2 \\ x_i & = & i, \quad i \in \{0, 1, \ldots, 5\} \\ z & = & 0.5, 3, 5, 10 \end{array}$$

Rezolvarea este:

```
deff('y=fct(x)','y=x.^2')
exec('. . .\lagrange.sci',-1)
x=0:5;
y=fct(x);
z=[0.5,3.5,10];
f=lagrange(z,x,y)
! 0.25   12.25   100  !
```

Facilităţile de programare cu polinoame din *Scilab* şi formula de recurenţă permite calculul simbolic al polinomului de interpolare Lagrange:

```
function lag=polyLagrange(x,y,s)
    z=poly(0,s);
    [k,n]=size(x);
    [k,m]=size(y);
    if m~=n then
        lag="Data error"
        return
    end
    v=zeros(1,n);
    w=zeros(1,n);
    for i=1:n do
        v(i)=y(i);
    end
    for k=1:n-1 do
        for i=1:n-k do
            w(i)=((z-x(i))*v(i+1)-(z-x(i+k))*v(i))/(x(i+k)-x(i));
        end
        for i=1:n-k do
            v(i)=w(i);
        end
    end
    lag=v(1);
endfunction
```

Pentru exemplul anterior calculul este

```
deff('y=fct(x)','y=x.^2')
exec('. . .\polyLagrange.sci',-1)
x=0:5;
y=fct(x);
polyLagrange(x,y,'x')
```

x^2

4.2 Interpolare cu funcţii spline cubice

Dacă $\mathscr{F} = \mathscr{S}_3$, mulţimea funcţiilor spline cubice atunci pentru rezolvarea problemei de interpolare utilizăm funcţiile *Scilab*

1. `d=splin(x,y);`
   sau
   `d=splin(x,y,[,spline_type[,der]])`[1]

2. `[f0 [,f1 [,f2 [,f3]]]]=interp(xd,x,y,d).`

Semnificaţia parametrilor este

- $x = (x_i)_{1 \le i \le n}$, şirul nodurilor de interpolare;
- $y = (x_i)_{1 \le i \le n}$, şirul valorilor interpolate;
- *spline_type* este un string care precizează condiţiile la limită utilizate şi poate fi:
  - `not_a_knot` - alegerea implicită: Dacă $x_1 < x_2 < \ldots < x_{n-1} < x_n$ atunci condiţiile la limită sunt

$$\begin{aligned} s^{(3)}(x_2 - 0) &= s^{(3)}(x_2 + 0) \\ s^{(3)}(x_{n-1} - 0) &= s^{(3)}(x_{n-1} + 0) \end{aligned}$$

  - `clamped`:

$$\begin{aligned} s'(x_1) &= der_1 \\ s'(x_n) &= der_2 \end{aligned}$$

[1] Parantezele pătrate nu fac parte din sintaxă. Ele arată caracterul opţional al elementelor cuprinse între ele.

– `natural`:

$$\begin{aligned} s''(x_1) &= 0 \\ s''(x_n) &= 0 \end{aligned}$$

– `periodic`: Dacă $y_1 = y_n$, condiţiile la limită sunt

$$\begin{aligned} s'(x_1) &= s'(x_n) \\ s''(x_1) &= s''(x_n) \end{aligned}$$

- d parametri funcţiei spline cubice de interpolare;
- xd este o matrice de numere. În fiecare element al matricei xd se calculează valoarea funcţiei spline cubice de interpolare. Aceste elemente trebuie să fie cuprinse în intervalul determinat de nodurile din x.
- $f0, f1, f2, f3$ sunt matrice de aceleaşi dimensiuni ca xd şi au ca elemente valorile funcţiei spline cubice de interpolare şi respectiv a derivatelor de ordinul 1, 2 şi 3, calculate în elementele corespunzătoare ale matricei xd.

Exemplul 4.2 *Să se calculeze valorile funcţiei spline cubice de interpolare şi ale derivatelor sale de ordin 1, 2, 3 pentru datele de interpolare*

$$\begin{aligned} f(x) &= x^3 \\ x_i &= i, \quad i \in \{0, 1, \ldots, 5\} \\ z &= 0.5, 3, 5, 10 \end{aligned}$$

Rezolvarea constă din

```
deff('y=f(x)','y=x^3')
x=0:5;
y=f(x);
z=[0.5,3.5,10];
d=splin(x,y);
[s,s1,s2,s3]=interp(z,x,y,d)
s3=
!  6. !
!  6. !
!  0. !
s2=
```

```
!  3.  !
!  21. !
!  0.  !
s1=
!   0.75 !
!  36.75 !
!  0.    !
s=
! .125   42.875    0  !
```

Capitolul 5

Derivare numerică

⟵ Cap. 4 ⟶ Cap. 6

5.1 Derivarea funcţiilor de o variabilă reală

Pentru calculul derivatelor de ordinul 1,2,4 ale unei funcţii într-un punct şi ale gradientului sau hessianului unei funcţii de mai multe variabile *Scilab* oferă funcţia :

$$\begin{aligned} g &= \text{numderivative}(f, x) \\ g &= \text{numderivative}(f, x, h) \\ g &= \text{numderivative}(f, x, h, order) \\ [J, H] &= \text{numderivative}(\ldots) \end{aligned}$$

unde

- f funcţia ale cărei derivate se cer calculate;
- x punctul în care se vor calcula derivatele;

- h pasul în formula de integrare numerică (opţional, h este matrice de ordin $dim(x) \times 1$);
- $order \in \{1,2,4\}$ ordinul derivatelor care se vor calcula (opţional, valoarea implicită este 2).

Exemplul 5.1 *Să se calculeze $f'(1)$ şi $f''(1)$ pentru $f(x) = x^3$.*

Rezolvarea este:

```
deff('y=f(x)','y=x^3')
numderivative(f,1)
 ans  =
    3.0000001
[d1,d2]=numderivative(f,1)
d2 =
   6
d1 =
   3
```

5.2 Cazul funcţiilor de mai multe variabile

Exemplul 5.2 *Să se calculeze jacobianul şi hessianul functiei $f(x,y) = \begin{pmatrix} x^2y \\ x^3+y \end{pmatrix}$ în $(1,2)$.*

Jacobianul şi hessianul funcţiei f sunt

$$f'(x,y) = \begin{pmatrix} 2xy & x^2 \\ 3x^2 & 1 \end{pmatrix}$$

$$f''(x,y) = \left(\begin{array}{cc|cc} 2y & 2x & 2x & 0 \\ 6x & 0 & 0 & 0 \end{array}\right)$$

Astfel rezultatele exacte sunt

$$f'(1,2) = \begin{pmatrix} 4 & 1 \\ 3 & 1 \end{pmatrix}$$

$$f''(1,2) = \left(\begin{array}{cc|cc} 4 & 2 & 2 & 0 \\ 6 & 0 & 0 & 0 \end{array}\right)$$

Rezolvarea *Scilab* este

```
deff('q=fct(p)',['x=p(1)','y=p(2)','q(1)=x.^2.*y','q(2)=x.^3+y'])
p=[1;2]
[J,H]=numderivative(fct,p)
 H  =

    4.    2.    2.    0.
    6.    0.    0.    0.
 J  =

    4.    1.
    3.    1.
```

Capitolul 6

Construirea unei funcţii de aproximare prin metoda celor mai mici pătrate

Cap. 5

Cap. 7

Cazul liniar. Determinarea unui polinom de aproximare construit prin metoda celor mai mici pătrate de grad m pentru datele $(x_i, y_i)_{1\le i\le n}$ $(m << n)$ se obţine cu ajutorul funcţiei

$$coef\texttt{=lq(}m,x,y\texttt{)}$$

unde

- $coef$ – sunt coeficienţii polinomului de aproximare;
- m – gradul polinomului de aproximare;
- x, y – doi vectori linie cu absisele şi respectiv, cu ordonatele datelor problemei de aproximare.

Textul sursă al funcţiei `lq` este

```
function coef=lq(m,x,y)
  [mx,nx]=size(x);
  [my,ny]=size(y);
  if ((nx~=ny)|(mx~=1)|(my~=1)),
    disp("data dimension error")
    abort
  end
  u=zeros(m+1,nx);
  for i=1:m+1,
    for j=1:nx,
      u(i,j)=x(j)^(i-1);
    end
  end
  coef=(u*u')^(-1)*u*y';
endfunction
```

Exemplul 6.1 *Să se calculeze polinoamele de aproximare de grad unu şi doi, constituite prin metoda celor mai mici pătrate, pentru datele* $(x_i, y_i)_{0\le i\le 20}$ *unde* $x_i = -2+0.2i, \quad y_i = f(x_i), \quad f(x) = |x|$. *Să se reprezinte grafic funcţiile astfel obţinute.*

```
x=-2:0.2:2;
y=abs(x);
exec('e:\scilab\lq.sci',-1)
c1=lq(1,x,y)
c1=
   1.047619
   1.110D-16
c2=lq(2,x,y)
c2=
   0.3883622
   1.110D-16
   0.4494933
//
// Reprezentarea grafica
//
deff('y=p1(x)',['y=c1(1)+c1(2)*x'])
deff('y=p2(x)',['y=c2(1)+c2(2)*x+c2(3)*x.*x'])
y1=p1(x);
y2=p2(x);
xx=[x',x',x'];
yy=[y',y1',y2'];
plot2d(xx,yy,[1,2,3],'121','abs@p1@p2')
```

Astfel, polinoamele de aproximare de gradul întâi şi doi sunt

$$p_1(x) = 1.047619$$

şi respectiv

$$p_2(x) = 0.3883622 + 0.4494933x^2$$

Graficele celor trei funcţii sunt date Fig. 6.1.

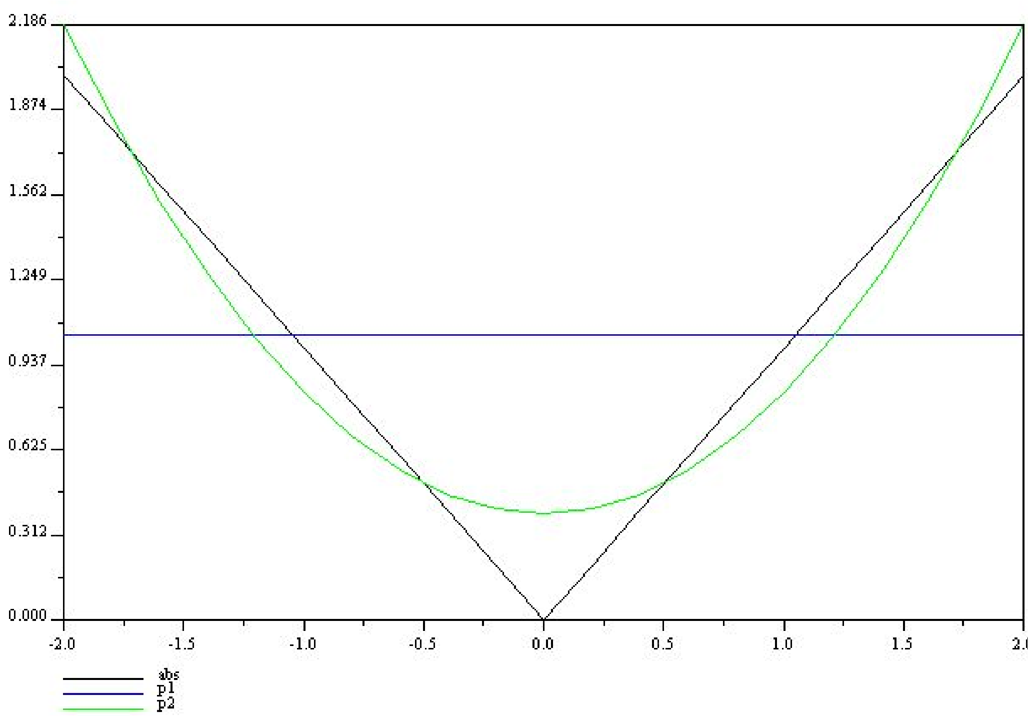

Figure 6.1: Aproximarea funcţiei x^2 prin polinom de grad 1 şi 2.

Cazul neliniar.

Pentru determinarea funcţiei de aproximare $y = \varphi(t, x_1, \ldots, x_n)$ care minimizează expresia

$$\Phi(x_1, \ldots, x_n) = \sum_{i=1}^{m} [\varphi(t_i, x_1, \ldots, x_n) - y_i]^2 \tag{6.1}$$

prezentăm două abordări prezente în *Scilab*.

1. **Metoda Gauss-Newton.** Se utilizează funcţia *Scilab* `leastsq`:

   `[`*fopt* `[,`*xopt* `[,`*gopt* `]]]=leqstsq(`*fct,x0*`)`
   `[`*fopt* `[,`*xopt* `[,`*gopt* `]]]=leqstsq(`*fct,dfct,x0*`)`

   unde

   - $x0$ este aproximaţia iniţială a parametrilor;
   - fct defineşte termenii din suma (6.1);
   - $dfct$ este jacobianul funcţiei fct;
   - $fopt$ valoarea funcţiei Φ calculată în $xopt$;
   - $xopt$ valoarea optimă găsită a parametrilor x;
   - $gopt$ gradientul funcţiei Φ calculată în $xopt$.

Exemplul 6.2 *Să se calculeze funcţia de aproximare de forma* $y = ae^{bt}$ *în cazul datelor*

```
t=0:5;
y=2*exp(-t);
```

În fişierul *fct.sce* se definesc funcţiile $fct(x,t,y)$ şi $f1(t,x)$. Funcţia $f1$ fixează expresia funcţiei de aproximare $\varphi(t,x_1,\ldots,x_n)$ iar funcţia fct calculează tabloul erorilor corespunzătoare datelor problemei.

Necunoscutele a,b devin componentele vectorului x ($a = x_1, b = x_2$).

Fişierul *fct.sce* este

```
function u=fct(x,t,y)
  u=f1(x,t)-y;
endfunction

function y=f1(x,t)
  y=x(1)*exp(x(2)*t);
endfunction
```

Rezolvarea este

```
exec('. . .\fct.sce',-1)
x=[2.5,-0.5];
[fopt,xopt,gopt]=leastsq(list(fct,t',y'),x)
 gopt  =
```

```
   0.    0.
xopt  =

   2.  - 1.
fopt  =

   0.
```

Dacă, în plus, se furnizează gradientul funcţiei $f1$: $df1 = (\frac{\partial f1}{\partial a}, \frac{\partial f1}{\partial b}) = (\frac{\partial f1}{\partial x_1}, \frac{\partial f1}{\partial x_2}) = (e^{bt}, ate^{bt})$ în fişierul *fct1.sce*, codul acestuia va fi

```
function u=fct1(x,t,y)
  u=f1(x,t)-y;
endfunction

function y=f1(x,t)
  y=x(1)*exp(x(2)*t);
endfunction

function u=df1(x,t,y)
  s=exp(x(2)*t);
  u=[s,x(1).*t.*s];
endfunction
```

atunci apelarea va fi

```
[fopt,xopt,gopt]=leastsq(list(fct1,t',y'),df1,x)
```

Deşi în formulele derivatelor parţiale variabila y nu apare, codul *Scilab* solicită includerea unui al treilea argument, y în cazul de faţă.

2. **Metoda Levenberg-Marquardt** este implementată de funcţia *scilab lsqrsolve:*

   [*xopt* [, *v*]]=lsqrsolve(*x0,fct,m*)
   [*xopt* [, *v*]]=lsqrsolve(*x0,fct,m,dcft*)

   unde

   - $x0$ este aproximaţia iniţială a parametrilor;
   - fct defineşte termenii din suma (6.1);
   - m reprezintă numărul termenilor din suma (6.1);
   - $dfct$ este jacobianul funcţiei fct;

- $xopt$ valoarea optimă găsită a parametrilor x;
- v valoarea fiecărui termen din suma (6.1) calculată în $xopt$.

Rezolvarea exemplului anterior cu metoda Levenberg-Marquardt constă din definirea funcţiilor *f1.sce, data.sce şi fctlm.sce* în fişierul *fctlm.sce:*

```
function u=fctlm(x,m)
  [t,y]=data(m);
  u=f1(x,t)-y;
endfunction

function y=f1(x,t)
  y=x(1)*exp(x(2)*t);
endfunction

function [t,y]=data(m)
  t=0:m-1;
  y=2*exp(-t);
endfunction
```

Funcţia *data* furnizează datele problemei. $f1$ fixează forma funcţiei de aproximare, ale cărui coeficienţi se caută prin metoda celor mai mici pătrate. Funcţia $fctlm(x,m)$ returnează tabloul erorilor cu

- x aproximaţia iniţială a coeficienţilor;
- m numărul datelor problemei.

Rezolvarea constă din

```
x=[2.5,-0.5];
exec('. . .\fctlm.sce',-1)
[xopt,v]=lsqrsolve(x,fctlm,6)
 v  =

 0
 0
 0
 0
 0
 0
 xopt  =

    2.  - 1.
```

Capitolul 7

Integrare numerică

⟵ Cap. 6 ⟶ ??

7.1 Integrarea numerică a funcţiilor de o variabilă reală

Pentru calculul integralei

$$I = \int_a^b f(x)\mathrm{d}x$$

unde $f : [a, b] \to R$ este o funcţie continuă, *Scilab* oferă mai multe programe:

- [x]=`integrate`(*exp, var, a, b* [, *ea* [,*er*]])
  unde:
  - x valuarea calculată a integralei;
  - *exp* este expresia funcţiei de integrat, dat ca un şir de caractere;
  - *var* este variabila de integrare dat ca string;
  - *a* extremitatea stângă a intervalului de integrare;
  - *b* extremitatea dreaptă a intervalului de integrare;

  - *ea, er* reprezintă o eroare absolută şi o eroare relativă. Precizând aceşti parametri, regula de oprire a programului este $|x-I| \leq \max\{ea, er\cdot|I|\}$.

Exemplul 7.1 *Să se calculeze integralele:*

$$1. \qquad \int_0^1 16 * x^{15}\mathrm{d}x$$

$$2. \qquad \int_0^{\frac{\pi}{2}} \frac{\sin(x)}{x}\mathrm{d}x$$

Rezolvările sunt:
1.

```
I=integrate('16*x.^15','x',0,1)
I=
        1.
```

2.

```
I=integrate('if x==0 then 1; else sin(x)/x; end','x',0,%pi/2)
I=
        1.3707622
```

- $[x, err]$= `intg`(a, b, f [,*ea* [, *er*]])
unde
  - x valuarea calculată a integralei;
  - err valoarea estimată a erorii absolute;
  - a extremitatea stângă a intervalului de integrare;
  - b extremitatea dreaptă a intervalului de integrare;
  - f identificatorul funcţiei *Scilab* de integrat;
  - *ea, er* reprezintă o eroare absolută şi o eroare relativă. Precizând aceşti parametri, regula de oprire a programului este $|x-I| \leq \max\{ea, er\cdot|I|\}$.

Utilizând `intg`, integralele exemplului anterior se calculează astfel:

1.

```
deff('y=f(x)','y=16*x.^15')
[x,err]=intg(a,b,f)
err=
        1.110 E-14
x=
        1.
```

2.

```
deff('y=g(x)','if x==0 then y=1; else y=sin(x)/x; end')
[x,err]=intg(0,%pi/2,g)
err=
        1.522 E-14
x=
        1.3707622
```

Dacă $f(x, p_1, p_2, \ldots)$ este o funcţie *Scilab* având pe prima poziţie variabila de integrare atunci

$$\texttt{intg}(a,b,\texttt{list}(f, p_1, p_2, \ldots))$$

calculează integrala

$$\int_a^b f(x, p_1, p_2, \ldots)\mathrm{d}x$$

7.2 Calculul numeric al integralelor duble

Pentru domeniul D definit prin

$$D = \{(x, y) : a \le x \le b,\ finf(x) \le y \le fsup(x)\}$$

integrala

$$\iint_D fct(x, y)\mathrm{d}x\mathrm{d}y$$

este calculată de programul $[integ, er]$=integr$(fct, a, b, finf, fsup, tol)$. Semnificaţia parametrilor formali este:

- fct funcţia de integrat;
- $a, b, finf, fsup$ datele domeniului de integrare;

- *tol* toleranţa utilizată în regula de oprire;
- *integ* – valoarea calculată a integralei duble;
- *er* – indicatorul de răspuns:
  - 0 – integrala s-a calculat cu succes;
  - 1 – nu s-a îndeplinit condiţia de convergenţă în 10 iteraţii.

Textele sursă ale funcţiilor sunt

1.
```
function [integ,er]=integr(fct,a,b,finf,fsup,tol)
  n=3
  u=init(n)
  old=integ2(u)
  nmi=10
  cont=0
  ni=0
  while cont==0 do
      ni=ni+1
      v=tab(u)
      new=integ2(v)
      nrm=abs(new-old)
      old=new
      u=v
      if nrm<tol then
          cont=1
          er=0;
      else
          er=1
          if ni==nmi,cont=1;end
      end
  end
  integ=new
endfunction
```

2. Calculul tabelului cu valorile funcţiei de integrat pentru valoarea iniţială a parametrului de discretizare:

```
function u=init(n)
  hx=(b-a)/n
  u=zeros(n+1,n+1)
  for j=1:n+1,
      x=a+(j-1)*hx
      fi=finf(x)
      fs=fsup(x)
      hy=(fs-fi)/n
      for i=1:n+1,
          y=fi+(i-1)*hy
          u(i,j)=fct(x,y)
      end
  end
endfunction
```

3. Extinderea tabelului cu valorile funcţiei de integrat pentru un nouă valoare a parametrului de discretizare:

```
function v=tab(u)
  [m,n]=size(u)
  hx=(b-a)/(n-1)
  for j=1:n,
      for i=1:m,
          v(2*i-1,2*j-1)=u(i,j)
      end
      x=a+(j-1)*hx
      fi=finf(x)
      fs=fsup(x)
      hy=(fs-fi)/(m-1)
      for i=1:m-1,
          y=fi+hy*(i-0.5)
          v(2*i,2*j-1)=fct(x,y)
      end
  end
  for j=1:m-1,
      x=a+(j-0.5)*hx
      fi=finf(x)
      fs=fsup(x)
      hy=(fs-fi)/(m-1)/2
      for i=1:2*n-1,
          y=fi+hy*(i-1)
          v(i,2*j)=fct(x,y)
      end
  end
endfunction
```

4. Calculul integralei duble pentru o valoare a parametrului de discretizare:

```
function z=integ2(u)
  [m,n]=size(u)
  hx=(b-a)/(n-1)
  w=zeros(1,n)
  c=ones(1,n)
  if n==3,
      c(2)=4
  else
      n0=(n-1)/2
      for j=1:n0,
          c(2*j)=4
      end
      for j=1:n0-1,
          c(2*j+1)=2
      end
  end
  for j=1:n,
      x=a+(j-1)*hx
      fi=finf(x)
      fs=fsup(x)
      hy=(fs-fi)/(m-1)
      w(j)=c*u(:,j)*hy/3
  end
  z=c*w'*hx/3
endfunction
```

5. Datele problemei:

```
function [fct,a,b,finf,fsup,tol]=datas()
  a=. . .;
  b=. . .;
  deff('z=fct(x,y)','z=. . .');
  deff('y=finf(x)','y=. . .');
  deff('y=fsup(x)','y=. . .');
  tol=. . .;
endfunction
```

6. Apelarea aplicaţiei:

```
function aplicatia(cale)
    exec(cale+'\datas.sci',-1)
    exec(cale+'\integr.sci',-1)
    [fct,a,b,finf,fsup,tol]=datas();
    [integ,er]=integr(fct,a,b,finf,fsup,tol);
    printf('error_code = %d\n',er)
    printf('computed_integral = %f\n',integ)
endfunction
```

Codurile funcţiilor *init, tab, integr2, prodscal* sunt editate în fişierul *integr.sci,* după codul funcţiei *integr.*

Utilizatorul trebuie să precizeze datele problemei în funcţia *Scilab* `datas`.

Exemplul 7.2 *Să se calculeze* $\iint_D xy\mathrm{d}x\mathrm{d}y$ *unde domeniul D este delimitat de curbele* $y = x^2$ *şi* $y = \sqrt{x}$.

În acest caz se defineşte funcţia:

```
function [fct,a,b,finf,fsup,tol]=datas()
  a=0;
  b=1;
  deff('z=fct(x,y)','z=x.*y');
  deff('y=finf(x)','y=x.*x');
  deff('y=fsup(x)','y=sqrt(x)');
  tol=1e-6;
endfunction
```

Rezolvarea finală este

```
exec('c:\lucru\scilab\aplicatia.sci',-1)
aplicatia('c:\lucru\scilab')
```

Rezultatele sunt

```
erro_code = 0

computed_integral =   .0833333
```

Part II

CULEGERE DE PROBLEME

Capitolul 8

Metode numerice în algebra liniară

I. Să se calculeze factorizarea LU a matricei:

1. $$\begin{pmatrix} 10 & 6 & -2 & 1 \\ 10 & 10 & -5 & 0 \\ -2 & 2 & -2 & 1 \\ 1 & 3 & -2 & 3 \end{pmatrix}$$

2. $$\begin{pmatrix} 5 & 3 & -11 \\ 4 & -5 & 4 \\ 3 & -13 & 19 \end{pmatrix}$$

3. $$\begin{pmatrix} 2 & -1 & 3 & 4 \\ 4 & -2 & 5 & 6 \\ 6 & -3 & 7 & 8 \\ 8 & -4 & 9 & 10 \end{pmatrix}$$

4. $$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 5 & 1 \\ 0 & 0 & 0 & 0 & 1 & \frac{7}{33} \end{pmatrix}$$

II. Să se calculeze descompunerea / factorizarea QR a matricei:

1. $$\begin{pmatrix} 4 & 5 & 2 \\ 3 & 0 & 3 \\ 0 & 4 & 6 \end{pmatrix}$$

2. $$\begin{pmatrix} 2 & 1 & 3 \\ 1 & 3 & 1 \\ 2 & 8 & 5 \end{pmatrix}$$

3. $$\begin{pmatrix} 3 & 4 & 7 & -2 \\ 5 & 4 & 9 & 3 \\ 1 & -1 & 0 & 3 \\ 1 & -1 & 0 & 0 \end{pmatrix}$$

III. Să se rezolve sistemele algebrice de ecuaţii liniare:

1. $$\begin{cases} x_1 + x_2 - x_3 - x_4 = 1 \\ 4x_1 - 4x_2 + 6x_3 - 2x_4 = -4 \\ x_1 + 3x_2 - 2x_3 - 2x_4 = 3 \\ 6x_1 - 6x_2 + 9x_3 - 3x_4 = -6 \end{cases}$$

2. $$\begin{cases} x_1 - 2x_2 + 3x_3 + 4x_4 = 22 \\ -4x_1 + x_2 + 2x_3 + 3x_4 = 6 \\ 3x_1 + 4x_2 - x_3 + 2x_4 = -4 \\ 2x_1 + 3x_2 + 4x_3 - x_4 = 6 \end{cases}$$

3. $$\begin{cases} 6x + 4y + z + 2t = 3 \\ 6x + 5y + 3z + 5t = 6 \\ 12x + 8y + z + 5t = 8 \\ 6x + 5y + 3z + 7t = 8 \end{cases}$$

4. $$\begin{cases} x + 2y + 3z + 4t = -4 \\ x + y + 2z + 3t = -2 \\ x + 3y + z + 2t = -3 \\ x + 3y + 3z + 2t = -5 \end{cases}$$

5. $$\begin{cases} 2x + y + z + t = 1 \\ 3x - 2y - 5z + 4t = -30 \\ x + 3y + 2z - 3t = 17 \\ x - y + z - t = 2 \end{cases}$$

6. $$\begin{cases} x + y + z + t = 4 \\ 2x + 2z + t = 4 \\ -2x + 2y - t = 4 \\ 3x + y - z = 0 \end{cases}$$

7. $$\begin{cases} 2x_1 + 3x_2 + 2x_3 + x_4 = 2 \\ x_1 + x_2 + 3x_3 + 2x_4 = 6 \\ -3x_1 + 2x_2 - x_3 - x_4 = 0 \\ x_1 - x_2 + x_3 + 3x_4 = 6 \end{cases}$$

8. $$\begin{cases} x + y + z + t = 2 \\ 2y + 2z + t = 2 \\ -2x + 2y - t = 2 \\ 3x + y - z = 2 \end{cases}$$

9. $$\begin{cases} x + y + 2z + 3t = 2 \\ 3x - y - 4z - 6t = 0 \\ 2x + 3y - 6z - 9t = -7 \\ x - 2y + 8z - 12t = 3 \end{cases}$$

10. $$\begin{cases} x + y + z - 2t + 7u = 10 \\ 2x + 5z - 2u = 32 \\ 3x + y - t = 1 \\ 2y - 5z + u = -39 \\ x - y + 3t = 7 \end{cases}$$

11. $$\begin{cases} 2x + 3y - z + t = 5 \\ x - y + 2z - 2t = -5 \\ 3x + y + 2z - 2t = -3 \\ -3x - y - 2z + 2t = 3 \end{cases}$$

12. $$\begin{cases} x + 2y + 3z + 4t = 30 \\ 2x - 3y + 5z - 2t = 3 \\ 3x + 4y - 2z - t = 1 \\ 4x - y + 6z - 3t = 8 \end{cases}$$

13. $$\begin{cases} x + 2y + 3z - 2t = 6 \\ 2x - y - 2z - 3t = 8 \\ 3x + 2y - z + 2t = 4 \\ 2x - 3y + 2z + t = -8 \end{cases}$$

14. $$\begin{cases} 2x - y + z - t = 1 \\ 2x - y - 3t = 2 \\ 3x - y + t = -3 \\ 2x + 2y - 2z + 5t = -6 \end{cases}$$

15. $$\begin{cases} x + 2y + 3z + 4t + 5u = 13 \\ 2x + y + 2z + 3t + 4u = 10 \\ 2x + 2y + z + 2t + 3u = 11 \\ 2x + 2y + 2z + t + 2u = 6 \\ 2x + 2y + 2z + 2t + u = 3 \end{cases}$$

16. $$\begin{cases} x + 2y - 3z + 4t - u = -1 \\ 2x - y + 3z - 4t + 2u = 8 \\ 3x + y - z + 2t - u = 3 \\ 4x + 3y + 4z + 2t + 2u = -2 \\ x - y - z + 2t - 3u = -3 \end{cases}$$

17. $$\begin{cases} x + y + z + t = 10 \\ x + y - z + 2t = -8 \\ 5x + 5y - z - 4t = -4 \\ x + y + 3z + 4t = 28 \end{cases}$$

IV. Să se rezolve sistemele algebrice de ecuaţii liniare incompatibile în sensul celor mai mici pătrate:

1. $$\begin{cases} 3x_1 - x_2 + x_3 = 3 \\ x_1 + 2x_2 - x_3 = -4 \\ 2x_1 - 3x_2 + 2x_3 = 3 \end{cases}$$

2. $$\begin{cases} x_1 - 3x_2 + x_3 + x_4 = 1 \\ x_1 - 3x_2 + x_3 - 2x_4 = -1 \\ x_1 - 3x_2 + x_3 + 5x_4 = 6 \end{cases}$$

3. $$\begin{cases} 2x - 3y + z = -1 \\ x + 2y - 3z = 0 \\ x - 12y + 11z = -1 \\ 4x - 15y + 9z = 0 \end{cases}$$

4. $$\begin{cases} 5x + 3y - 11z = 13 \\ 4x - 5y + 4z = 18 \\ 3x - 13y + 19z = 22 \end{cases}$$

Capitolul 9

Sisteme şi ecuaţii algebrice

I. Să se rezolve ecuaţiile polinomiale:

1. $x^6 - 2x^5 + x^3 - 6x + 6 = 0$
2. $2x^6 - x^5 - 5x^4 + 2x^3 + 20x^2 - 9x - 9 = 0$
3. $x^5 - 56x^4 - 10x^3 + 560x^2 + x - 56 = 0$
4. $x^5 - x^4 - 2x^3 + 4x^2 - 4 = 0$
5. $x^5 - 4x^4 - 9x^3 + 18x^2 + 14x - 20 = 0$
6. $x^6 - 5x^5 + 5x^4 - 2x^3 + 13x^2 + 3x + 9 = 0$
7. $x^5 - 12x^4 + 50x^3 - 88x^2 + 96x - 128 = 0$
8. $x^6 - 6x^5 + 8x^4 + 3x^3 - 16x^2 + 18x - 8 = 0$
9. $x^5 - 3x^4 + 2x^3 - x^2 + 5x - 2 = 0$
10. $x^5 - 4x^3 - 8x^2 + 32 = 0$
11. $x^n - nx + 1 = 0,\ n = 3, 4, 5$
12. $x^n - (1 + x + \ldots + x^{n-1}) = 0,\ n = 3, 4, 5$
13. $x^{n+1} - (1 + x + \ldots + x^{n-1}) = 0,\ n = 2, 3, 4$
14. $x^{n+1} - x^n - 1 = 0,\ n = 2, 3, 4$

Observaţie. Justificaţi afirmaţiile:

1. Pentru orice $n \in \mathbb{N}, n > 2$ ecuaţia $x^n - nx + 1 = 0$ are două rădăcini pozitive, $a_n \in (0,1),\ b_n \in (1,2)$.

2. Pentru orice $p \in \mathbb{N}$, ecuaţia $x^{p+1} - x^p - 1 = 0$ are o singură rădăcină $x_p > 1$ şi $\lim_{p\to\infty} x_p = 1$.

3. Pentru orice $n \in \mathbb{N}^*,\ p \in \mathbb{N}$, ecuaţia $x^{n+p} - (1 + x + \ldots + x^{n-1}) = 0$ are o singură rădăcină $x_{n,p} > 1$ şi $\lim_{n\to\infty} x_{n,p} = x_p$.

II. Să se rezolve ecuaţiile algebrice:

1. $2x - \ln x - 4 = 0$
2. $\ln 8x - 9x + 3 = 0$
3. $x - 2\cos x = 0$
4. $e^{-0.5x} - x = 0$
5. $\ln x = \frac{1}{x}$
6. $\ln x = x^2 - 1$
7. $x^2 - 2\cos x = 0$
8. $x \ln x - 14 = 0$
9. $x^x + 2x - 6 = 0$
10. $e^x + e^{-3x} - 4 = 0$
11. $\ln 7x - 8x + 1 = 0$
12. $x + e^x = e$

III. Să se rezolve sistemele algebrice de ecuaţii neliniare

1. $$\begin{cases} 2x^3 - y^2 - 1 = 0 \\ xy^3 - y - 4 = 0 \end{cases}$$

2. $$\begin{cases} e^x - y = 0 \\ x - e^{-y} = 0 \end{cases}$$

3. $$\begin{cases} e^{-xy} = x^2 - y + 1 \\ (x+0.5)^2 + y^2 = 0.6 \end{cases}$$

4. $$\begin{cases} x^2 + y^2 + z^2 = 3 \\ xy + xz - 3yz = 1 \\ x^2 + y^2 - z^2 = 1 \end{cases}$$

5. $$\begin{cases} x^2 + y^2 - 1 = 0 \\ x^3 - y = 0 \end{cases}$$

6. $$\begin{cases} 2x^2 - xy - 5x + 1 = 0 \\ x + 3\ln x - y^2 = 0 \end{cases}$$

7. $$\begin{cases} 2x^2 - xy - y^2 + 2x - 2y + 6 = 0 \\ y - x - 1 = 0 \end{cases}$$

8. $$\begin{cases} x^3 - y^2 - 1 = 0 \\ xy^3 - y - 4 = 0 \end{cases}$$

9. $$\begin{cases} x^2 - x - y^2 - y + z^2 = 0 \\ y - e^x = 0 \\ z - \ln(y) = 0 \end{cases}$$

10. $$\begin{cases} x^2 - x - y^2 = 1 \\ y - e^x = 0 \end{cases}$$

11. $$\begin{cases} x + 3\ln x - y^2 = 0 \\ 2x^2 - xy - 5x + 1 = 0 \end{cases}$$

12. $$\begin{cases} x + x^2 - 2yz - 0.1 = 0 \\ y - y^2 + 3xz + 0.2 = 0 \\ z - z^2 + 2xy - 0.3 = 0 \end{cases}$$

13. $$\begin{cases} x^3 - y + 1 = 0 \\ 0.25x^2 + y^2 - 1 = 0 \end{cases}$$

14. $$\begin{cases} x - \cos(x - y - z) = 0 \\ y - \cos(y - x - z) = 0 \\ z - \cos(z - x - y) = 0 \end{cases}$$

15. $$\begin{cases} \tan(x) - \cos(1.5y) = 0 \\ 2y^3 - x^2 - 4x - 3 = 0 \end{cases}$$

16. $\begin{cases} 3x^2 + 14y^2 - 1 = 0 \\ \sin(3x + 0.1y) + x = 0 \end{cases}$

17. $\begin{cases} 1 + x^2 - y^2 + e^x \cos y = 0 \\ 2xy + e^x \sin y = 0 \end{cases}$ $\quad \begin{array}{l} x_0 = -1 \\ y_0 = 4 \end{array}$

18. $\begin{cases} \sin x + \cos y + e^{xy} = 0 \\ \arctan(x + y) - xy = 0 \end{cases}$

Capitolul 10

Probleme de interpolare

I. Să se calculeze $L(P_n, x_0, x_1, ..., x_n; f)(z)$ pentru

1. $f(x) = x^3 - 5x^2 + x - 1$
   $x_i = 2i + 1, \quad i \in \{0,1,2,3,4,5\}$
   $z = 4$

2. $f(x) = x^4 - 10x^3 + 2x^2 + 3x + 1$
   $x_i = 1 + i, \quad i \in \{0,1,2,3,4,5,6,7,8,9\}$
   $z = 4.5$

3. $f(x) = \lg(x)$
   $x_i = e^i, \quad i \in \{0,1,2,3,4,5\}$
   $z = 1.7$

4. $f(x) = e^x$
   $x_i = -3 + i, \quad i \in \{0,1,2,3,4,5,6\}$
   $z = 1.5$

5. $f(x) = \sin(x)$
   $x_i = -\frac{\pi}{2} + i\frac{\pi}{10}, \quad i \in \{0,1,2,3,4,5,6,7,8,9,10\}$
   $z = \frac{\pi}{13}$

6. $f(x) = \cos(x)$
   $x_i = i\frac{\pi}{10}, \quad i \in \{0,1,2,3,4,5,6,7,8,9\}$
   $z = \frac{\pi}{7}$

7. $f(x) = \sqrt{x}$
   $x_i = i^2, \quad i \in \{0,1,2,3,4,5\}$
   $z = 5$

8	$f(x) = \frac{x^2+1}{x}$ $x_i = 1 + \frac{i}{5}, \quad i \in \{0,1,2,3,4,5\}$ $z = 1.3, z = 1.5, z = 1.7$

II. Să se deducă expresia polinomului de interpolare pentru datele problemei I.

III. Să se calculeze valoarea funcţiei spline cubice de interpolare pentru datele problemei I.

Capitolul 11

Derivare numerică

I. Să se calculeze derivate funcţiilor în punctul indicat:

1.	$f(x) = 2x - \ln x - 4$	$x_0 = 1$
2.	$f(x) = \ln 8x - 9x + 3$	$x_0 = 1$
3.	$f(x) = x - 2\cos x$	$x_0 = 1$
4.	$f(x) = e^{-0.5x} - x$	$x_0 = 1$
5.	$f(x) = \ln x - \frac{1}{x}$	$x_0 = 1$
6.	$f(x) = \ln x - x^2 + 1$	$x_0 = 1$
7.	$f(x) = x^2 - 2\cos x$	$x_0 = 1$
8.	$f(x) = x^x + 2x - 6$	$x_0 = 2$
9.	$f(x) = e^x + e^{-3x} - 4$	$x_0 = 1$

II. Să se calculeze jacobianul şi hessianul funcţiilor în punctul indicat:

1. $f(x,y) = \begin{pmatrix} 2x^3 - y^2 - 1 \\ xy^3 - y - 4 \end{pmatrix}$ $(x,y) = (1,2)$

2. $f(x,y) = \begin{pmatrix} x^2 - y \\ x(y+1) \end{pmatrix}$ $(x,y) = (1,2)$

3. $f(x,y) = \begin{pmatrix} \tan(xy) - x^2 \\ 0.5x^2 + 2y^2 - 1 \end{pmatrix}$ $(x,y) = (1,2)$

4. $f(x,y) = \begin{pmatrix} e^{-x^y} - x^2 - y - 1 \\ (x+0.5)^2 + y^2 - 0.6 \end{pmatrix}$ $(x,y) = (2,2)$

5. $f(x,y) = \begin{pmatrix} x^3 + y^3 - 6x + 3 \\ x^3 - y^3 - 6y + 2 \end{pmatrix}$ $(x,y) = (1,2)$

6. $f(x,y) = \begin{pmatrix} x^2 + y^2 - 1 \\ x^3 - y \end{pmatrix}$ $(x,y) = (1,2)$

7. $f(x,y) = \begin{pmatrix} 2x^2 - xy - 5x + 1 \\ x + 3\ln x - y^2 \end{pmatrix}$ $(x,y) = (1,2)$

8. $f(x,y) = \begin{pmatrix} 2x^2 - xy - y^2 + 2x - 2y + 6 \\ y - x - 1 \end{pmatrix}$ $(x,y) = (1,2)$

Capitolul 12

Calculul unui element de aproximare prin metoda celor mai mici pătrate

I. Să se calculeze polinoamele de aproximare de grad unu, doi si trei construite prin metoda celor mai mici pătrate pentru datele Problemei 1 din capitolul Interpolare.

II. Să se calculeze funcţiile de aproximare construite prin metoda celor mai mici pătrate pentru metoda Gauss-Newton de forma şi datele indicate ($t = 0:9$):

1. $\varphi(t) = a \ln(bt + c)$ $\qquad y = 2\ln(3t + 1)$
2. $\varphi(t) = a \exp(bt^2 + c)$ $\qquad y = 2\exp(-t^2)$
3. $\varphi(t) = a \sin(bt) \exp(-ct)$ $\qquad y = 2\sin(t)\exp(-2t)$
4. $\varphi(t) = a + bt + c\exp(dt)$ $\qquad y = 1 - 2t + 2\exp(-t)$

III. Să se calculeze funcţiile de aproximare construite prin metoda celor mai mici pătrate pentru metoda Levenberg-Marquardt pentru cazurile exerciţiului II.

Capitolul 13

Integrare numerică

I. Să se calculeze integralele:

1.	$\int\limits_0^1 x^2 e^x \mathrm{d}x$	2.	$\int\limits_0^\pi x^2 \cos x \mathrm{d}x$
3.	$\int\limits_4^5 \sqrt{x^2-9}\mathrm{d}x$	4.	$\int\limits_1^2 x^2 \ln x \mathrm{d}x$
5.	$\int\limits_0^{\frac{\pi}{4}} x \tan^2 x \mathrm{d}x$	6.	$\int\limits_0^{\frac{\pi}{2}} \frac{\sin 2x}{1+\sin^2 x}\mathrm{d}x$
7.	$\int\limits_0^2 \sqrt{4x-x^2}\mathrm{d}x$	8	$\int\limits_0^1 \arcsin \frac{x^2-1}{x^2+1}\mathrm{d}x$
9.	$\int\limits_0^\pi \cos^2 x \cos 4x \mathrm{d}x$	10.	$\int\limits_{\frac{\pi}{6}}^{\frac{\pi}{2}} \frac{1+\sin 2x+\cos 2x}{\sin x+\cos x}\mathrm{d}x$
11.	$\int\limits_{-1}^1 x^3\sqrt{1+e^{x^2}}\mathrm{d}x$	12.	$\int\limits_0^1 \frac{2x}{\sqrt{3+4x^2}}\mathrm{d}x$
13.	$\int\limits_{\frac{\sqrt{3}}{3}}^{\sqrt{3}} \arctan \frac{1}{x}\mathrm{d}x$	14.	$\int\limits_0^{\frac{\pi}{4}} \frac{\tan x}{1+\sqrt{2}\cos x}\mathrm{d}x$

II. Să se calculeze integralele duble:

1. $\iint\limits_D (x^2+y)\,\mathrm{d}x\mathrm{d}y$ $\quad D:$ (delimitat de) $y=x^2;\ y^2=x.$

2. $\iint\limits_D \frac{x^2}{y^2}\,\mathrm{d}x\mathrm{d}y$ $\quad D:\ y=\frac{1}{x};\ y=x;\ x=1;\ x=2.$

3. $\iint\limits_D \frac{x}{x^2+y^2}\,\mathrm{d}x\mathrm{d}y$ $\quad D:\ y=x;\ y=(x-1)^2.$

4. $\iint\limits_D \sqrt{4x^2-y^2}\,\mathrm{d}x\mathrm{d}y$ $\quad D:\ y=x;\ y=0;\ x=1.$

5. $\iint\limits_D \frac{1}{\sqrt{x}+\sqrt{y}}\,\mathrm{d}x\mathrm{d}y$ $\quad D:\ x=1,\ y=0,\ x-y=\frac{1}{2}.$

6. $\iint\limits_D \frac{2x}{\sqrt{1+y^4-x^4}}\,\mathrm{d}x\mathrm{d}y$ $\quad D:\ y=x,\ x=0,\ y=1.$

7. $\iint\limits_D (x^2y\sqrt{1-x^2-y^2}\,\mathrm{d}x\mathrm{d}y$ $\quad D:\ x\ge 0,\ y\ge 0,\ x^2+y^2=1.$

8. $\iint\limits_D \sqrt{xy-y^2}\,\mathrm{d}x\mathrm{d}y$ $\quad D$ este triunghiul cu vârfurile $O(0,0), A(10,1), B(1,1).$

Bibliografie

[1] BLAGA P., COMAN GH., POP S., TRÂMBIŢAŞ R., VASARU D., 1994, *Analiză numerică.* Îndrumar de laborator. Univ. "Babeş–Bolyai" Cluj-Napoca (litografiat).

[2] CIRA O., MĂRUŞTER Ş., 2008, *Metode numerice pentru ecuaţii neliniare.* Ed. Matrix-Rom, Bucureşti.

[3] CURTEANU S., 2001, *Calculul numeric şi simbolic în Mathcad.* Ed. Matrix-Rom, Bucureşti.

[4] DINU M., LINCĂ Gh., 1999, *Algoritmi şi teme speciale de analiză numerică.* Ed. Matrix rom, Bucureşti.

[5] GAVRILĂ C., 2007, *SCILAB Aplicaţii, Modele şi simulare Scicos* Ed. Matrix-Rom, Bucureşti.

[6] KINCAID D., CHENEY W., 1991, *Numerical Analysis.* Brooks / Cole, Pacific Grove, Ca.

[7] MARINESCU GH., BADEA L., GRIGORE G., JAMBOR C., MAZILU P., RIZZOLI I., ŞTEFAN C., 1978, *Probleme de analiză numerică.* E.D.P., Bucureşti.

[8] MARINESCU GH., RIZZOLI I., POPESCU I., ŞTEFAN C., 1987, *Probleme de analiză numerică rezolvate cu calculatorul.* Ed. Acad. R.S.R., Bucureşti.

[9] MARTIN O., 1998, *Probleme de analiză numerică.* Ed. Matrix rom, Bucureşti.

[10] ПЛИС А. И., СЛИВИНА Н.А., 1983, Лабораторныи практикум по бысшеи математике. Высшая Школа, Москва.

[11] SCHEIBER E., LIXĂNDROIU D., CISMAŞIU C., 1982, *Analiză numerică. Îndrumar de laborator.* Univ. Braşov (litografiat).

[12] SCHEIBER E., SÂNGEORZAN L., GROVU M., 1993, *Analiză numerică. Îndrumar de laborator.* Univ. "Transilvania" Braşov (litografiat).

[13] SCHEIBER E., 2010, *Java în calculul ştiinţific.* Ed. Univ. *Transilvania* Braşov.