Metode numerice în ingineria electrică

Îndrumar de laborator pentru studenții Facultății de Inginerie Electrică

Gabriela Ciuprina, Mihai Rebican, Daniel Ioan

Editura PRINTECH

GABRIELA CIUPRINA
MIHAI REBICAN • DANIEL IOAN

METODE NUMERICE ÎN INGINERIA ELECTRICĂ

ÎNDRUMAR DE LABORATOR
PENTRU STUDENȚII FACULTĂȚII DE INGINERIE ELECTRICĂ

Gabriela Ciuprina, Mihai Rebican, Daniel Ioan

Metode numerice în ingineria electrică

Îndrumar de laborator
pentru studenții facultății de Inginerie Electrică

Editura PRINTECH
BUCUREȘTI 2013

Editura PRINTECH

Tipar executat la:
S.C. ANDOR TIPO S.R.L. – Editura PRINTECH
Site: www.andortipo.ro; www.printech.ro
Adresa: Str. Tunari nr.11, Sector 2, Bucureşti
Tel./Fax: 021.211.37.12; 021.212.49.51
E-mail: comenzi@andortipo.ro

Descrierea CIP a Bibliotecii Naționale a României
CIUPRINA, GABRIELA
Metode numerice în ingineria electrică : îndrumar de laborator pentru studenții Facultății de Inginerie Electrică /
Gabriela Ciuprina, Mihai Rebican, Daniel Ioan. - București : Printech, 2013
Bibliogr.
ISBN 978-606-23-0077-7

I. Rebican, Mihai
II. Ioan, Daniel

519.6:621.3(075.8)

Gabriela Ciuprina, Mihai Rebican, Daniel Ioan

Metode numerice în ingineria electrică

Îndrumar de laborator

pentru studenţii facultăţii de Inginerie Electrică

2013

Cuprins

Notă

Acest îndrumar prezintă lucrările de laborator efectuate în prezent de studenţii Facultăţii de Inginerie Electrică, în cadrul disciplinei *Metode numerice*. Partea teoretică a acestor lucrări este preluată în cea mai mare parte din cartea *Metode numerice în ingineria electrică* [1], în timp ce programele demonstrative la care se face referire, au fost rescrise integral în Scilab (G. Ciuprina). De asemenea, în lucrarea de faţă au fost adăugate exerciţii explicative (G. Ciuprina şi M. Rebican) pentru a facilita înţelegerea metodelor explicate.

Lucrarea 0

Preliminarii asupra *Laboratorului de Metode Numerice*

0.1 Informaţii utile

Lista de lucrări

1. Algoritmi si structuri de date
2. Erori in calculele numericei
3. Metoda Gauss - fără şi cu pivotare
4. Metode iterative pentru rezolvarea sistemelor liniare
5. Rezolvarea circuitelor rezistive liniare
6. Interpolarea numerică a funcţiilor
7. Derivarea numerică a funcţiilor
8. Integrarea numerică a funcţiilor
9. Rezolvarea ecuaţiilor neliniare
10. Rezolvarea ecuaţiilor diferenţiale

Calendar

- Săptămâna 1 (S1): Prezentarea laboratorului

- S2: L1 + L2 - seminar
- S3: L1 + L2 - laborator
- S4: L3 + L4 - seminar
- S5: L3 + L4 - laborator
- S6: L5 seminar
- S7: T1 - Test de aplicaţii numerice şi întrebări - partea I
- S8: L6 - seminar
- S9: L6 - laborator
- S10: L7 + L8 - seminar
- S11: L7 + L8 - laborator
- S12: L9 + L10 - seminar
- S13: L19 + L10 - laborator
- S14: T2 - Test de aplicaţii numerice şi întrebări - partea a II-a; test de implementare

Regulament

Un student poate **intra în examen** doar dacă participă la **minim** 5 şedinţe de seminar (din maxim 6), 4 şedinţe de laborator (din maxim 5) şi cele două teste de aplicaţii numerice şi întrebărişi testul de implementare. Practic, se admite o singură absenţă la seminar şi o singură absenţă la laborator.

De menţionat că laboratorul reprezintă 50 % din punctajul notei la această disciplină. Deşi teoretic este posibil ca un student care are punctaj zero la laborator să promoveze această disciplină, acumulând punctaj maxim la examen, experienţa anilor anteriori nu a oferit niciun astfel de exemplu.

Deoarece posibilităţile de recuperare a laboratoarelor sunt limitate, **refacerea** unei lucrări de seminar sau laborator se poate realiza numai în cursul săptămânii afectate lucrării şi numai în măsura în care există un calculator liber. Este permisă refacerea lucrării numai la grupele din **seria** de care aparţine studentul respectiv. **Nu se refac lucrări de seminar sau laborator la sfârşitul semestrului!**

Înainte de fiecare şedinţă de seminar se recomandă ca fiecare student **să citească** lucrarea care urmează a fi efectuată, conform calendarului. Se va pune accent în special pe aspecte teoretice şi exemple rezolvate.

La începutul unei şedinţe de laborator fiecare student trebuie să prezinte un **pre-referat** al lucrării care urmează a fi efectuată, conform calendarului. Studenţii care nu au pre-referate nu vor fi primiţi să efectueze lucrarea. Pe parcursul şedinţei, studentul îşi completează pre-referatul cu date, grafice şi concluzii. Studenţii trebuie sa aibă la ei hartie milimetrică pentru trasarea graficelor. Astfel, se realizează **referatul**, care trebuie **predat la începutul şedinţei din săptămâna următoare**.

Modul de notare al laboratorului

Fiecare temă de laborator (L*) este punctată cu o notă între 1 şi 10 pentru referat.

Referatul va conţine:

1. Numele studentului şi grupa din care face parte
2. Numele cadrului didactic îndrumator
3. Titlul lucrarii
4. Scopul lucrarii
5. Rezultate experimentale, grafice, etc. (conform cerinţelor din îndrumar)
6. Observaţii şi concluzii

Pre-referatul va conţine punctele 1, 2, 3, 4 din referat.

Foarte important:

- Observaţiile şi, mai ales, concluziile au ponderea cea mai mare în nota primită pe referat. Un referat fără observaţii şi concluzii poate avea nota maximă 4 din 10. Două sau mai multe referate care conţin observaţii şi concluzii identice vor avea fiecare nota 1.
- Referatul nu trebuie să conţină: descrierea lucrării, principiul algoritmilor, pseudocodul algoritmilor.
- Testele de aplicaţii numerice şi întrebări (din săptămânile 7 şi 14) vor fi realizate în scris şi constau în rezolvarea numerică a unor exemple cu ajutorul metodelor studiate. Testele de aplicaţii şi întrebări vor fi punctate cu note între 1 şi 10.
- Testul de implementare (din săptămâna 14) va consta în implementarea în limbajul de programare C sau limbajul Matlab (Scilab) a unuia din pseudocodurile lucrărilor studiate. Testul de implementare va fi punctat cu o notă între 1 şi 10. Pentru pregătirea acestui test, studenţii vor fi încurajaţi să exerseze implementarea algoritmilor pe parcursul şedinţelor de laborator.

- Nota de laborator reprezintă 50% din nota disciplinei *Metode Numerice în Ingineria Electrică*, şi se calculează astfel:
  - 15% pentru referate;
  - 30% pentru cele două teste de aplicaţii numerice;
  - 5% pentru testul de implementare.

  Modul de calcul al notei finale va fi prezentat la curs.

0.2 Modul de desfăşurare a unei şedinţe de seminar sau laborator

Fiecare şedinţă de seminar sau laborator durează 2 ore şi are o anumită tematică (vezi subcapitolul 0.1).

În cadrul şedinţei de seminar sunt prezentate studenţilor de către cadrul didactic următoarele chestiuni despre lucrarea respectivă: aspecte teoretice şi exemple rezolvate. De asemenea, se vor prezenta pe scurt chestiunile de studiat si modul de lucru, aspecte necesare pentru efectuarea lucrării de laborator în săptămâna următoare.

De menţionat că exemplele rezolvate la fiecare şedinţă de seminar sunt esenţiale pentru cele două teste de aplicaţii numerice.

Activităţile propriu-zise pe care trebuie să le desfăşoare fiecare student într-o şedinţă de laborator sunt prezentate mai jos.

Prima activitate constă în **exploatarea unor programe demonstrative** ce ilustrează tematica lucrării.

Sistemul de operare sub care se lucrează este Linux (`http://www.linux.org/`). Programele demonstrative sunt scrise in Scilab (`http://www.scilab.org/`) dar exploatarea lor nu necesită cunoaşterea acestui limbaj de programare. În urma exploatării acestor programe, studentul trebuie **să redacteze un referat**, conform cerinţelor fiecărei lucrări, şi să îl **predea la sfârşitul şedinţei**.

În măsura timpului disponibil, se va **implementa** cel puţin unul din algoritmii studiaţi în cadrul temei respective. Implementarea se va efectua în limbajul C sau limbajul Scilab, în concordanţă cu pseudocodul prezentat în lucrare.

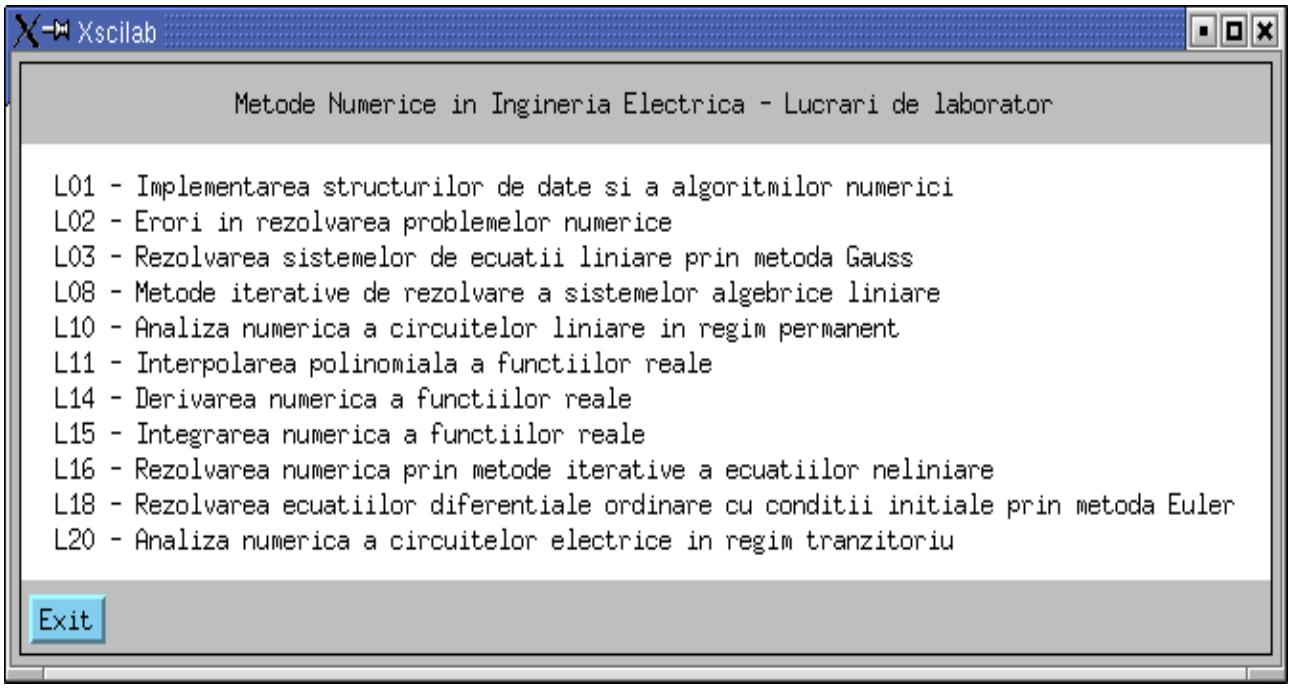

Figura 1: Meniul principal al programelor demonstrative.

0.3 Programe demonstrative

Pentru lansarea programelor demonstrative faceţi un singur clic pe icoana cu sigla LMN. În continuare urmaţi modul de lucru descris în îndrumar.

Observaţie: programele demonstrative sunt disponibile la adresa `http://mn.lmn.pub.ro/`. Pentru a putea lucra cu ele trebuie să aveţi instalate sistemul de operare Linux şi pachetul de programe Scilab. Arhiva `LMN.tar.gz` trebuie decomprimată şi dezarhivată. Lansaţi Scilab şi la consola acestuia tastaţi `exec(main.sci)`. Meniul principal este cel prezentat în figura 1.

Notă importantă

Pentru ca un referat să fie luat în considerare, prezenţa la laborator este obligatorie. Este obligatoriu ca exploatarea acestor programe să fie făcută sub îndrumarea cadrului didactic.

Exerciţii:

1. Intraţi în contul dvs.;
2. Remarcaţi icoana LMN şi executaţi un singur clic pe ea;
3. Inchideţi programele demonstrative.

0.4 Implementare in C

Înainte de toate, vă este utilă o reîmprospătare a cunoştiinţelor de C dobândite în anii anteriori.

Metodele numerice ce vor fi studiate implementează în exclusivitate algoritmi numerici, care efectuează de cele mai multe ori calcule cu vectori şi matrice.

De aceea, în cele ce urmează vom face câteva consideraţii asupra declarării şi alocării vectorilor şi matricelor, astfel încât activitatea de programare în cadrul acestui laborator să fie cât mai eficientă.

Pentru început însă, vă recomandăm să studiaţi cu atenţie următoarele exemple şi apoi, pentru întelegerea lor, să citiţi paragrafele următoare.

0.4.1 Exemple

La adresa `http://mn.lmn.pub.ro/` găsiţi două exemple pentru a vă familiariza cu stilul de lucru al acestui laborator.

1. Creaţi un director numit L0 cu comanda `mkdir L0`
2. Descarcaţi fişierele `nrutil_lmn.c, nrutil_lmn.h, aduna_vec.c, rw_matrix.c.`
3. Comentaţi conţinutul acestor fişiere.
4. Compilaţi primul exemplu cu comanda

   ```
   gcc aduna_vec.c nrutil_lmn.c -o aduna_vec
   ```

5. Executaţi programul cu comanda

   ```
   ./aduna_vec
   ```

6. Compilaţi al doilea exemplu cu comanda

   ```
   gcc rw_matrix.c nrutil_lmn.c -o rw_matrix
   ```

7. Executaţi programul cu comanda

   ```
   ./rw_matrix
   ```

Observaţie:

Comanda de compilare conţine un fişier sursă C (în care se află funţia `main` şi o funcţie ce implementează un anumit pseudocod), apoi fişierul `nrutil_lmn.c` (în care se află funcţiile de alocare/dealocare de memorie). Numele ce urmează după `-o` este numele programului executabil generat.

Iată conţinutul acestor fişiere:

Fişierul `aduna_vec.c`

```
#include "nrutil_lmn.h"

void aduna_vectori1 (int , VECTOR , VECTOR , VECTOR );

int
main (void)
{
/* program principal - adunarea a doi vectori
 * apeleaza aduna_vectori */

  int n; /* dimensiunea vectorilor */
  VECTOR a, b; /* vectorii de intrare */
  VECTOR c; /* vectorul rezultat c = a + b */

  int i;

  printf ("\n Introduceti dimensiunea vectorilor ");
  scanf ("%d", &n);

/* aloca spatiu de memorie pentru vectori */
  a = vector (1, n);
  b = vector (1, n);
  c = vector (1, n);

/* citeste vectorii a si b */
  for (i = 1; i <= n; i++)
    {
      printf ("\n a[%d] = ", i);
      scanf ("%f", &a[i]);
    }
  for (i = 1; i <= n; i++)
    {
      printf ("\n b[%d] = ", i);
      scanf ("%f", &b[i]);
    }

  aduna_vectori1 (n, a, b, c);

/* afiseaza rezultat */
  printf ("\n Rezultatul este \n");
  for (i = 1; i <= n; i++)
    printf ("c[%d] = %f \n", i, c[i]);


/* elibereaza spatiu de memorie */
  free_vector (a, 1, n);
  free_vector (b, 1, n);
  free_vector (c, 1, n);
  return (0);
}

void
aduna_vectori1 (int n, VECTOR a, VECTOR b, VECTOR c)
{
/* aduna doi vectori a + b, a caror indecsi incep de la 1 */
  int i;
  for (i = 1; i <= n; i++)
    c[i] = a[i] + b[i];
}
```

Fişierul `rw_matrix.c`:

```
#include "nrutil_lmn.h"
```

```
int
main (void)
{
/* program principal - citeste o matrice patrata, reala,  si o scrie */

  int n; /* dimensiunea matricelor */
  MATRIX a; /* matricea */

  int i, j;

  printf ("\n Introduceti dimensiunea matricei ");
  scanf ("%d", &n);

/* aloca spatiu de memorie pentru matrice  */
  a = matrix (1, n, 1, n);

/* citeste matricea */
  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
      {
printf ("\n a[%d][%d] = ", i, j);
scanf ("%f", &a[i][j]);
      }

/* afiseaza matricea  */
  printf ("\n Matricea citita este \n");
  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
      printf ("a[%d][%d] = %f \n", i, j, a[i][j]);


/* elibereaza spatiu de memorie */
  free_matrix (a, 1, n, 1, n);
  return (0);
}
```

Fişierul `nrutil_lmn.c`

```
#include "nrutil_lmn.h"

void
nrerror (char error_text[])
{
  fprintf (stderr, "Run-time error...\n");
  fprintf (stderr, "%s\n", error_text);
  fprintf (stderr, "...now exiting to system...\n");
  exit (1);
}



VECTOR
vector (int nl, int nh)
{
  VECTOR v;

  v = (float *) malloc ((unsigned) (nh - nl + 1) * sizeof (float));
  if (!v)
    nrerror ("allocation failure in vector()");
  return v - nl;
}


IVECTOR
ivector (int nl, int nh)
{
  IVECTOR v;

  v = (int *) malloc ((unsigned) (nh - nl + 1) * sizeof (int));
  if (!v)
    nrerror ("allocation failure in ivector()");
  return v - nl;
}


MATRIX
matrix (int nrl, int nrh, int ncl, int nch)
{
  int i;
  MATRIX m;

  m = (float **) malloc ((unsigned) (nrh - nrl + 1) * sizeof (float *));
```

```
  if (!m)
    nrerror ("allocation failure 1 in matrix()");
  m -= nrl;

  for (i = nrl; i <= nrh; i++)
    {
      m[i] = (float *) malloc ((unsigned) (nch - ncl + 1) * sizeof (float));
      if (!m[i])
nrerror ("allocation failure 2 in matrix()");
      m[i] -= ncl;
    }
  return m;
}


IMATRIX
imatrix (int nrl, int nrh, int ncl, int nch)
{
  int i;
  IMATRIX m;

  m = (int **) malloc ((unsigned) (nrh - nrl + 1) * sizeof (int *));
  if (!m)
    nrerror ("allocation failure 1 in imatrix()");
  m -= nrl;

  for (i = nrl; i <= nrh; i++)
    {
      m[i] = (int *) malloc ((unsigned) (nch - ncl + 1) * sizeof (int));
      if (!m[i])
nrerror ("allocation failure 2 in imatrix()");
      m[i] -= ncl;
    }
  return m;
}

void
free_vector (VECTOR v, int nl, int nh)
{
  free ((char *) (v + nl));
}

void
free_ivector (IVECTOR v, int nl, int nh)
{
  free ((char *) (v + nl));
}

void
free_matrix (MATRIX m, int nrl, int nrh, int ncl, int nch)
{
  int i;

  for (i = nrh; i >= nrl; i--)
    free ((char *) (m[i] + ncl));
  free ((char *) (m + nrl));
}

void
free_imatrix (IMATRIX m, int nrl, int nrh, int ncl, int nch)
{
  int i;

  for (i = nrh; i >= nrl; i--)
    free ((char *) (m[i] + ncl));
  free ((char *) (m + nrl));
}
```

Fişierul `nrutil_lmn.h`

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<malloc.h>
#include<time.h>

typedef float **MATRIX;
typedef float *VECTOR;
typedef int **IMATRIX;
typedef int *IVECTOR;

void nrerror (char error_text[]);
```

```
VECTOR vector (int nl, int nh);
IVECTOR ivector (int nl, int nh);
MATRIX matrix (int nrl, int nrh, int ncl, int nch);
IMATRIX imatrix (int nrl, int nrh, int ncl, int nch);
void free_vector (VECTOR v, int nl, int nh);
void free_ivector (IVECTOR v, int nl, int nh);
void free_matrix (MATRIX m, int nrl, int nrh, int ncl, int nch);
void free_imatrix (IMATRIX m, int nrl, int nrh, int ncl, int nch);
```

0.4.2 Vectori

În C există o strânsă corespondenţă între adrese (pointeri) şi tablouri. În acest paragraf vom considera tablourile unidimensionale.

Valoarea reprezentată de `a[j]` este acelaşi lucru cu `*(a+j)` adică "conţinutul adresei obţinute incrementând pointer-ul `a` cu `j`. O consecinţă a acestei definiţii este aceea că dacă `a` este adresa unei locaţii valide, atunci `a[0]` este întotdeauna definit. Tablourile unidimenionale au în C, în mod natural, originea în 0. Un şir definit de

```
float b[4]
```

are referinţele valide `b[0], b[1], b[2]` şi `b[3]`, dar nu şi `b[4]`.

Problema este că mulţi algoritmi sunt descrişi în mod natural cu indici care încep de la 1. Cu siguranţă că aceşti algoritmi pot fi modificaţi, dar aceasta presupune o aritmetică suplimentară în operarea cu indici, lucru care nu este prea plăcut. Putem însă folosi puterea limbajului C pentru ca această problemă să dispară. Ideea este simplă:

```
float b[4], *bb;
bb = b - 1;
```

Pointer-ul `bb` indică acum o locaţie înaintea lui `b`. În consecinţă, elementele `bb[1], bb[2], bb[3]` şi `bb[4]` există şi vectorul `bb` are indici ce pornesc de la 1.

Uneori este convenabil să avem vectori care pornesc din 0, iar alteori este convenabil să avem vectori care pornesc din 1. De exemplu, coeficienţii unui polinom $a_0 + a_1 x + \ldots a_n x^n$ necesită un vector cu indici ce încep cu 0, pe când termenul liber al unui sistem de ecuaţii $b_i, i = 1, \ldots, n$ necesită un vector cu indici ce încep din 1.

Pentru a evita rescrierea algoritmilor ce sunt deduşi în mod natural cu indici ce pornesc de la 1, puteţi folosi o funcţie cu următoarea definiţie.

```
typedef float *VECTOR;

VECTOR vector (int nl, int nh)
{
```

```
  VECTOR v;

  v = (float *) malloc ((unsigned) (nh - nl + 1) * sizeof (float));
  if (!v)
    nrerror ("allocation failure in vector()");
  return v - nl;
}
```

Această funcţie alocă un vector de variable de tip `float`, care vor fi accesate cu indici cuprinşi între `nl` şi `nh`.

O utilizare tipică a acestei funcţii este

```
float *b;
b = vector(1,7);
```

Această funcţie precum şi funcţii similare ce alocă vectori de întregi se găsesc în fişierul `nrutil_lmn.c` pe care îl puteţi descărca de la adresa `http://mn.lmn.pub.ro/`.

Acest fişier conţine şi rutinele corespunzătoare de dealocare. De exemplu, pentru dealocarea memoriei ocupate de vectorul definit mai sus, instrucţiunea este

```
free_vector(b,1,7);
```

0.4.3 Matrice

Problema indicilor ce pornesc de la 0 sau de la 1 apare şi în cazul matricelor. În sintaxa C, lucrul cu tabele bidimensionale este puţin mai complicat. Să considerăm o valoare reală `a[i][j]` unde `i` şi `j` sunt întregi. Un compilator de C va genera coduri maşină diferite pentru această referinţă, aceasta depinzând de declaraţia pentru variabila `a`. Dacă `a` a fost declarată de dimensiune fixă, de exemplu `float a[2][4]` atunci codul maşină ar putea fi descris astfel: "la adresa `a` adună de 4 ori `i`, apoi adună `j` şi întoarce valoarea astfel adresată. Observaţi că valoarea constantă 4 trebuie cunoscută pentru a efectua în mod corect calculele.

Să presupunem că `a` a fost declarat ca `float **a`. Atunci codul maşină `a[i][j]` este "la adresa `a` adună `i`, valoarea astfel adresată consider-o o nouă adresă, la care adună `j` şi întoarce valoarea astfel adresată.

Ilustrarea celor două moduri de accesare a unei valori este dată în figura 2. Observaţi că în al doilea caz nu este necesară cunoaşterea dimensiunii matricei şi că nu este nevoie de nici o înmulţire. O indirectare suplimentară înlocuieşte aceste informaţii. Această a doua schemă o recomandăm pentru lucrul la laboratorul de metode numerice.

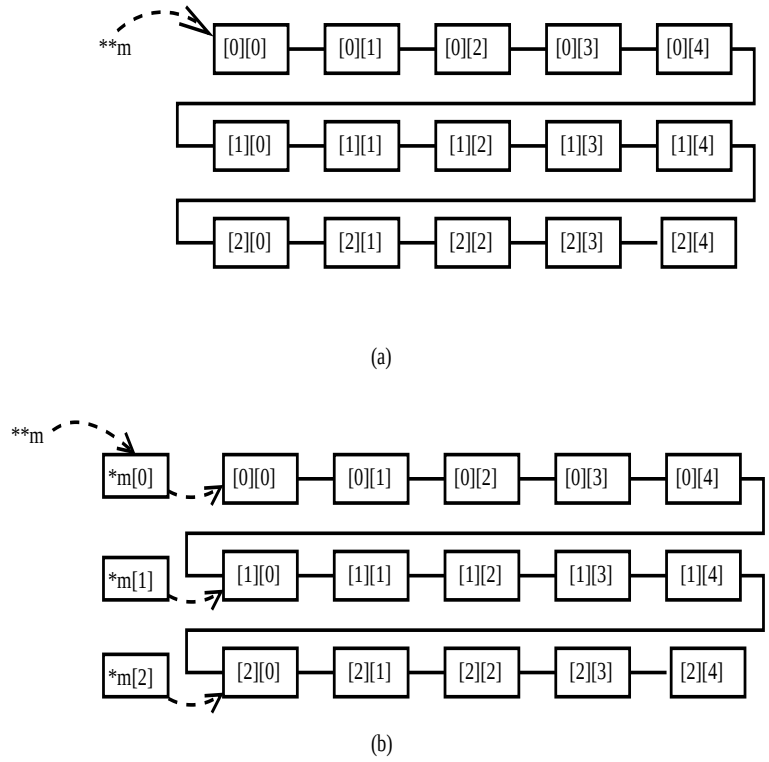

Figura 2: Două scheme de memorare pentru matricea `m`. Liniile întrerupte reprezintă pointeri la adrese, iar liniile continue conectează locaţii de memorie secvenţiale. (a) Pointer la un tablou bidimensional de dimensiune fixată; (b) Pointer la un tablou de pointeri către pointeri la linii.

Ideea pe scurt: vom evita tablourile bidimensionale de dimensiune fixă. Ele nu sunt structuri de date potrivite pentru reprezentarea matricelor în calculele ştiinţifice.

Lucrarea 1

Implementarea structurilor de date şi a algoritmilor numerici

1.1 Caracterizarea lucrării

Rezolvarea problemelor ştiinţifice şi tehnice cu ajutorul calculatorului numeric presupune identificarea unei metode matematice de rezolvare şi apoi implementarea acesteia pe un sistem de calcul.

În afara algoritmului propriu-zis de rezolvare, un rol la fel de important în implementare îl are şi alegerea exactă a structurilor de date. În final, programul se bazează pe algoritm (descrierea operaţiilor ce vor fi efectuate pentru obţinerea soluţiei) şi pe structura de date (modul în care se reprezintă datele de intrare, variabilele intermediare şi datele de ieşire). Pentru descrierea structurilor de date şi a algoritmilor se poate utiliza un limbaj de programare (Basic, FORTRAN, Pascal, C, etc.) sau un pseudolimbaj cu o sintaxă mai puţin rigidă. Pentru a evidenţia invarianţa algoritmilor la limbajul ales se preferă a doua metodă.

Scopul acestei lucrări este de a familiariza utilizatorii cu **gândirea algoritmică structurată** (prin folosirea pseudocodului), de a evidenţia metodele de implementare a pseudocodului în diferite limbaje de programare şi, nu în ultimul rând, de a evidenţia importanţa tipurilor abstracte de date cu caracter matematic (vectori, matrice, numere complexe, etc.).

1.2 Descrierea pseudolimbajului

Pseudocodul (pseudolimbajul) este o metodă simplă şi eficientă pentru reprezentarea unui algoritm şi a structurilor de date asociate. Pseudocodul este de fapt un text alcătuit

din linii (rânduri), fiecare conţinând, de regulă, o *declaraţie* (al cărei scop principal constă în descrierea datelor) sau o *instrucţiune* (care descrie o operaţie ce va fi efectuată).

O linie de pseudocod este alcătuită din cuvinte şi simboluri (caractere speciale nealfanumerice). Anumite cuvinte, cu o semnificaţie bine determinată, independente de aplicaţie, se numesc **cuvinte cheie** şi pentru a fi deosebite de celelalte cuvinte, specifice aplicaţiei sunt de obicei îngroşate.

Orice linie poate conţine precizări suplimentare, numite *comentarii*, care ajută la înţelegerea pseudocodului, fără să facă parte din descrierea propriu-zisă a algoritmului sau a structurii de date. Comentariile sunt plasate la sfârşitul liniei cu caracterul ";".

1.2.1 Structuri de date

Declaraţiile se referă la datele cu care se operează şi care pot fi de tip *simplu (fundamental)* sau *structurate (agregate)*.

Se consideră următoarele categorii de *date fundamentale*:

- **logic** - date cu două valori (0 = fals şi 1 = adevărat);
- **întreg** - date care pot avea valori întregi;
- **real** - aproximări ale numerelor reale;
- **caracter** - literă, cifră sau semn special (aritmetic sau de punctuaţie).

Exemple de declaraţii ale unor variabile de tip fundamental:

logic $l1, l2, l3$
întreg i, j, s
real a, m, x, j
caracter c

Se constată că o declaraţie conţine cuvântul cheie ce specifică tipul, urmat de lista numelor variabilelor de tipul respectiv, nume separate prin virgulă. Numele variabilelor sunt cuvinte alcătuite din litere şi cifre (primul caracter trebuie să fie literă) şi au semnificaţia limitată la aplicaţia respectivă. Se recomandă ca numele variabilelor să fie sugestiv alese (de exemplu: "masa", "volum", "curent", etc.), iar pentru eliminarea ambiguităţilor, fiecare mărime cu care se operează într-o problemă să aibă propria sa declaraţie, eventual însoţită de un comentariu:

real q ; sarcina corpului
real i ; intensitatea curentului

Pentru rezolvarea unor probleme mai complicate, tipurile fundamentale de date nu sunt suficiente, ci este necesară o "agregare" a datelor. Acest lucru se realizează prin folosirea cuvintelor cheie:

- **tablou** - structură de date ce conţine un număr cunoscut de elemente de acelaşi tip;
- **înregistrare** - structură de date ce poate conţine elemente de tipuri diferite.

O declaraţie de variabilă de tip tablou conţine cuvântul cheie **tablou**, numele tabloului (variabilei) şi dimensiunea acestuia.

Exemple de declaraţii de tablou:

- **tablou real** $V[3]$; V este un tablou de trei elemente reale
- **tablou întreg** $a[5], b[7]$; a este un tablou de 5 intregi şi b este un tablou de 7 întregi

Cu toate că tabloul are un singur nume pentru întreaga structură de date, elementele acestuia se identifică folosind indexul (numărul de ordine al elementului), ca de exemplu: $V(1)$, $V(2)$, $b(5)$, etc. În pseudocod se permite şi folosirea notaţiilor V_1, V_2, b_5, etc. sau V_i, cu condiţia ca i să fie cuprins între 1 şi dimensiunea tabloului (pentru tabloul V de nau sus, $1 \leq i \leq 3$).

Exemplu de declaraţie a unei înregistrări:

înregistrare punct
    logic cartezian
    real x_1
    real x_2

Această înregistrare se referă la conceptul de "punct", care este definit ca o agregare a unei variabile logice "cartezian" şi a două variabile reale x_1, x_2. Punctul poate fi descris prin coordonate carteziene (cartezian = adevărat, $x_1 = x$, $x_2 = y$) sau polare (cartezian = fals, x_1 = raza, x_2 = unghiul).

Un alt exemplu îl constituie :

înregistrare monom
    întreg n
    real a

care reprezintă monomul ax^n, descris de ordinul întreg n şi coeficientul real a.

Referirea la un element al înregistrării se face prin numele înregistrării urmat de numele câmpului (elementului), nume separate prin caracterul ".", de exemplu: monom.a, punct.cartezian, etc.

În pseudolimbaj se admit mai multe niveluri de agregare, deci se pot construi tabele de tabele, tablou de înregistrări sau alte combinaţii.

Pentru a uşura astfel de construcţii se introduce cuvântul cheie **tip**, care atunci când precede o declaraţie are ca efect extinderea tipurilor standard **logic**, **întreg**, **real**, **caracter**, **tablou**, **înregistrare** cu un nou tip de date definit de utilizator şi indicat de numele "variabilei" introdus prin declaraţia respectivă.

De exemplu:

```
tip tablou real vector[3]  ; introduce tipul vector
înregistrare monom         ; introduce tipul monom
        întreg n
        real a
```

Declaraţia:

```
tip tablou monom polinom[10]
```

introduce tipul "polinom" ca fiind un tablou de 10 elemente, fiecare element fiind de tip monom.

1.2.2 Structuri de control

Instrucţiunile unui pseudocod descriu operaţiile pe care le va efectua sistemul de calcul cu datele descrise anterior prin declaraţii. Instrucţiunile sunt de două feluri: *simple* şi *structurate*.

Instrucţiunile simple sunt:

- de atribuire;
- de intrare;
- de ieşire.

Instrucţiunea de atribuire are sintaxa:

variabilă = expresie

în care "variabilă" este numele unei variabile a cărei valoare va fi modificată în urma instrucţiunii, iar "expresie" este o construcţie sintactică alcătuită din constante, variabile, operatori şi paranteze, după regulile uzuale din algebră. Efectul execuţiei instrucţiunii de atribuire constă în evaluarea expresiei şi modificarea în concordanţă a variabilei al cărei nume se află la stânga semnului egal.

Se consideră că operanzii care intervin în expresii au valori corespunzătoare unuia din tipurile fundamentale. Dacă operanzii sunt de tip logic, atunci se admit operatorii logici **nu**, **sau**, **şi**, ca în exemplele:

logic l1, l2, l3
l1 = **nu** (l2)
l3 = l1 **sau** l2
l3 = l1 **şi** l2

Dacă operanzii sunt numerici (**real** sau **întreg**) se admit operatori aritmetici (+, -, *, /) sau de relaţie ($=, \neq, \leq, \geq, <, >$), ca în exemplele:

real a, b, x, y;
logic l1;
$y = (ax + b)/(2x - b)$
$l1 = (x \leq b)$ **şi** $(x > a)$

În primul caz rezultatul este de tip numeric, iar în al doilea caz, de tip logic.

Operanzii de tip caracter admit doar operatori de relaţie (se admite ordonarea lexicografică conform codului ASCII), ca în exemplul:

$l2 = C > 'A'$; caracterul conținut de variabila C este ulterior literei $'A'$

Instrucţiunile de intrare-ieşire au sintaxa:

citește variabile
scrie variabile

Prima instrucţiune are ca efect transferul pe canalul de intrare al unei valori (de exemplu introducerea ei de la tastatură), care modifică valoarea variabilei specificate, iar a doua are ca efect transferul valorii variabilei pe canalul de ieşire (de exemplu afişarea pe ecran sau tipărirea la imprimantă).

De exemplu, pseudocodul:

real x, y, s, p
citeşte x, y
$s = x + y$
$p = xy$
scrie s, p
stop

reprezintă un program simplu, capabil să calculeze suma şi produsul a două numere reale. Programul se încheie printr-o altă instrucţiune simplă, cu sintaxa:

stop

care are ca efect *terminarea execuţiei* programului respectiv.

Pentru realizarea unor operaţii mai complicate se folosesc în afara instrucţiunilor simple, instrucţiunile structurate, care sunt:

- secvenţa;
- decizia (cu sau fără alternativă);
- ciclul (cu test iniţial, cu test final sau cu contor);
- rutina (procedură sau funcţie).

Secvenţa (sau blocul de instrucţiuni) reprezintă un şir de instrucţiuni simple sau structurate (scrise câte una pe linie, în linii succesive), care se execută una după alta, în ordinea în care au fost scrise.

Programul anterior este un exemplu de secvenţă.

Decizia este o instrucţiune care permite controlul execuţiei şi are una din următoarele variante de secvenţă:

- *decizia simplă*

dacă condiţie **atunci**
secvenţă

- *decizia cu alternativă*

dacă condiţie **atunci**
secvenţă 1
altfel
secvenţa 2

în care "condiţie" este o expresie de tip logic, iar "secvenţa" este o secvenţă de una sau mai multe instrucţiuni. Pentru a uşura înţelegerea acestei instrucţiuni se constată că secvenţele sunt scrise indentat (retrase faţă de cuvântul cheie **dacă**). În urma execuţiei acestei instrucţiuni se evaluează expresia logică "condiţie". Dacă valoarea rezultată este adevărată, atunci se execută "secvenţa" (respectiv "secvenţa 1"), altfel se continuă cu instrucţiunea următoare (respectiv se execută "secvenţa 2").

Exemple de instrucţiuni de decizie:

; funcţia matematică modul $y = |x|$
dacă $x \geq 0$ **atunci**
$y = x$
altfel
$y = -x$

Pentru implementarea funcţiei definite pe porţiuni:

$$y = \begin{cases} 0, & x < -1 \\ 2x, & -1 \leq x \leq 1 \\ x^2, & x > 1 \end{cases}$$

se poate folosi pseudocodul:

dacă $x < -1$ **atunci**
$y = 0$
altfel dacă $(x \geq -1)$ **şi** $(x \leq 1)$
$y = 2x$
altfel
$y = x$

Ciclul reprezintă o instrucţiune care permite repetarea unei secvenţe. Se deosebesc trei feluri de cicluri:

- cu test iniţial;
- cu test final;
- cu contor.

Ciclul cu test iniţial are sintaxa:

cât timp condiţie **repetă**
    secvenţa

Efectul este evaluarea expresiei logice "condiţie". Dacă rezultatul este afirmativ (adevărat), atunci se execută secvenţa şi ciclul se reia până când "condiţie" devine falsă, după care se sare peste "secvenţă" şi se continuă cu următoarea instrucţiune. Se constată că este posibilă repetarea infinită a ciclului dacă valoarea logică a condiţiei rămâne mereu adevărată. Este responsabilitatea programatorului să asigure caracterul finit al ciclului.

În exemplul următor:

$k = 1$
$s = 0$
cât timp $a_k > 0$ **repetă**
    $s = s + a_k$
    $k = k + 1$
scrie s

este utilizat ciclul cu test iniţial pentru a aduna elementele unui tablou până la întâlnirea primului element negativ.

Ciclul cu test final are sintaxa:

repetă
    secvenţă
până când condiţie

"secvenţa" fiind executată repetat până când "condiţie" devine adevărată. Spre deosebire de ciclul cu test iniţial, în acest caz corpul ciclului este executat cel puţin o dată.

În exemplul următor se calculează cu eroare impusă suma seriei cu termenul general $(-1)^k/k!$

întreg k
real s, t, eps
$t = 1$
$k = 1$
$s = 0$
repetă
    $t = \frac{-t}{k}$
    $s = s + t$
    $k = k + 1$
până când $|t| < eps$

Ciclul cu contor permite repetarea unei secvenţe de un număr determinat de ori. Sintaxa ciclului cu contor este:

pentru contor = val_in, val_fin, pas **repetă**
    secvenţă

în care "contor" este numele unei variabile de tip **întreg**, "val_in", "val_fin" şi "pas" sunt constante sau expresii de tip **întreg**. Secvenţa de instrucţiuni din corpul ciclului este repetată pentru valori succesive ale contorului, pornind de la valoarea iniţială "val_in", incrementat cu pasul "pas" până când acesta depăşeşte valoarea finală "val_fin". Dacă în instrucţiune lipseşte valoarea "pas" se presupune că aceasta are valoarea implicită "1".

În exemplul următor:

$s = 0$
pentru $k = 1, n$
    $s = s + a_k$
scrie s

se calculează suma primelor n elemente ale tabloului a.

Se întâlnesc des situaţii în care o anumită secvenţă de instrucţiuni trebuie executată de mai multe ori, în momente diferite ale execuţiei unui program. Pentru a evita rescrierea de mai multe ori a acestei secvenţe se foloseşte conceptul de *rutină*.

O rutină reprezintă o secvenţă de declaraţii şi instrucţiuni căreia i se atribuie un nume. Dacă într-un program se face apel la o rutină, controlul se transferă rutinei, iar după încheierea acesteia se revine în programul apelant. În consecinţă, o rutină presupune pe de o parte *definiţia* acesteia (prin specificarea declaraţiilor şi instrucţiunilor care o

alcătuiesc), iar pe de altă parte *apelul* ei. Utilizarea rutinelor este justificată chiar şi în cazul în care ele sunt apelate o singură dată în program, deoarece ele permit *structurarea modulară* a unui algoritm. Pentru a realiza aceasta, o rutină trebuie să aibă o anumită consistenţă, să îndeplinească o funcţie bine definită, ceea ce permite reutilizarea ei şi în alte programe.

După modul de apelare, rutinele se împart în două categorii:

- proceduri;
- funcţii.

Procedura este definită printr-o construcţie sintactică de forma:

procedură nume(listă parametri formali)
secvenţa
retur

în care "nume" este numele procedurii alcătuit din caractere alfanumerice, iar "lista parametrilor formali" conţine nume de variabile separate prin virgule. O parte din parametrii formali sunt *parametrii de intrare* (ai căror valori provin din programul apelant şi se transferă procedurii), iar restul sunt *parametrii de ieşire* (ai căror valori se determină în procedură şi se transferă programului apelant). Apelul unei proceduri astfel definite se face prin invocarea numelui ei urmat de lista parametrilor actuali:

nume (lista de parametri actuali)

Parametrii actuali trebuie să concorde ca număr, tip şi ordine cu cei formali (dar nu obligatoriu şi ca nume).

În exemplul următor:

citeşte a, b
sumaprod (a, b, s, p)
scrie s, p
sumaprod $(s, p, s1, p1)$
scrie $s1, p1$

se face apel la procedura definită astfel:

procedură sumaprod $(x, y, suma, prod)$
real $x, y, suma, prod$
    suma $= x + y$
    prod $= xy$
retur

în care are x, y sunt parametri formali de intrare şi $suma, prod$ sunt parametri formali de ieşire. În urma primului apel al acestei proceduri (cu parametri actuali de intrare a, b) se calculează şi se afişează $s = a + b$, $p = ab$, iar în urma celui de al doilea apel se calculează şi se afişează $s1 = s + p = a + b + ab$, $p1 = sp = (a + b)ab$.

În cazul unei proceduri, numărul parametrilor de intrare sau de ieşire este arbitrar (poate fi inclusiv nul).

O altă variantă de rutină este *funcţia*, la care toţi parametrii formali sunt parametri de intrare, dar rutina de tip funcţie întoarce o valoare. Definiţia funcţiei se realizează prin construcţia sintactică:

funcţie nume (lista parametrii formali)
    secvenţa
întoarce valoare

Funcţia poate fi apelată ca operand într-o expresie, în particular într-o atribuire de tipul

valoare = nume (lista de parametri actuali)

Funcţia se aseamănă cu o procedură cu un singur parametru de ieşire dar, faţă de aceasta, are o flexibilitate suplimentară în apel. Exemplul următor:

real a, b, x
citeşte a, b
$x =$ min$(a, b)/$max(a, b)
scrie x

foloseşte două funcţii max şi min definite prin:

funcţie max(x, y) ; valoare maximă
    real x, y, m

```
    dacă x > y atunci
        m = x
    altfel
        m = y
    întoarce m
```

```
funcţie min(x, y)        ; valoare minimă
    real x, y, m
    dacă x > y atunci
        m = y
    altfel
        m = x
    întoarce m
```

Funcţiile elementare (modul, radical, putere, exponenţială, logaritm, sin, cos, tg, sh, ch, th) se consideră predefinite în pseudocod, deoarece majoritatea limbajelor de programare de nivel înalt le au implementate.

1.3 Tipuri abstracte de date

Tipurile abstracte de date reprezintă concepte importante ale programării structurate moderne.

Un tip abstract de date este o structură de date definită de utilizator (folosind cuvântul cheie **tip**) completată cu o serie de rutine (proceduri sau funcţii), care definesc operaţiile ce pot fi executate cu date de acest tip. În acest fel se extind tipurile fundamentale de date (**logic**, **întreg**, **real**, care reprezintă algebra booleană, inelul $\mathbb{Z}$ şi corpul $\mathbb{R}$) prin implementarea şi a altor structuri matematice (algebrice sau topologice) cum sunt spaţiile vectoriale, corpul numerelor complexe, inelul polinoamelor sau al matricelor.

Aceste structuri de primă importanţă în matematică se întâlnesc în rezolvarea pe cale numerică a multor probleme ştiinţifice sau tehnice. Ele nu sunt legate de o anumită aplicaţie ci au un caracter general.

În continuare sunt prezentate în pseudocod câteva din tipurile abstracte de date uzuale.

Spaţiul vectorial n-dimensional $\mathbb{R}^n$

Un element $v = (v_1, v_2, \ldots, v_n)$ al spaţiului $\mathbb{R}^n$ poate fi reprezentat ca un tablou cu n componente reale. Operaţiile algebrice caracteristice unui spaţiu vectorial sunt:

- adunarea $u, v \in \mathbb{R}^n$

$$u + v = (u_1, \ldots, u_n) + (v_1, \ldots, v_n) = (u_1 + v_1, \ldots, u_n + v_n)$$

- înmulţirea cu un scalar $a \in \mathbb{R}$, $v \in \mathbb{R}^n$

$$av = a(v_1, v_2, \ldots, v_n) = (av_1, av_2, \ldots, av_n).$$

Elementul nul $0 = (0, 0...0) \in \mathbb{R}^n$ are toate componentele nule, iar opusul unui vector $-v = (-v_1, \ldots, -v_n) = -1 \cdot v$. Implementarea acestui tip abstract de date este realizată de pseudocodul:

tip tablou real vector (n)
procedură adv(u, v, w) ; adunarea vectorilor w = u + v
    vector u, v ; date de intrare
    vector w ; suma - data de ieşire
    pentru $i = 1, n$
        $w_i = u_i + v_i$
retur

procedură inm(a, v, w) ; înmulteşte vectorul v cu scalarul a
    real a ; data de intrare
    vector v ; vector de intrare
    vector w ; rezultă w =av
    pentru $i = 1, n$
        $w_i = av_i$
retur

Spaţiul euclidian

Spaţiul euclidian este un spaţiu vectorial înzestrat cu o normă care provine dintr-un produs scalar:

$$u \cdot v = (u_1, u_2, \ldots, u_n) \cdot (v_1, v_2, \ldots, v_n) = \sum_{i=1}^{n} u_i v_i$$

unde $u, v \in \mathbb{R}^n$. Norma euclidiană este o aplicaţie pozitiv definită:

$$\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$$

$$\|u\| = \sqrt{u \cdot u}$$

cunoscută şi sub numele de modulul vectorului.

Pentru definirea unui spaţiu euclidian se foloseşte pseudocodul:

tip vector euclidian
funcţie produs(u, v) ; produsul scalar
euclidian u, v ; factori
real p ; produsul scalar p = uv
$p = 0$
pentru $i = 1, n$
$p = p + u_i v_i$
întoarce p

funcţia norma (u) ; norma euclidiană a vectorului u
euclidian u, v
real norma
norma $= \sqrt{\text{produs}(u, u)}$
întoarce norma

Inelul polinoamelor

Un polinom algebric de gradul n este o construcţie de forma:

$$P(x) = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$$

definită de coeficienţii $p = (a_0, a_1, \ldots, a_n)$, deci este reprezentabil printr-un tablou real. Un polinom este un element al unui spaţiu liniar (vectorial) deoarece suma a două polinoame se realizează prin suma coeficienţilor, iar produsul cu un scalar prin înmulţirea tuturor coeficienţilor cu scalarul respectiv. Prin produsul a două polinoame de acelaşi grad se obţine un polinom de grad dublu. Dacă:

$$P(x) = \sum_{i=0}^{n} a_i x^i$$

şi

$$Q(x) = \sum_{i=0}^{n} b_i x^i,$$

atunci

$$\begin{aligned} P(x)Q(x) &= \left(\sum_{i=0}^{n} a_i x^i\right)\left(\sum_{i=0}^{n} b_i x^i\right) = \\ right &= a_0 b_0 + (a_0 b_1 + a_1 b_0)x + (a_0 b_2 + a_1 b_1 + a_2 b_0)x^2 + \ldots \end{aligned}$$

O operaţie importantă este cea de evaluare a unui polinom. Pentru a micşora efortul de calcul, aceasta va fi efectuată sub forma:

$$P(x) = a_0 + a_1 x + a_2 x^2 + .. + a_n x^n = a_0 + x(a_1 + x(a_2 + x(a_3 + x(\ldots))))$$

care evită calculul puterilor x^k.

Următorul pseudocod permite implementarea acestui tip abstract de date:

```
tip vector polinom
procedură prodp(n, r, q, p); înmulțirea polinoamelor p = rq
        intreg n                        ; gradul factorilor
        polinom r, q                    ; factori
        polinom p                       ; rezultat
        întreg i, j, k                  ;variabile intermediare
        pentru i = 0, 2n
                p_i = 0
        pentru i = 0, n
                pentru j = 0, n
                        k = i + j       ; gradul monomului rezultat
                        p_k = p_k + r_i q_j
retur


funcție evalp(n, p, x)          ; evaluează polinomul p(x)
        intreg n                        ; gradul polinomului
        polinom p                       ; tabelul coeficienților
        real x                          ; variabile independentă
        real v                          ; valoare polinom
        v = p_n
        pentru i = n − 1, 0, −1
                        v = p_i + vx
întoarce v
```

Inelul matricelor pătratice ($\mathbb{R}^{n\times n}$)

O matrice pătrată $A \in \mathbb{R}^{n\times n}$ este un tablou bidimensional:

$$A = [a_{ij}]_{i=1,n;j=1,n} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}.$$

Suma a două matrice: $A = [a_{ij}], B = [b_{i,j}] \in \mathbb{R}^{n\times n}$ se realizează pe elemente:

$$A + B = [a_{ij} + b_{ij}],$$

înmulţirea cu un scalar $\alpha \in \mathbb{R}$ este definită de:

$$\alpha A = [\alpha a_{ij}],$$

iar produsul a două matrice pătrate este o matrice de acelaşi tip:

$$AB = [a_{ij}][b_{ij}] = [\sum_{k=1}^{n} a_{ik} b_{kj}]_{i=1,n;j=1,n}.$$

Pentru implementarea acestui tip abstract de date se foloseşte pseudocodul:

```
tip tablou real matrice (n, n)
procedură adm(a, b, c)        ; adunarea matricelor c = a + b
      matrice a, b            ; termeni
      matrice c               ; rezultat
      pentru i = 1, n
            pentru j = 1, n
                  c_ij = a_ij + b_ij
retur


procedură inm(α, b, c)        ; înmulțește matricea b cu scalarul α
      real α
      matrice b               ; termeni
      matrice c               ; rezultat
      pentru i = 1, n
            pentru j = 1, n
                  c_ij = α b_ij
retur


procedură prodm(a, b, c)      ; înmulțirea matricelor
      matrice a, b            ; factori
      matrice c               ; produs
      pentru i = 1, n
            pentru j = 1, n
                  c_ij = 0
                  pentru k = 1, n
                        c_ij = a_ik b_kj + c_ij
retur
```

Corpul numerelor complexe

Un număr complex $z = a + jb$ este reprezentat printr-o pereche de numere reale (a - partea reală, b - partea imaginară).

Adunarea numerelor complexe $z_1 = a_1 + jb_1$, $z_2 = a_2 + jb_2$ se realizează conform relaţiei:

$$z_1 + z_2 = (a_1 + a_2) + j(b_1 + b_2).$$

Elementul neutru faţă de adunare este $0 = 0 + j0$, iar opusul unui număr complex $-z = -(a + jb) = -a - jb$.

Înmulţirea numerelor complexe $z_1 = a_1 + jb_1$, $z_2 = a_2 + jb_2$ este realizată de operaţia:

$$z_1 z_2 = (a_1 + jb_1)(a_2 + jb_2) = a_1 a_2 - b_1 b_2 + j(a_1 b_2 + a_2 b_1).$$

Elementul neutru faţă de înmulţire este $1 = 1 + j0$, iar inversul unui număr complex $1/z = 1/(a + jb) = (a - jb)/(a^2 + b^2)$.

O funcţie importantă este modulul unui număr complex, definit prin:

$$|z| = |a + jb| = \sqrt{a^2 + b^2}.$$

Următorul pseudocod implementează acest tip abstract de date:

tip înregistrare complex
    real : re
    real : im

procedură sumac(u, v, w) ; calculează suma numerelor complexe
    complex u, v ; termeni
    complex w ; w = u + v
    $w.re = u.re + v.re$
    $w.im = u.im + v.im$
retur

procedură prodc(u, v, w) ; calculează produsul numerelor complexe
    complex u, v ; termeni
    complex w ; w = uv
    $w.re = u.re \cdot v.re - u.im \cdot v.im$
    $w.im = u.re \cdot v.im + u.im \cdot v.re$
retur

procedură difc(u, v, w) ; calculează diferenţa numerelor complexe
    complex u, v ; termeni
    complex w
    $w.re = u.re - v.re$
    $w.im = u.im - v.im$
retur

procedură rapc(u, v, w) ; calculează raportul numerelor complexe
    complex u, v ; termeni
    complex w
    real $v2$
    $v2 = v.re \cdot v.re + v.im \cdot v.im$
    $w.re = (u.re \cdot v.re + u.im \cdot v.im)/v2$
    $w.im = (u.im \cdot v.re - u.re \cdot v.im)/v2$
retur

funcţia modulc(u) ; calculează modulul numărului complex
    complex u
    real m
    $m = \sqrt{(u.re \cdot u.re + u.im \cdot u.im)}$
întoarce m

Următorul exemplu de program:

complex $Z1, Z2, Z3, Z4, Z$
citeşte $Z1, Z2$
prodc $(Z1, Z2, Z3)$
sumac $(Z1, Z2, Z4)$
rapc $(Z3, Z4, Z)$
scrie Z
stop

permite calculul impedanţei complexe echivalente a două elemente cu impedanţe complexe date, conectate în paralel.

1.4 Complexitatea algoritmilor

Calitatea unui algoritm care se presupune că permite obţinerea soluţiei corecte este apreciată prin *eficienţa sa spaţială* (memoria necesară datelor şi programului) şi *temporală* (timpul de calcul necesar obţinerii soluţiei).

De exemplu, algoritmul pentru înmulţirea matricelor pătrate n dimensionale necesită $3n^2$ locaţii de memorie (în fiecare memorându-se un număr real, element al matricei A, B sau C). Dacă dimensiunea n a matricei creşte de 10 ori, spaţiul necesar creşte de 100 ori, motiv pentru care se spune că algoritmul este de ordinul 2 şi se scrie $\mathrm{M} = O(n^2)$.

Pentru evaluarea modului în care timpul de calcul depinde de dimensiunea problemei rezolvată de algoritm, se adoptă ca referinţă timpul necesar efectuării unei operaţii elementare (adunare sau înmulţire de numere reale) şi apoi se evaluează numărul acestor operaţii. De exemplu, pentru calculul produsului scalar a doi vectori n dimensionali se efectuează $n-1$ adunări şi n înmulţiri deci un total de $2n-1$ operaţii. Deoarece la dublarea dimensiunii se dublează practic numărul de operaţii, se spune ca acest algoritm are ordinul de complexitate liniar şi se notează $\mathrm{T} = O(n)$.

În algoritmii numerici timpul de calcul este consumat mai ales în operaţiile repetate ciclic. Ordinul de complexiate este în general dat de numărul ciclurilor incluse unul în altul, ca în exemplele:

$c = ab$; ordinul 0

pentru $i = 1, n$
    $c_i = a_i \cdot b_i$; ordinul 1

pentru $i = 1, n$
    pentru $j = 1, n$
        $c_{ij} = a_i - b_j$; ordinul 2

pentru $i = 1, n$
    pentru $j = 1, n$
        pentru $k = 1, n$
            $c_{ij} = a_{ik} b_{kj}$; ordinul 3

În consecinţă, produsul scalar a 2 vectori este un algoritm de ordinul 1, $\mathrm{T} = \mathrm{O}(n)$, iar produsul a două matrice pătrate este de ordinul 3, $\mathrm{T} = \mathrm{O}(n^3)$.

Aceeaşi problemă poate avea doi sau mai mulţi algoritmi de rezolvare cu ordine de complexitate diferite. De exemplu, pentru evaluarea unui polinom se poate folosi procedura:

```
funcţie evalpp (n, p, x)
    întreg n              ; gradul polinomului
    polinom p             ; coeficientii polinomului
    real x                ; variabila independentă
    real v                ; valoarea polinomului
    real xk               ; variabilă intermediară
    v = p_0
    pentru i = 1, n
            xk = p_i
            pentru k = 1, i
                    xk = xk · x
            v = v + xk
întoarce v
```

Acest algoritm are ordinul de complexitate $\mathrm{T} = O(n^2/2)$, deci timpul de calcul creşte practic cu pătratul gradului polinomului, faţă de algoritmul folosit în procedura evalp, care are ordinul de complexitate liniar $\mathrm{T} = O(n)$ şi care este deci mai avantajos.

O metodă eficientă pentru micşorarea timpului de calcul constă în scoaterea în afara ciclurilor a calculelor repetate, care au rezultate identice. De exemplu, dacă se doreşte generarea matricei $A = [a_{ij}] \in \mathbb{R}^{n\times n}$ cu $a_{ij} = f(x_i)g(y_j)$ în care evaluările funcţiilor f şi g cer un efort de calcul mult mai mare decât o înmulţire, se poate folosi algoritmul "natural":

```
pentru i = 1, n
    pentru j = 1, n
            a_ij = f(x_i) · g(y_j)
```

cu ordinul de complexitate $O(2n^2)$, în care s-a folosit ca operaţie de referinţă evaluarea funcţiei f sau g.

Dacă evaluarea funcţiei f se face în afara ciclului interior:

```
pentru i = 1, n
    fx = f(x_i)
    pentru j = 1, n
```

$$a_{ij} = fx \cdot g(y_j)$$

timpul de calcul scade practic la jumătate, numărul de evaluări ale funcţiilor fiind $n(n+1)$ şi ordinul de complexitate $O(n^2)$. Acest ordin se reduce şi mai mult, dacă se adoptă algoritmul:

pentru $i = 1, n$
    $fx_i = f(x_i)$
pentru $j = 1, n$
    $gy_j = g(y_j)$
pentru $i = 1, n$
    pentru $j = 1, n$
        $a_{ij} = fx_i \cdot gy_j$

cu ordinul de complexitate liniar $O(2n)$. Reducerea timpului de calcul s-a făcut pe seama creşterii necesarului de memorie. În prima variantă M = O($n^2 + 2n$), în a doua M = O(n^2+2n+1), iar în a treia M = $O(n^2+4n)$. La dimensiuni mari ale lui n însă, necesarul de memorie suplimentar este nesemnificativ, astfel încât varianta a treia, cu timp de calcul liniar este cea mai bună.

1.5 Chestiuni de studiat

1. Analiza experimentală a complexităţii unui algoritm;
2. Utilizarea tipurilor abstracte de date în electrotehnică;
3. Utilizarea unor tipuri abstracte de date într-un limbaj de nivel înalt;
4. Căutarea pe Internet a unor informaţii legate de implementarea unor structuri de date.

1.6 Modul de lucru

Pentru desfăşurarea lucrării se lansează programele demonstrative (clic pe pictograma LMN) şi apoi se selectează lucrarea *Implementarea structurilor de date şi a algoritmilor numerici* din meniul principal afişat.

Se afişează meniul lucrării care are două opţiuni:

- Analiza algoritmilor
- Calcule în complex

Utilizatorul va selecta succesiv opţiunile dorite.

a) *Analiza experimentală a complexităţii unui algoritm*

Se selectează opţiunea *Analiza algoritmilor* din meniul lucrării. Aceasta are ca efect lansarea unui submeniu din care se pot apela următoarele proceduri:

- produs scalar a doi vectori n dimensionali;
- produsul a două polinoame de grad n;
- produsul a două matrice pătrate de $n \times n$ elemente.

Pentru fiecare din cele trei cazuri, utilizatorul trebuie să aleagă valorile lui n. De exemplu, pentru valoarea iniţială 0, valoare finală 100000 şi pas 10000, produsul scalar va fi apelat, succesiv pentru $n = 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000, 100000$. Programul calculează produsul scalar a doi vectori aleatori de dimensiune n şi afişează în consola Scilab timpul de calcul.

Datele trebuie trecute într-un tabel de tipul

n	20000	40000	60000	80000	100000
Timp CPU [s] (produs scalar)					

Pentru produsul a două polinoame, dimensiunile recomandate sunt $n = 50$, 100, 150, 200, 250, 300.

Pentru produsul a două matrice, dimensiunile recomandate sunt $n = 10$, 20, 30, 40, 50, 60, 70, 80..

Se vor reprezenta grafic pe hârtie milimetrică cele trei curbe şi se vor compara rezultatele cu estimările teoretice.

Indicaţie: se va verifica în cazul produsului scalar că raportul timp/dimensiune este aproximativ constant, în cazul produsului a două polinoame că raportul timp/(dimensiune la pătrat) este aproximativ constant, etc..

Din datele obţinute în cele trei cazuri se va estima timpul CPU al unei operaţii elementare (se consideră că timpul unei adunări este egal cu timpul unui înmulţiri).

b) *Utilizarea tipurilor abstracte de date în electrotehnică*

Se selectează opţiunea *Calcule în complex*, care are ca efect lansarea unui program ce permite efectuarea diferitelor calcule algebrice cu numere complexe. Programul emulează un calculator de buzunar, care foloseşte notaţia poloneză inversă (operand, operand, operator), dar care nu operează cu numere reale ci cu numere complexe.

Introducerea operanzilor în stivă se face prin selectarea butonului:

`Stivă` → `Inserare element`.

Selectarea butonului `Stivă` permite şi apelul unor funţii utile ca:

`Stivă` → `Ştergere element` sau

`Stivă` → `Repetă vârf stivă`

Programul permite aplicarea unor operatori unari elementului care se află în vârful stivei. Aceştia sunt:

`Operatori_unari` → `Opus`

`Operatori_unari` → `Invers`

`Operatori_unari` → `Conjugat`

Programul permite şi aplicarea unor operatori binari, operanzii fiind în acest caz penultimul şi ultimul element (vârful) din stivă. Pentru aceasta selectaţi, respectiv:

`Operatori_binari` → `Adunare`

`Operatori_binari` → `Scădere`

`Operatori_binari` → `Înmulţire`

`Operatori_binari` → `Împărţire`

Stiva este vizualizată pe ecran pe cinci coloane (numar curent, parte reală, parte imaginară, modul şi argument exprimat în radiani. Operatorii unari acţionează asupra ultimului element introdus în stivă, înlocuindu-l cu valoarea obţinută în urma aplicării operatorului, iar stiva nu îşi modifică dimensiunea.

Operatorii binari folosesc ca operanzi ultimele 2 elemente din stivă iar rezultatul înlocuieşte aceste 2 elemente (stiva se micşorează cu un element).

În consecinţă operatorii unari pot fi aplicaţi, dacă în stivă se află cel puţin un element, iar operatorii binari, dacă în stivă se află cel puţin 2 elemente.

Stiva este alimentată prin introducerea unui nou număr complex (dimensiunea ei creşte cu un element).

Pentru a facilita manipularea stivei sunt disponibili alţi 2 operatori cu caracter nenumeric (repetă vârful stivei, şterge vârful stivei), care modifică (incrementează respectiv decrementează) înălţimea stivei.

Se propune folosirea acestui program în rezolvarea unei probleme de electrotehnică şi anume studiul unui circuit de curent alternativ format din trei impedanţe $\underline{Z}_1 = 10 + 5j$, $\underline{Z}_2 = 3 - 7j$, $\underline{Z}_3 = 5 + 2j$ conectate mixt ($\underline{Z}_1$ în paralel cu $\underline{Z}_2$ şi rezultatul în serie cu $\underline{Z}_3$) alimentat de la o sursă cu t.e.m reprezentată în complex ca $\underline{E} = 100j$.

Se vor calcula succesiv

$$\begin{aligned}
\underline{Z}_{12} &= (\underline{Z}_1\underline{Z}_2)/(\underline{Z}_1 + \underline{Z}_2) = 1/(1/\underline{Z}_1 + 1/\underline{Z}_2) \\
\underline{Z} &= \underline{Z}_{12} + \underline{Z}_3 \\
\underline{I} &= \underline{E}/\underline{Z} \\
\underline{I}_1 &= \underline{I}\underline{Z}_2/(\underline{Z}_1 + \underline{Z}_2) = \underline{I}/(\underline{Z}_1/\underline{Z}_2 + 1) \\
\underline{I}_2 &= \underline{I}\underline{Z}_1/(\underline{Z}_1 + \underline{Z}_2) = \underline{I} - \underline{I}_1 \\
\underline{S}_g &= \underline{E}\underline{I}^* \\
\underline{S}_1 &= \underline{Z}_1 I_1^2 \\
\underline{S}_2 &= \underline{Z}_2 I_2^2 \\
\underline{S}_3 &= \underline{Z}_3 I^2 \\
\underline{S}_c &= \underline{S}_1 + \underline{S}_2 + \underline{S}_3 \\
i_1(t) &= I_1\sqrt{2}\sin(\omega t + \phi_1) \\
i_2(t) &= I_2\sqrt{2}\sin(\omega t + \phi_2)
\end{aligned}$$

c) *Implementarea unor tipuri abstracte de date într-un limbaj de nivel înalt*

Descărcaţi de la adresa `http://mn.lmn.pub.ro/` fişierele `complex.c`, `complex.h` şi `elth.c`. Fişierele `complex.*` sunt aceleaşi cu `http://www.numerical-recipes.com/pubdom/complex.c.txt`, respectiv `http://www.numerical-recipes.com/pubdom/complex.h.txt`. Studiaţi modul de implementare corpul numerelor complexe în limbajul C.

În fişierul `elth.c` este începută rezolvarea problemei de electrotehnică de la punctul b). Comanda de compilare este

```
gcc elth.c complex.c -lm -o elth
```

Programul obţinut se execută cu

```
./elth
```

Continuaţi scrierea programului care vă va permite să rezolvaţi aceeaşi problemă. Verificaţi rezultatele cu cele pe care le-aţi obţinut la punctul b.

d) *Căutarea pe Internet a unor informaţii legate de implementarea unor structuri de date*

Căutaţi pe Internet biblioteci informaţii legate de implementarea matricelor rare (matrice cu multe elemente nule). Cuvinte cheie recomandate: *Sparse Matrix Compression Formats*, *Sparse Matrix Storage Format*.

1.7 Exemple

1.7.1 Exemple rezolvate

1. *Fie pseudocodul de mai jos*:

```
; datele de intrare ale problemei
întreg n                    ; dimensiunea problemei
real c
tablou real a[n], b[n]      ; indicele tabloului începe de la 1
; datele de ieșire ale problemei
real s
; variabile intermediare
întreg i
; introducerea datelor de intrare
citește n
citește c
pentru i = 1, n
        citește a_i, b_i
; inițializarea soluției
s = 0
; calculul soluției
pentru i = 1, n
        s = s + a_i/b_i · c
; afișarea soluției
scrie s
```

Se cer:

(a) să se estimeze ordinul de complexitate al algoritmului din punct de vedere al timpului de calcul şi al necesarului de memorie. Se consideră ca operaţii elementare (de referinţă) următoarele operaţii: +, -, *, /, iar ca locaţie elementară de memorie, locaţia ocupată de un numărul real;

(b) să se scrie formula matematică de calcul a variabilei s, în funcţie de datele problemei;

(c) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze noul timp de calcul.

Rezolvare:

(a) Algoritmul conţine un singur ciclu cu contor, în cadrul căruia se repetă de n ori o instrucţiune de atribuire, în care se efectuează operaţii elementare. Din aceste motive, ne aşteptăm ca timpul de calcul să depindă liniar de dimensiunea problemei, n ($T \approx O(n)$).

În continuare vom estima exact timpul de calcul. Pentru fiecare valoare a contorului i, în interiorul ciclului, se efectuează 3 operaţii elementare (+, /, *). Deoarece i ia valori de la 1 la n, numărul total de operaţii elementare efectuate este $3n$. Astfel, ordinul de complexitate al algoritmului din punct de vedere al timpului de calcul este $T = O(3n) \cong O(n)$. De exemplu, dacă dimensiunea problemei creşte de 10 ori, timpul de calcul va creşte de 10 ori. Acest algoritm are ordinul de complexitate liniar (de ordinul 1).

Pentru estimarea necesarului de memorie este necesar să se evalueze numărul de locaţii de memorie utilizate, mai precis, numărul de variabile reale declarate la începutul algoritmului. Astfel, sunt declarate 2 numere reale (s şi c) şi 2 tablouri de numere reale de dimensiune n (a şi b), în total $2n + 2$ numerele reale. Spunem că ordinul de complexitate al algoritmului din punct de vedere al necesarului de memorie este $M = O(2n + 2) \cong O(n)$. De exemplu, dacă se creşte de 10 ori dimensiunea tablourilor, spaţiul de memorie necesar creşte de aproximativ 10 ori. Din punct de vedere al necesarului de memorie, algoritmul este de ordinul 1.

(b) Se observă că algoritmul implementează următoarea formulă matematică:

$$s = \sum_{i=1}^{n} \frac{a_i}{b_i} \cdot c.$$

(c) Pentru reducerea timpului de calcul, înmulţirea cu variabila c poate fi scoasă în afara ciclului:

$s = 0$
pentru $i = 1, n$
    $s = s + a_i/b_i$
$s = s \cdot c$

Aceasta corespunde evaluării expresiei s cu formula $s = c \sum_{i=1}^{n} a_i/b_i$. Numărul de operaţii elementare efectuate este $2n + 1$, iar ordinul de complexitate din punct de vedere al timpului de calcul devine $T = O(2n + 1) \cong O(n)$. Această variantă a algoritmului este mai rapidă decât varianta iniţială (de aproximativ 1.5 ori).

2. *Fie pseudocodul de mai jos*:

; declararea variabilelor
întreg n, i
real s
tablou real $a[n][n]$
; introducerea datelor de intrare
. . . . . .
; instrucţiuni
$s = 0$
pentru $i = 1, n$
    pentru $j = 1, n$
        $s = s + a_{i,j} \cdot v_j \cdot \sqrt{c}$
scrie s

Se cer:

(a) să se declare variabilele j, c şi v;

(b) să se estimeze ordinul de complexitate al algoritmului din punct de vedere al timpului de calcul şi al necesarului de memorie. Se consideră ca operaţii elementare (de referinţă) următoarele operaţii: +, -, *, /, $\sqrt{}$, iar ca locaţie elementară de memorie, locaţia ocupată de un numărul real;

(c) să se scrie formula matematică de calcul a variabilei s, în funcţie de datele problemei;

(d) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze timpul de calcul.

Rezolvare:

(a) Deoarece variabila j este un contor, cu valori de la 1 la n (variabilă intreagă), j trebuie declarat ca un întreg. Variabilele c şi v sunt utilizate la calculul rezultatului final, variabila reală s, astfel, c şi v sunt numere reale. Mai mult, în algoritm, referirea la variabila v se face prin v_j, unde $j = 1, n$, ceea ce înseamnă că variabila v reprezintă un tablou de numere reale de dimensiune n. În concluzie, declaraţiile sunt:
    întreg j
    real c
    tablou real $v[n]$

(b) Algoritmul conţine două cicluri cu contor (de la 1 la n) imbricate. Instrucţiunea de atribuire din interiorul ciclurilor se repetă de n^2 ori şi conţine operaţii elementare. Ne aşteptăm ca ordinul de complexitate să fie $T \approx O(n^2)$.

    Pentru fiecare valoare a contoarelor i şi j, în interiorul ciclurilor, se efectuează 4 operaţii elementare (+, *, *, $\sqrt{}$). Pentru fiecare valoare a lui i se efectuează

$4n$ operaţii. Cum sunt n valori posibile pentru i, numărul total de operaţii elementare efectuate va fi $n*(4n) = 4n^2$. Ordinul de complexitate al algoritmului din punct de vedere al timpului de calcul este $T = O(4n^2) \cong O(n^2)$. Acest algoritm are ordinul de complexitate pătratic (de ordinul 2). De exemplu, dacă n creşte de 10 ori, timpul de calcul creşte de $100 = 10^2$ ori.

În algoritm sunt declarate două variabile reale (s şi c), un tablou unidimensional (vector) de numere reale de dimensiune n (v) şi un tablou bidimensional (matrice) de numere reale de dimensiune $n \times n$ (a). În total, sunt declarate $n^2 + n + 2$ numerele reale. Astfel, ordinul de complexitate al algoritmului din punct de vedere al necesarului de memorie este $M = O(n^2 + n + 2) \cong O(n^2)$. Se observă că, şi din punctul de vedere al necesarului de memorie, algoritmul este de ordinul 2.

(c) Algoritmul implementează următoarea formulă matematică:

$$s = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} \cdot v_j \cdot \sqrt{c}$$

(d) Reducerea timpului de calcul se poate realiza prin scoaterea înmulţirii cu $\sqrt{c}$ în afara ciclurilor:

$s = 0$
pentru $i = 1, n$
    pentru $j = 1, n$
    $s = s + a_{i,j} \cdot v_j$
$s = s \cdot \sqrt{c}$

Aceasta corespunde implementării formulei matematice:

$$s = \sqrt{c} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} \cdot v_j.$$

Acum, numărul de operaţii elementare efectuate este $2n^2 + 2$, iar ordinul de complexitate din punct de vedere al timpului de calcul devine $T = O(2n^2+2) \cong O(n^2)$.

3. *Fie pseudocodul de mai jos*:

; declaraţii variabile
întreg n, i
real p, a, rez, $term$
tablou real $b[n]$
; introducere date de intrare
citeşte n

pentru $i = 1, n$
    citește b_i
citește a, p
; inițializare soluție
$rez = 0$
; calcul soluție
pentru $i = 1, n$
    $term = p + f(b_i)$
    $term = term \cdot f(a)$
    $rez = rez + term$
; afișare soluție
scrie rez

Se cer:

(a) să se estimeze ordinul de complexitate al algoritmului din punct de vedere al timpului de calcul şi al necesarului de memorie. Se consideră ca operaţie elementară (de referinţă) evaluarea funcţiei f, iar ca locaţie elementară de memorie, locaţia ocupată de un numărul real;

(b) să se scrie formula matematică de calcul a rezultatului rez, în funcţie de datele problemei;

(c) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze timpul de calcul.

Rezolvare:

(a) Există un singur ciclu cu contor în interiorul căruia există operaţia de referinţă. Ne aşteptăm ca ordinul de complexitate din punct de vedere al timpului de calcul să fie $T \cong O(n)$.

Într-adevăr, pentru fiecare valoare i, în interiorul ciclului se efectuează 2 evaluări ale funcţiei, deci 2 operaţii elementare. Deoarece i poate lua n valori, rezultă că numărul total de operaţii elementare este $2n$. Deci, $T = O(2n) \cong O(n)$.

Pentru a estima necesarul de memorie, este util să se inspecteze declaraţiile ce conţin cuvântul cheie **real**. Sunt declaraţi 4 scalari reali şi un tablou de reali cu n componente. În consecinţă, $M = O(n + 4) \cong O(n)$.

(b) Se observă că algoritmul implementează următoarea formulă matematică:

$$rez = \sum_{i=1}^{n} [p + f(b_i)] \cdot f(a).$$

(c) Algoritmul poate fi îmbunătăţit din punct de vedere al timpului de calcul observând că $f(a)$ este evaluat în mod inutil de n ori, de fiecare dată când se schimbă valoarea contorului. Această evaluare poate fi scoasă în afara ciclului, astfel:

real val
$rez = 0$
$val = f(a)$
pentru $i = 1, n$
    $term = p + f(b_i)$
    $term = term \cdot val$
    $rez = rez + term$

Acum, avem $T = O(n)$ şi $M = O(n + 5) \cong O(n)$.

Această variantă este de 2 ori mai rapidă decât varianta iniţială. Costul îmbunătăţirii este nesemnificativ, memorarea unei noi variabile reale val.

Această implementare corespunde evaluării rezultatului cu formula:

$$rez = f(a) \sum_{i=1}^{n} [p + f(b_i)] .$$

4. *Fie pseudocodul de mai jos*:

întreg n, i, j, k
real p, rez, s, t_1, t_2
tablou real $a[n]$, $b[n]$
citeşte n
pentru $i = 1, n$
    citeşte a_i, b_i
citeşte p
$rez = 0$
pentru $i = 1, n$
    $rez = rez + f(a_i)$
    $t_1 = p$
    pentru $j = 1, n$
        $t_2 = g(b_j)$
        $s = 0$
        pentru $k = 1, n$
            $s = s + f(a_i)$
        $t_2 = t_2 + s$
        $t_1 = t_1 \cdot t_2$
    $rez = rez + t_1$
scrie rez

Se cer:

(a) să se estimeze ordinul de complexitate al algoritmului din punct de vedere al timpului de calcul şi al necesarului de memorie. Se consideră ca operaţii elementare (de referinţă) evaluările funcţiilor f şi g, iar ca locaţie elementară de memorie, locaţia ocupată de un numărul real;

(b) să se scrie formula matematică de calcul a rezultatului rez, în funcţie de datele problemei;

(c) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze timpul de calcul.

Rezolvare:

(a) Sunt 3 cicluri imbricate, ciclul din interior conţinând operaţia elementară, astfel este de aşteptat să avem $T = O(n^3)$.

Într-adevăr, numărul de operaţii elementare pentru valori fixe ale contorilor i, j, k este 1 (o evaluare a funcţiei f); numărul de operaţii elementare pentru valori fixe ale contorilor i, j este $n+1$ (n evaluări ale funcţiei f şi o evaluare a funcţiei g), numărul de operaţii elementare pentru i fixat este $n(n+1)+1 = n^2+n+1$. Astfel, numărul total de operaţii elementare este $n(n^2+n+1) = n^3+n^2+n$. Deci, $T = O(n^3+n^2+n) \cong O(n^3)$. Inspectând declaraţiile care conţin cuvântul cheie **real**, se observă că necesarul de memorie este $M = O(2n+5) \cong O(n)$.

(b) Formula matematică:

$$rez = \sum_{i=1}^{n} \left\{ f(a_i) + p \prod_{i=1}^{n} \left[g(b_j) + \sum_{k=1}^{n} f(a_i) \right] \right\}.$$

(c) Simplificarea pseudocodului se realizează simplificând formula de calcul. Notăm:

$$S = \sum_{i=1}^{n} f(a_i), \qquad P = p \prod_{i=1}^{n} \left[g(b_j) + S \right].$$

Atunci:

$$rez = \sum_{i=1}^{n} \left[f(a_i) + P \right] = \sum_{i=1}^{n} f(a_i) + nP = S + nP.$$

Algoritmul devine:

$S = 0$
pentru $i = 1, n$
    $S = S + f(a_i)$
$P = p$
pentru $j = 1, n$
    $P = p \cdot [g(b_j) + S]$
$rez = S + n \cdot P$

Acum, ordinul de complexitate din punct de vedere al timpului de calcul este $T = O(2n) \cong O(n)$.

1.7.2 Exemple propuse

1. *Fie pseudocodul de mai jos*:

   întreg n
   real s
   tablou real $b[n]$; indicele tabloului începe de la 1
   . . . . . .
   $s = 0$
   pentru $i = 1, n$
   $s = s + \sqrt{a_i/b_i} \cdot \ln(c)$

   Se cer:
   (a) să se declare variabilele i, c şi a;
   (b) să se estimeze ordinul de complexitate al algoritmului din punct de vedere al timpului de calcul şi al necesarului de memorie. Se consideră ca operaţii elementare (de referinţă) următoarele operaţii: +, -, *, /, $\sqrt{}$, ln , iar ca locaţie elementară de memorie, locaţia ocupată de un numărul real;
   (c) să se scrie formula matematică de calcul a variabilei s, în funcţie de datele problemei;
   (d) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze timpul de calcul.

2. *Fie pseudocodul de mai jos*:

   întreg n
   real p
   tablou real $x[n]$
   . . . . . .
   $p = 1$
   pentru $i = 1, n$
   $p = p \cdot \sin(x_i/y_i - z_i) \cdot \sqrt{w}$

   Se cer:
   (a) să se declare variabilele i, w, y şi z;
   (b) să se estimeze $T = O(?)$ şi $M = O(?)$. Operaţiile elementare se consideră: +, -, *, /, $\sqrt{}$, sin . Locaţia elementară de memorie este locaţia ocupată de un numărul real;
   (c) să se scrie formula matematică de calcul a variabilei p, în funcţie de datele problemei;
   (d) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze timpul de calcul.

3. *Fie pseudocodul de mai jos*:

   întreg n, i
   real rez, t, t_1, t_2
   tablou real $a[n]$
   . . . . . .
   $rez = 0$
   pentru $i = 1, n$
   $t1 = f(p) \cdot a_i$
   $t2 = f(p) \cdot f(a_i)$
   $t = t_1 + t_2$
   $rez = rez + t$

   Se cer:

   (a) să se estimeze $T = O(?)$ şi $M = O(?)$. Se consideră ca operaţie elementară evaluarea funcţiei f. Locaţia elementară de memorie este locaţia ocupată de un numărul real;

   (b) să se scrie formula matematică de calcul a variabilei rez, în funcţie de datele problemei;

   (c) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze timpul de calcul.

4. *Fie pseudocodul de mai jos*:

   întreg n, i
   real rez, t, a, b
   tablou real $p[n]$
   . . . . . .
   $rez = 0$
   pentru $i = 1, n$
   $t = f(p_i) + b$
   $t = t \cdot f(a)$
   $rez = rez + t$

   Se cer:

   (a) să se estimeze $T = O(?)$ şi $M = O(?)$. Se consideră ca operaţie elementară evaluarea funcţiei f. Locaţia elementară de memorie este locaţia ocupată de un numărul real;

   (b) să se scrie formula matematică de calcul a variabilei rez, în funcţie de datele problemei;

   (c) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze timpul de calcul.

5. *Fie pseudocodul de mai jos*:

întreg n, i
real x
tablou real $c[n][n]$
. . . . . .
pentru $i = 1, n$
  pentru $j = 1, n$
    $c_{i,j} = \sqrt{x} \cdot a_{i,j} + \ln(y) \cdot b_{i,j}$

Se cer:

(a) să se declare variabilele j, y, a şi b;

(b) să se estimeze $T = O(?)$ şi $M = O(?)$. Operaţiile elementare se consideră: +, -, *, /, $\sqrt{}$, ln . Locaţia elementară de memorie este locaţia ocupată de un numărul real;

(c) să se scrie formula matematică de calcul a variabilei c, în funcţie de datele problemei;

(d) să se optimizeze algoritmul în sensul reducerii timpului de calcul şi să se precizeze timpul de calcul.

1.8 Întrebări şi probleme

1. Să se implementeze un tip abstract de date, care să permită reprezentarea cu maximă eficienţă spaţială a polinoamelor cu mulţi coeficienţi nuli, (de exemplu $p(x) = 1 + x^{205}$).

2. Să se implementeze un tip abstract de date care să permită reprezentarea eficientă a matricelor cu multe elemente nule ("matrice rare").

3. Să se definească o funcţie care să permită calculul normei Cebîşev a unui vector

$$||v|| = \max_{i=1,n} |v_1|$$

şi a unei matrice

$$\|A\| = \max_{i=1,n} \sum_{j=1}^{n} |a_{ij}|$$

4. Să se scrie pseudocodul procedurilor de împărţire, derivare şi integrare a polinoamelor.

5. Să se definească următoarele tipuri de date: spaţiu vectorial complex ($\mathbb{C}^n$), inelul matricelor pătrate cu elemente complexe ($\mathbb{C}^{n\times n}$), inelul polinoamelor cu coeficienţi complecşi.

6. Să se scrie o procedură de înmulţire a matricelor dreptunghiulare.

7. Să se definească şi implementeze un tip abstract de date având ca elemente vectori tridimensionali $\mathbb{R}^3$, cu care să se efectueze adunări, înmulţiri cu scalari, înmulţiri scalare şi vectoriale.

8. Să se definească şi să se implementeze un tip abstrat de date având ca elemente impedanţele cu care se pot face operaţii de conexiune serie şi paralel. Să se conceapă un program capabil să determine impedanţa complexă de transfer a unui cuadripol pasiv.

9. Să se implementeze acelaşi tip abstract de date în două sau trei limbaje diferite de programare. Să se compare rezultatele din punctul de vedere al eficienţei spaţiale şi temporale.

10. Să se studieze rutinele dedicate manipulării vectorilor, matricelor, şi polinoamelor dintr-o bibliotecă matematică standard.

11. Să se definească şi implementeze un tip abstract de date ale cărui elemente sunt funcţii raţionale cu coeficienţi reali. Cum poate fi folosit un astfel de tip de date în analiza circuitelor electrice liniare în regim tranzitoriu?

12. Să se definească un tip abstract de date ale cărui elemente reprezintă serii Fourier trunchiate. Cum poate fi folosit acest tip de date în analiza cu calculatorul a circuitelor liniare în regim periodic permanent nesinusoidal?

Lucrarea 2

Erori în rezolvarea problemelor numerice

2.1 Caracterizarea lucrării

În majoritatea cazurilor, algoritmii cu caracter numeric, după implementarea lor pe un sistem de calcul determină nu soluţia exactă a problemei ci o aproximare numerică a acesteia.

Scopul acestei lucrări este de a evidenţia modul în care pot fi caracterizate erorile numerice, motivele apariţiei acestora şi modul în care acestea se propagă. Se studiază *erorile inerente* (datorate datelor de intrare), *erorile de rotunjire* (datorate reprezentării finite, aproximative a numerelor reale) şi *erorile de trunchiere* (datorate aproximării finite a unor procese teoretic infinite).

2.2 Principiul lucrării

Pentru a caracteriza abaterea unei variabile numerice de la valoarea sa exactă se poate folosi *eroarea absolută*, definită prin:

$$e_x = x - x^*, \tag{2.1}$$

în care x este valoarea exactă a variabilei, iar x^* este valoarea sa aproximativă. Deoarece în majoritatea cazurilor valoarea exactă nu este cunoscută se preferă utilizarea unei *margini superioare a erorii absolute*:

$$a_x \geq |e_x| = |x - x^*|. \tag{2.2}$$

În acest caz, în locul valorii exacte x se operează cu intervalul:

$$x^* - a_x \leq x \leq x^* + a_x. \tag{2.3}$$

O altă modalitate de caracterizare a abaterii unei variabile de la valoarea exactă este *eroarea relativă*:

$$\varepsilon_x = \frac{e_x}{x} = \frac{x - x^*}{x}, \tag{2.4}$$

sau *marginea superioară a acesteia*:

$$\Gamma_x = \frac{a_x}{|x|} \geq |eps_x| = \frac{|x - x^*|}{|x|}. \tag{2.5}$$

În majoritatea calculelor tehnice, mărimile rezultate în urma măsurării sunt cunoscute cu 3 maxim 6 cifre semnificative exacte, ceea ce corespunde unor erori relative cuprinse în intervalul $10^{-2} \div 10^{-5}$.

2.2.1 Erori de rotunjire

Una din cauzele frecvente de eroare în calculele cu numere reale se datorează reprezentării finite a acestor numere în sistemele de calcul.

Modul uzual de reprezentare a unui număr real într-un sistem de calcul este de forma:

$$x^* = \pm 0, kkk \ldots k \cdot 10^{\pm kk} = m \cdot 10^e, \tag{2.6}$$

unde k este cifră $(0, 1, \ldots, 9)$, m se numeşte mantisa, iar e exponent. Cu excepţia numărului nul, la care $m = 0$, în rest mantisa satisface relaţia:

$$0.1 \leq |m| \leq 1. \tag{2.7}$$

Presupunând că mantisa este reprezentată cu n cifre, rezultă că prin această reprezentare cifrele "l" din reprezentarea exactă:

$$x = \pm 0. \underbrace{kk \ldots k}_{n} ll \ldots \cdot 10^{\pm kk} \tag{2.8}$$

sunt pierdute prin rotunjire.

În consecinţă, eroarea relativă de rotunjire

$$\varepsilon_x = \frac{|x - x^*|}{|x|} = \frac{0, 0 \ldots 0lll \ldots \cdot 10^{\pm kk}}{0, kk \ldots ll \ldots \cdot 10^{\pm kk}} = \frac{0, lll \ldots}{0.kk \ldots} 10^{-n} = 10^{-n+1} \tag{2.9}$$

depinde de numărul de cifre semnificative folosite în reprezentarea numărului real şi nu de valoarea numărului. Această eroare relativă de rotunjire, specifică sistemului de calcul (calculator + mediu de programare) este cel mai mare număr real, care adăugat la unitate nu "modifică" valoarea acesteia. Ordinul de mărime al erorii relative de rotunjire în sistemele uzuale de calcul este $10^{-5} \div 10^{-20}$ şi poate fi determinat pe fiecare sistem de calcul cu următorul algoritm:

```
; calculează eroarea relativă de rotunjire
real err
err=1
repetă
        err=err/2
până când (1 + err = 1)
scrie err
```

Eroarea relativă de rotunjire *err* este cunoscută sub numele de *zeroul maşinii* şi nu trebuie confundată cu cel mai mic număr pozitiv, nenul, reprezentabil în calculator.

2.2.2 Erori inerente

Datele de intrare folosite în rezolvarea unei probleme tehnice provin în multe cazuri din determinări experimentale. Rezultatul oricărei măsurători este susceptibil de erori şi chiar dacă datele de intrare sunt cunoscute cu maximă precizie ele pot fi afectate de erori de rotunjire.

Erorile datelor de intrare într-un algoritm se numesc erori inerente. Aceste erori se propagă în procesul de calcul şi afectează în final soluţia problemei. Chiar dacă algoritmul de calcul nu poate fi făcut responsabil de prezenţa erorilor în datele de intrare, el poate influenţa precizia soluţiei. Un algoritm la care eroarea relativă a soluţiei nu depăşeşte erorile relative ale datelor de intrare este un *algoritm stabil din punct de vedere numeric*. În schimb, dacă erorile relative ale soluţiei sunt mult mai mari decât cele ale datelor de intrare se spune că algoritmul de rezolvare prezintă instabilităţi numerice. În acest caz este posibil ca abateri foarte mici ale datelor de intrare să determine abateri mari ale soluţiei numerice faţă de cea exactă, şi să facă soluţia numerică inutilizabilă. Dacă se noteaza cu y soluţia problemei, iar cu $x_1, x_2, ..., x_n$ datele acesteia, procedeul de calcul va consta în fond în evaluarea funcţiei $y = f(x_1, x_2, ..., x_n)$

Considerând erorile absolute suficient de mici, acestea se pot aproxima cu diferenţiala:

$$\begin{aligned} \mathrm{d}y &= \frac{\partial f}{\partial x_1}\mathrm{d}x_1 + \frac{\partial f}{\partial x_2}\mathrm{d}x_2 + \cdots + \frac{\partial f}{\partial x_n}\mathrm{d}x_n, \\ e_y &= \frac{\partial f}{\partial x_1}e_{x_1} + \frac{\partial f}{\partial x_2}e_{x_2} + \cdots + \frac{\partial f}{\partial x_n}e_{x_n}. \end{aligned} \tag{2.10}$$

Erorile relative satisfac relaţia:

$$y \cdot \varepsilon_y = \frac{\partial f}{\partial x_1}x_1 \cdot \varepsilon_{x_1} + \frac{\partial f}{\partial x_2}x_2 \cdot \varepsilon_{x_2} + \cdots + \frac{\partial f}{\partial x_n}x_n \cdot \varepsilon_{x_n}. \tag{2.11}$$

Deoarece în sistemele numerice de calcul, cele mai complicate proceduri se reduc în final la operaţii aritmetice, va fi analizată propagarea erorilor în cazul celor patru operaţii aritmetice elementare:

Adunarea:

$$\begin{aligned} y &= f(x_1, x_2) = x_1 + x_2, \\ e_y &= e_{x_1} + e_{x_2}, \\ \varepsilon_y &= \frac{x_1}{x_1 + x_2}\varepsilon_{x_1} + \frac{x_2}{x_1 + x_2}\varepsilon_{x_2}. \end{aligned} \tag{2.12}$$

Dacă cei doi termeni x_1, x_2 au acelaşi semn, adunarea este o operaţie stabilă numeric, deoarece

$$\left|\frac{x_1}{x_1 + x_2}\right| \leq 1, \quad \left|\frac{x_2}{x_1 + x_2}\right| \leq 1, \quad |\varepsilon_y| \leq |\varepsilon_{x_1}| + |\varepsilon_{x_2}|\,.$$

Scăderea:

$$\begin{aligned} y &= f(x_1, x_2) = x_1 - x_2, \\ e_y &= e_{x_1} - e_{x_2}, \\ \varepsilon_y &= \frac{x_1}{x_1 - x_2}\varepsilon_{x_1} - \frac{x_2}{x_1 - x_2}\varepsilon_{x_2}. \end{aligned} \tag{2.13}$$

Dacă x_1 şi x_2 au acelaşi semn, operaţia de scădere este instabilă numeric. Eroarea relativă a diferenţei poate depăşi cu multe ordine de mărime eroarea termenilor, prin fenomenul numit de anulare prin scădere, ca în exemplul:

$x_1 = 0,12345 \pm 1\%$; $x_2 = 0,12344 \pm 1\%$; $y = x_1 - x_2 = 0,00001 \pm 3 \cdot 10^4\%$.

Înmulţirea:

$$\begin{aligned} y &= f(x_1, x_2) = x_1 x_2 \\ e_y &= x_2 e_{x_1} + x_1 e_{x_2} \\ \varepsilon_y &= \varepsilon_{x_1} + \varepsilon_{x_2} \end{aligned} \tag{2.14}$$

Înmulţirea este o operaţie stabilă din punct de vedere numeric deoarece

$$|\varepsilon_y| \leq |\varepsilon_{x_1}| + |\varepsilon_{x_2}|.$$

Împărţirea:

$$\begin{aligned} y &= f(x_1, x_2) = \frac{x_1}{x_2}, \\ e_y &= \frac{1}{x_2}e_{x_1} - \frac{x_1}{(x_2)^2}e_{x_2}, \\ \varepsilon_y &= \varepsilon_{x_1} - \varepsilon_{x_2}. \end{aligned} \tag{2.15}$$

Şi împărţirea este o operaţie stabilă din punct de vedere numeric deoarece

$$|\varepsilon_y| \leq |\varepsilon_{x_1}| + |\varepsilon_{x_2}|.$$

În aceste evaluări ale erorilor s-a considerat că toate calculele se efectuează exact. În realitate, rezultatul este rotunjit (la numărul de cifre semnificative specifice sistemului de calcul) şi eroarea relativă a rezultatului este majorată cu eroarea relativă de rotunjire.

În cazul în care prezintă importanţă deosebită evaluarea erorii unui calcul numeric se poate utiliza următorul tip abstract de date care permite controlul propagării erorii. În acest tip de date numerele reale x sunt reprezentate prin valoarea x şi marginea erorii relative Γ_x, deci prin intervalul:

$$x^*(1-\Gamma_x) \leq x \leq x^*(1+\Gamma_x). \tag{2.16}$$

real tip inregistrare interval
  real val ;valoare
  real er ;marginea erorii relative
procedura suma (x,y,s) ;calculează s = x + y
  interval x, y, s
  $s.val = x.val + y.val$
  $s.er = err + (|x.val \cdot x.er| + |y.val \cdot y.er|)/|s.val|$
retur
procedura dif(x,y,d) ;calculează d = x - y
  interval x, y, d
  $d.val = x.val - y.val$
  $d.er = err + (|x.val \cdot x.er| + |y.val \cdot y.er|)/|d.val|$
retur
procedura prod(x,y,p) ;calculează p = x * y
  interval x, y, p
  $p.val = x.val \cdot y.val$
  $p.er = err + |x.er| + |y.er|$
retur
procedura rap(x,y,c) ;calculează c = x / y
  interval x, y, c
  $c.val = x.val/y.val$
  $c.er = err + |x.er| + |y.er|$
retur

2.2.3 Erori de trunchiere

În esenţă, metodele numerice constau în reducerea rezolvării unei probleme complicate la un număr finit de etape simple, care în fond sunt operaţii aritmetice elementare. Datorită caracterului finit al oricărui algoritm, încercarea de a rezolva probleme de analiză

matematică, ce presupun teoretic o infinitate de paşi (cum sunt de exemplu calculele limitelor) determină apariţia unor erori de metodă numite erori de trunchiere.

Calculul numeric al limitelor de şiruri, serii sau funcţii (inclusiv calculul numeric al derivatelor şi integralelor, care se reduc în fond tot la calculul limitelor) presupune trunchierea unui proces numeric infinit. Pentru a evidenţia eroarea de trunchiere se consideră un şir numeric $x_k = 1, 2, \ldots, n$ convergent către limita $x = \lim_{k\to\infty} x_k$. Conform definiţiei convergenţei, pentru orice ε există un n astfel încât $|x - x_k| \leq \varepsilon$, pentru orice $k > n$. În consecinţă, pentru o eroare ε impusă există un rang finit n, astfel încât x_n reprezintă o aproximare satisfăcătoare pentru limita x. Rezultă că, după evaluarea unui numar n suficient de mare de termeni, ultimul poate fi adoptat ca soluţie numerică $x^* = x_n$, cu o eroare de trunchiere dependentă de acest n. În general, eroarea de trunchiere este cu atât mai mică cu cât numărul de termeni calculaţi este mai mare. Modul în care eroarea de trunchiere depinde de n caracterizează viteza de convergenţă a algoritmului. Se spune că un algoritm are *viteza de convergenţă* de ordinul p, dacă există o constantă $c > 0$, astfel încât eroarea de trunchiere satisface inegalitatea:

$$|\varepsilon_n| = |x - x_n| \leq \frac{c}{n^p}, \tag{2.17}$$

pentru orice $n > 1$. Se constată că ordinul vitezei de convergenţă a unui algoritm este dat de panta dreptei ce mărgineşte superior graficul funcţiei $|\varepsilon| = f(n)$ în scară dublu logaritmică.

O categorie importantă de procese numerice infinite o reprezintă seriile Taylor. Acestea, fiind serii de puteri, pot fi utilizate la evaluarea funcţiilor elementare (sinus, cosinus, exponenţială, logaritm, etc.) şi a celor speciale (Bessel, integrale eliptice) prin reducerea la operaţii aritmetice elementare (adunări si înmulţiri).

Pentru a exemplifica modul în care poate fi sumată numeric o serie cu controlul erorii de trunchiere se consideră dezvoltarea în serie Taylor a funcţiei $y = \sin x$:

$$y = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots \tag{2.18}$$

Termenul general al seriei poate fi calculat recursiv:

$$T_k = -\frac{(-1)^k}{(2k-1)!}x^{2k-1} = -T_{k-1}\frac{x^2}{(2k-1)(2k-2)}. \tag{2.19}$$

Prin trunchierea seriei la rangul n se obţine

$$y^* = \sum_{k=1}^{n} \frac{(-1)^{k+1}}{(2k-1)!} \cdot x^{2k-1}. \tag{2.20}$$

Deoarece seria este alternantă, eroarea de trunchiere este majorată de modulul primului termen neglijat:

$$|\varepsilon_y| = |y - y^*| \leq \frac{|x|^{2n+1}}{(2n+1)!}. \tag{2.21}$$

Rezultă următorul pseudocod pentru evaluarea funcţiei sinus cu o eroare de trunchiere mai mică decât eroarea de rotunjire.

funcţia sinus(x)
    real x, t, s
    întreg k
    $t = x$
    $s = t$
    $k = 1$
    repetă
        $k = k + 2$
        $t = -tx^2/k/(k-1)$
        $s = s + t$
    până când $|t| <$err
întoarce s

Efortul de calcul pentru evaluarea funcţiei creşte odată cu scăderea erorii impuse, dar şi cu creşterea modulului variabilei x. Deoarece funcţia $\sin(x)$ este periodică, algoritmul poate fi îmbunătăţit prin reducerea variabilei la primul cerc (sau chiar la primul cadran).

2.3 Chestiuni de studiat

1. Determinarea erorii relative de rotunjire a sistemului de calcul;
2. Analiza propagării erorilor inerente;
3. Analiza erorii de trunchiere;
4. Implementarea unor algoritmi numerici care controlează eroarea.

2.4 Modul de lucru

Pentru desfăşurarea lucrării lansaţi lucrarea *Erori în rezolvarea problemelor numerice* din meniul general. Aceasta are ca urmare lansarea următorului meniu:

- Eroarea de rotunjire
- Calcule cu controlul erorii
- Erori de trunchiere

2.4.1 Determinarea erorii relative de rotunjire a sistemului de calcul

Se selectează opţiunea *Eroarea de rotunjire* care are ca efect calculul şi afişarea erorii relative de rotunjire specifice sistemului.

2.4.2 Analiza propagării erorilor inerente

Se selectează opţiunea *Calcule cu controlul erorii* din meniul principal, care are ca efect lansarea unui program ce emulează un calculator de buzunar care foloseşte notaţia poloneză inversă (operand, operand, operator). Acest calculator nu operează cu numere reale ci cu intervale (valoare numerică şi margine superioară a erorii relative).

Programul permite introducerea fie a operanzilor (valoare şi eroare relativă) fie a operatorilor unari (I - invers, O - opus, V - radical) sau binari (+ - * /).

Operanzii sunt introduşi într-o stivă vizualizată pe ecran de 5 coloane (număr de ordine, valoare numerică, eroare relativă, limita minimă şi limita maximă a valorii exacte).

Operatorii unari acţionează asupra ultimului element introdus în stivă, înlocuindu-l cu valoarea obţinută în urma aplicării operatorului (stiva nu îşi modifică dimensiunea).

Operatori binari folosesc ca operanzi ultimele două elemente din stivă, iar rezultatul înlocuieşte aceste elemente (stiva se micşorează cu o unitate).

Pentru a facilita manipularea stivei sunt disponibili încă doi operatori unari, cu caracter nenumeric (R - repetă vârful stivei şi E - extrage element din stivă), care modifică (incrementează respectiv decrementează) înălţimea stivei.

Se propune rezolvarea ecuaţiei de gradul doi:

$$ax^2 + bx + c = 0,$$

în care $a = 1$, $b = -100.01$, $c = 1$ (cu rădăcinile $x_1 = 100$; $x_2 = 0,01$). Valorile a, b, c se vor introduce cu erorile relative de $0.1\% = 0.001$. Se vor nota valorile intermediare:

$$\begin{aligned} \Delta &= b^2 - 4ac, \\ x_1 &= \frac{-b + \sqrt{\Delta}}{2a}, \\ x_2 &= \frac{-b - \sqrt{\Delta}}{2a}, \end{aligned}$$

şi erorile relative asociate.

Pentru a evita erorile datorate anulării prin scădere, se va calcula a doua rădăcină pe baza relaţiei $x_1 x_2 = c/a$

$$x_2 = \frac{c}{ax_1}.$$

Se vor compara erorile relative şi absolute ale rezultatelor obţinute pe cele două căi.

2.4.3 Analiza erorii de trunchiere

Se selectează opţiunea *Erori de trunchiere* din meniul principal, care are ca efect lansarea unui program care realizează sumarea următoarelor serii:

- Taylor pentru funcţia exponenţială
- Taylor pentru funcţia sinus
- Taylor pentru funcţia logaritm natural
- Fourier pentru funcţia crenel
- armonica alternantă
- armonica alternantă cu convergenţă îmbunătăţită prin metoda Euler

După selectarea seriei şi alegerea valorii variabilei independente x, programul calculează şi afişează suma parţială a seriei şi eroarea absolută de trunchiere pentru diferite ordine, apoi reprezintă grafic variaţia erorii în funcţie de numărul de termeni sumaţi. Pentru a putea facilita comparaţiile se pot reprezenta şi mai multe grafice de erori simultan.

Se recomandă observarea convergenţei

- seriilor Taylor pentru evaluarea expresiilor:
  - exp(0), exp(1), exp(5), exp(-10), exp(10),
  - sin(0), sin(1), sin(10),
  - ln(1), ln(0.1), ln(1.1).

  Reamintim că dezvoltările în serie Taylor sunt:

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots \\ \sin(x) &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots \\ \ln(x) &= (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \cdots \end{aligned}$$

- seriei Fourie pentru evaluarea expresiilor:
  - crenel(1), crenel(0.1).

Funcţia crenel este definită astfel

$$\text{crenel}\,(x) = \begin{cases} \frac{\pi}{4} & \text{dacă}\,\sin(x) > 0 \\ -\frac{\pi}{4} & \text{dacă}\,\sin(x) \leq 0 \end{cases}$$

iar dezvoltarea ei în serie Fourier conduce la

$$\text{crenel}\,(x) = \sin(x) + \frac{\sin(3x)}{3} + \frac{\sin(5x)}{5} + \cdots$$

- armonicii alternante

$$A = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \ldots = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{1}{k}$$

- armonicii alternante cu convergenţă îmbunătăţită

$$A = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \ldots = 0.5 + \sum_{k=1}^{\infty} \frac{\left(\frac{1}{2k-1} - \frac{1}{2k+1}\right)}{4k}.$$

Se vor comenta rezultatele şi se va aprecia viteza de convergenţă.

Indicaţie: Pentru estimarea vitezei de convergenţă se calculează panta dreptei ce mărgineşte graficul erorii, în scara dublu logaritmică.

2.4.4 Implementarea unor algoritmi cu controlul erorii

Se va genera pseudocodul funcţiei cosinus, determinată prin serie Taylor trunchiată până la atingerea unei erori de trunchiere impuse. Se va implementa acest algoritm într-un limbaj de programare sub forma unei funcţii cosinus(x, ert), în care x este variabilă independentă iar ert este eroarea de trunchiere impusă. Se va implementa algoritmul de determinare al erorii relative de rotunjire err, iar în final se va implementa algoritmul de evaluare a funcţiei cosinus cu $ert = err$ şi cu controlul propagării erorii, sub forma unei proceduri care să calculeze cos(x) şi eroarea relativă asociată.

2.5 Exemple

2.5.1 Exemple rezolvate

1. Să se calculeze o margine a erorii relative pentru aproximarea numărului $\sqrt{2}$ cu 2 cifre semnificative exacte.

   Rezolvare:

Notăm cu $x = \sqrt{2} = 1.4142\ldots$, valoarea exactă a numărului $\sqrt{2}$. Aproximarea cu 2 cifre semnificative exacte a lui $\sqrt{2}$ este: $x^* = 1.41$.

Eroarea absolută este: $e_x = x - x^* = 1.4142\ldots - 1.41 = 0.0042\ldots$.

Eroarea relativă este:

$$\varepsilon_x = \frac{e_x}{x} = \frac{0.0042\ldots}{1.4142\ldots},$$

care este mărginită de:

$$|\varepsilon_x| < \frac{0.005}{1.4} = 0.0035\ldots < 0.0036 = 0.36\%.$$

Deci: $\sqrt{2} = 1.41 \pm 0.36\%$.

2. Fie $x = 5.43 \pm 1\%$, $y = -5.42 \pm 2\%$, $z = 3 \pm 1\%$.

   Să se calculeze $(x + y) \cdot z$. Comentaţi rezultatul obţinut.

   Rezolvare:

   Mai întâi se efectuează operaţia din paranteze, care este, de fapt, o scădere: $x + y = 5.43 - 5.42 = 0.01$. Eroarea relativă va fi mărginită de:

$$\Gamma_{x+y} = \left|\frac{x}{x+y}\right|\Gamma_x + \left|\frac{y}{x+y}\right|\Gamma_y = \frac{5.43}{0.01}\cdot\frac{1}{100} + \frac{5.42}{0.01}\cdot\frac{2}{100} =$$

$$= 5.43 + 10.84 = 16.27 = 1627\%.$$

   Eroarea este foarte mare deoarece au fost scăzute două numere foarte apropiate (a apărut fenomenul de anulare prin scădere).

   Rezultatul final $(x + y) \cdot z = 0.03$ este calculat cu o eroare relativă mărginită de:

$$\Gamma_{(x+y)\cdot z} = \Gamma_{x+y} + \Gamma_z = \frac{1627}{100} + \frac{1}{100} = \frac{1628}{100} = 1628\%.$$

3. Fie $x = 2.3 \pm 2\%$, $y = 3.2 \pm 3\%$, $z = 5.5 \pm 5\%$.

   Să se calculeze $(x + y) + z$ si $x + (y + z)$. Comentaţi rezultatele obţinute.

   Rezolvare:

   Să calculăm $(x + y) + z$. Mai întâi se efectuează operaţia din paranteze: $x + y = 2.3 + 3.2 = 5.5$. Eroarea relativă va fi mărginită de:

$$\Gamma_{x+y} = \left|\frac{x}{x+y}\right|\Gamma_x + \left|\frac{y}{x+y}\right|\Gamma_y; \tag{2.22}$$

$$\Gamma_{x+y} = \frac{2.3}{5.5}\cdot\frac{2}{100} + \frac{3.2}{5.5}\cdot\frac{3}{100} = \frac{2.58}{100} = 2.58\%.$$

Rezultatul final $(x+y)+z = 5.5+5.5 = 11.0$ este calculat cu o eroare relativă mărginită de:

$$\Gamma_{(x+y)+z} = \left|\frac{x+y}{(x+y)+z}\right|\Gamma_{x+y} + \left|\frac{z}{(x+y)+z}\right|\Gamma_z; \tag{2.23}$$

$$\Gamma_{(x+y)+z} = \frac{5.5}{11}\cdot\frac{14.2}{5.5}\cdot\frac{1}{100} + \frac{5.5}{11}\cdot\frac{5}{100} = \frac{3.79}{100} = 3.79\%.$$

In continuare calculăm $x+(y+z)$. Mai întâi se efectuează operaţia din paranteze $y+z = 3.2+5.5 = 8.7$. Eroarea relativă va fi mărginită de:

$$\Gamma_{y+z} = \left|\frac{y}{y+z}\right|\Gamma_y + \left|\frac{z}{y+z}\right|\Gamma_z; \tag{2.24}$$

$$\Gamma_{y+z} = \frac{3.2}{8.7}\cdot\frac{3}{100} + \frac{5.5}{8.7}\cdot\frac{5}{100} = \frac{4.26}{100} = 4.26\%.$$

Rezultatul final $x+(y+z) = 2.3+8.7 = 11.0$ este calculat cu o eroare relativă mărginită de:

$$\Gamma_{x+(y+z)} = \left|\frac{x}{x+(y+z)}\right|\Gamma_x + \left|\frac{y+z}{x+(y+z)}\right|\Gamma_{y+z}; \tag{2.25}$$

$$\Gamma_{x+(y+z)} = \frac{2.3}{11}\cdot\frac{2}{100} + \frac{8.7}{11}\cdot\frac{37.1}{8.7}\cdot\frac{1}{100} = \frac{3.79}{100} = 3.79\%.$$

Observăm că:

$$\Gamma_{(x+y)+z} = \Gamma_{x+(y+z)}.$$

Într-adevăr, înlocuind relaţia (2.22) în relaţia (2.23), obţinem:

$$\Gamma_{(x+y)+z} = \left|\frac{x}{x+y+z}\right|\Gamma_x + \left|\frac{y}{x+y+z}\right|\Gamma_y + \left|\frac{z}{x+y+z}\right|\Gamma_z. \tag{2.26}$$

Similar, înlocuind (2.24) în (2.25), obţinem exact acelaşi rezultat, (2.26).

În realitate, în aceste calcule nu am ţinut cont de erorile de rotunjire *eps*. Acestea se suprapun peste erorile inerente, astfel încât este mai corect să scriem:

$$\Gamma_{x+y} = \left|\frac{x}{x+y}\right|\Gamma_x + \left|\frac{y}{x+y}\right|\Gamma_y + eps. \tag{2.27}$$

Urmează că:

$$\Gamma_{(x+y)+z} = \left|\frac{x+y}{(x+y)+z}\right|\Gamma_{x+y} + \left|\frac{z}{(x+y)+z}\right|\Gamma_z + eps. \tag{2.28}$$

Înlocuind (2.27) în (2.28), rezultă că:

$$\Gamma_{(x+y)+z} = \left|\frac{x}{x+y+z}\right|\Gamma_x + \left|\frac{y}{x+y+z}\right|\Gamma_y + \left|\frac{z}{x+y+z}\right|\Gamma_z + \left|\frac{x+y}{x+y+z}\right| eps + eps.$$

Similar se obţine:

$$\Gamma_{x+(y+z)} = \left|\frac{x}{x+y+z}\right|\Gamma_x + \left|\frac{y}{x+y+z}\right|\Gamma_y + \left|\frac{z}{x+y+z}\right|\Gamma_z + \left|\frac{y+z}{x+y+z}\right| eps + eps.$$

În consecinţă:

$$\Gamma_{(x+y)+z} \neq \Gamma_{x+(y+z)}.$$

Adunarea numerelor reale pe un sistem de calcul nu este asociativă.

2.5.2 Exemple propuse

1. Să se calculeze o margine a erorii relative pentru aproximarea numărului π cu 3 cifre semnificative exacte.
2. Să se calculeze o margine a erorii relative pentru aproximarea numărului e cu 2 cifre semnificative exacte.
3. Fie $x = 2.01 \pm 1\%$, $y = 4.24 \pm 4\%$, $z = 2.12 \pm 2\%$.

   Să se calculeze $x - y/z$. Comentaţi rezultatul obţinut.
4. Fie $x = 3.12 \pm 1\%$, $y = 2 \pm 1\%$, $z = 1.21 \pm 2\%$.

   Să se calculeze $x * y + z$.

2.6 Întrebări şi probleme

1. Cum poate fi caracterizată eroarea unei variabile vectoriale x?
2. Implementaţi algoritmul de determinare a erorii relative de rotunjire în diferite limbaje de programare (în simplă şi dublă precizie) şi apoi comparaţi rezultatele.
3. Ce modificări trebuie aduse algoritmului de determinare a erorii relative de rotunjire pentru ca acesta să determine nu numai ordinul de mărime al erorii ci şi valoarea sa exactă?
4. Definiţi şi implementaţi un tip abstract de date, care să reprezinte fiecare număr real ca o pereche de numere reale ce îl încadrează (valoare maximă, valoare minimă).

5. Este adunarea numerelor reale rotunjite o operaţie asociativă? Pentru a aduna mai multe numere reale diferite cu eroare minimă ele trebuie sortate în ordine crescătoare sau descrescătoare?

6. Pentru a micşora erorile de rotunjire la sumarea unei serii cum trebuie adunaţi termenii, în ordine crescătoare sau descrescătoare?

7. Folosiţi opţiunea *Calcule cu controlul erorii* pentru a determina rezistenţa de şunt ce trebuie folosită la transformarea unui galvanometru de 1mA (în clasa 2%) a cărui rezistenţă $R = 57.5\Omega$ a fost determinată cu precizie de 0,5(%) într-un ampermetru de 1A.

8. Generaţi un algoritm pentru rezolvarea unei ecuaţii de gradul doi, care evită fenomenul de anulare prin scădere.

9. Generaţi şi implementaţi un algoritm, pentru evaluarea funcţiilor Bessel.

Brook Taylor (1685, Anglia - 1731, Anglia)

http://www-groups.dcs.st-and.ac.uk/ history/Mathematicians/Taylor.html

Jean Baptiste Joseph Fourier (1768, France - 1830, France)

http://www-groups.dcs.st-and.ac.uk/ history/Mathematicians/Fourier.html

Lucrarea 3

Rezolvarea sistemelor de ecuaţii liniare prin metoda Gauss. Strategii de pivotare în rezolvarea sistemelor algebrice liniare

3.1 Metoda Gauss fără pivotare

3.1.1 Caracterizarea metodei

Metoda Gauss este o metodă directă de rezolvare a sistemelor de ecuaţii algebrice liniare care au matricea sistemului pătrată şi nesingulară. Soluţia sistemului se obţine după un număr finit de paşi.

În lucrare se prezintă principiul metodei şi se analizează algoritmul atât din punctul de vedere al complexităţii cât şi al stabilităţii numerice, evidenţiindu-se limitele acestei metode.

3.1.2 Principiul metodei

Fie sistemul de ecuaţii:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \ldots\ldots & \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned} \tag{3.1}$$

sau în forma matriceală

$$Ax = b \tag{3.2}$$

A fiind matricea pătrată $n \times n$ - dimensională, iar b si x doi vectori n - dimensionali. Cunoscând matricea sistemului A şi termenul liber b se pune problema determinării soluţiei x.

Înmulţind prima ecuaţie, pe rând, cu $-a_{i1}/a_{11}$, pentru $i = 2, \ldots, n$, apoi adunând-o la ecuaţia cu numărul i, se elimină x_1 din toate ecuaţiile, cu excepţia primei ecuaţii:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \ldots\ldots & \\ a_{n2}x_2 + \ldots + a_{nn}x_n &= b_n \end{aligned} \tag{3.3}$$

Coeficienţii care apar în linile 2, ..., n, deşi modificaţi, s-au notat în acelaşi mod, deoarece ei se vor memora în aceleaşi locaţii de memorie.

Înmulţind a doua ecuaţie cu a_{i2}/a_{22} şi adunând-o apoi la ecuaţia i, pentru $i = 3, \ldots, n$, se elimina x_2 din toate ecuaţiile aflate sub linia a doua.

Se continuă astfel până se aduce sistemul la forma:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &= b_1 \\ a_{22}x_2 + \ldots + a_{2n}x_n &= b_2 \\ \ldots\ldots & \\ a_{nn}x_n &= b_n \end{aligned} \tag{3.4}$$

unde $a_{11}, \ldots, a_{nn}$, $b_1, \ldots, b_n$ au alte valori decât în sistemul iniţial.

Necunoscutele se determină prin retrosubstituţie:

$$\begin{aligned} x_n &= \frac{b_n}{a_{nn}}, \\ x_{n-1} &= \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}, \\ &\ldots\ldots \\ x_k &= \frac{b_k - \sum_{j=k+1}^{n} a_{kj}x_j}{a_{k,k}}, \\ &\ldots\ldots \\ x_1 &= \frac{b_1 - \sum_{j=2}^{n} a_{1,j}x_j}{a_{11}}. \end{aligned} \tag{3.5}$$

Etapa de aducere a matricei A la forma triunghiulară se numeşte *eliminare* sau *triangularizare*, iar etapa de determinare a soluţiilor - *substituţie regresivă* sau *retrosubstituţie*. Fiecare din elementele a_{kk}, cu $k = 1, 2, \ldots, n$ se numeşte *pivot*.

Dacă la o etapă a algoritmului se întâlneşte un pivot nul, pentru a evita o împărţire la 0 este necesară o permutare a liniilor (sau a liniilor şi coloanelor) în scopul aducerii pe linia i, coloana i a unui element nenul. Această operaţie poartă numele de *pivotare*.

3.1.3 Pseudocodul algoritmului Gauss fără pivotare

procedura gauss_fp(n, a, b, x)
  tablou real $a(n,n), b(n), x(n)$
  ;eliminare
  pentru $k = 1, n-1$;etapele eliminării
    pentru $i = k+1, n$;parcurge coloana de sub pivot
      $p = a_{ik}/a_{kk}$
      pentru $j = k+1, n$;parcurge linia i, la dreapta
        $a_{ij} = a_{ij} - a_{kj}p$;pivotul
      $b_i = b_i - b_k p$
  ;retrosubstituţie
  $x_n = b_n/a_{nn}$
  pentru $i = n-1, 1, -1$
    $s = b_i$
    pentru $j = n, i+1, -1$
      $s = s - a_{ij}x_j$
    $x_i = s/a_{ii}$
retur

Procedura are parametrii:

- de intrare:
  - n = dimensiunea sistemului;
  - $a(n,n)$ = matricea sistemului;
  - $b(n)$ = vectorul termenilor liberi;
- de ieşire:
  - $x(n)$ = vectorul soluţie.

Matricea A este adusă la forma superior triunghiulară după $n-1$ paşi. La fiecare pas k, linia k şi cele de deasupra ei, colona k şi cele din stânga ei nu sunt afectate.

3.1.4 Analiza complexităţii

Efort de calcul

Pentru trecerea de la pasul k la pasul $k+1$ se efectuează:

- $(n-k)(n-k+1)$ înmulţiri,
- $(n-k)(n-k+1)$ adunări,
- $(n-k)$ împărţiri.

În consecinţă, pentru trecerea de la matricea iniţială la forma superior triunghiulară, se vor efectua:

- $\sum_{k=1}^{n-1}(n-k)(n-k+1) = n(n^2-1)/3$ înmulţiri,
- $\sum_{k=1}^{n-1}(n-k)(n-k+1) = n(n^2-1)/3$ adunări,
- $\sum_{k=1}^{n-1}(n-k) = n(n-1)/2$ împărţiri.

În etapa de retrosubstituţie se mai efectuează:

- $\sum_{i=1}^{n-1}(n-i) = n(n-1)/2$ înmulţiri,
- $\sum_{i=1}^{n-1}(n-i) = n(n-1)/2$ adunări,
- n împărţiri.

Deci, în total se efectuează:

- $n(n-1)(2n+5)/6$ înmulţiri,
- $n(n-1)(2n+5)/6$ adunări,
- $n(n+1)/2$ împărţiri.

Deoarece, în general, timpul necesar calculatorului pentru a efectua o înmulţire sau o împarţire este aproximativ egal cu cel consumat pentru o adunare, efortul de calcul este $2n(2n^2+6n-2)/6$ operaţii aritmetice elementare.

Dacă n este mare ($n > 2$), efortul de calcul este de ordinul $2n^3/3$ şi se notează $O(2n^3/3)$.

Necesarul de memorie

Pentru memorarea matricei A şi a vectorilor b şi x sunt necesare: $n^2+n+n = n(n+2)$ locaţii de memorie, fiecare locaţie de memorie fiind rezervată unui număr real.

3.1.5 Analiza erorilor

Rulat pe un calculator ipotetic de precizie infinită, algoritmul Gauss permite determinarea soluţiei exacte a sistemului. Cu alte cuvinte acest algoritm nu introduce eroare de metodă.

Pot apare însă erori de rotunjire, datorate preciziei finite a echipamentului de calcul. Aceste erori cresc odată cu creşterea dimensiunii n a sistemului.

Metoda Gauss în această variantă (fără pivotare) poate genera instabilităţi numerice chiar dacă nu se întâlneşte vreun element a_{ii} nul, din cauza erorilor de rotunjire.

Pentru evaluarea erorii se poate folosi reziduul

$$r = Ax - b, \tag{3.6}$$

mai precis norma lui. Această normă poate fi evaluată euclidian:

$$\|r| = \sqrt{\sum_k r_k^2} \tag{3.7}$$

sau în sens Cebîşev:

$$\|r\| = \max \|r_k\|, \quad k = 1, \ldots, n. \tag{3.8}$$

Eroarea relativă se defineşte ca

$$\varepsilon_r = \|r\|/\|b\|. \tag{3.9}$$

Norma reziduului definită anterior, nu trebuie confundată cu norma erorii adevărate:

$$e = \|x_{\text{adevărat}} - x\|. \tag{3.10}$$

Eroarea relativă adevărată se defineşte ca:

$$\varepsilon = e/\|x_{\text{adevărat}}\|. \tag{3.11}$$

Calculul erorii (3.10) ar necesita însă cunoaşterea soluţiei exacte a sistemului. De aceea, în practică se face doar o *evaluare* a erorii, folosind norma reziduului, care însă poate avea o valoare mai mare sau mai mică decât eroarea reală.

3.2 Strategii de pivotare în rezolvarea sistemelor algebrice liniare

3.2.1 Caracterizarea metodei

Pivotarea este o metodă care permite extinderea domeniului de aplicabilitate a metodei Gauss şi îmbunătăţirea stabilităţii ei numerice.

În acest paragraf se analizează diferite strategii de pivotare şi efectele lor asupra erorii cu care se obţine soluţia.

3.2.2 Principiul metodei

La rezolvarea unui sistem de ecuaţii prin metode directe, de exemplu prin metoda Gauss, poate apare situaţia în care un element pivot a_{kk} este nul, chiar dacă matricea sistemului este nesingulară. Metodele directe necesitând împărţirea la pivot, sistemul nu se poate rezolva în forma dată.

Permutarea liniilor şi/sau a coloanelor astfel încât să se aducă pe poziţia diagonală (k, k), un element nenul se numeşte *pivotare*. Pentru a obţine o bună stabilitate numerică (erori de rotunjire minime) se preferă alegerea ca pivot a unui element de modul maxim.

Se pot utiliza trei strategii de pivotare: parţială, totală sau diagonală.

Pivotarea parţială constă în căutarea pivotului de modul maxim printre toţi coeficienţii aflaţi sub linia k şi pe coloana k şi aducerea acestui element pe poziţia (k, k) prin permutări de linii (ceea ce este echivalent cu schimbarea ordinii ecuaţiilor).

Pivotarea totală constă în căutarea pivotului de modul maxim printre toţi coeficienţii aflaţi sub linia k şi la dreapta coloanei k şi aducerea acestui element pe poziţia (k, k) prin permutări de linii şi de coloane (se schimbă atât ordinea liniilor cât şi cea a coloanelor).

Pivotarea diagonală constă în căutarea pivotului optim doar printre elementele diagonale a_{ii}, cu $i \geq k$. Aducerea acestui element pe poziţia (k, k) se face, ca şi la pivotarea totală, prin permutări de linii şi coloane. Avantajul acestei strategii de pivotare este păstrarea simetriei matricei sistemului.

La metoda cu pivotare parţială, unde se fac doar permutări de linii, ordinea necunoscutelor în fiecare ecuaţie nu se modifică. La metodele cu pivotare totală sau diagonală, datorită permutării coloanelor, se modifică şi ordinea necunoscutelor în sistem. De aceea, aceste strategii impun memorarea permutărilor de coloane făcute pe parcursul algoritmului, pentru a se putea reconstitui la sfârşit ordinea iniţială a necunoscutelor.

3.2.3 Pseudocodul algoritmului Gauss cu pivotare parţială

```
funcţia gausspp (n, a, b, x)
    tablou real a(n,n), b(n), x(n)
    ; eliminare                          ; etapele eliminării
    pentru k = 1, n-1                    ; găsire pivot
        max = k
        pentru m = k+1, n
```

```
            dacă |a_mx| > |a_max,k| atunci max = m
        dacă a_max,k = 0 atunci întoarce 1          ; eroare
        dacă max ≠ k atunci                         ; permută liniile max, k
            pentru m = k, n
                temp = a_km
                a_km = a_max,m
                a_max,m = temp
            temp = b_k                              ; permută termenii liberi
            b_k = b_max
            b_max = temp                            ; s-a găsit pivotul
        pentru i = k + 1, n                         ; parcurge coloana de sub pivot
        dacă a_ik ≠ 0 atunci
            p = a_ik / a_kk
            pentru j = k + 1, n                     ; parcurge linia i
                a_ij = a_ij − a_kj · p              ; la dreapta pivotului
            b_i = b_i − b_k · p
; retrosubstituţie
dacă a_n,n = 0 atunci întoarce 1                    ; eroare
x_n = b_n / a_n,n
pentru i = n − 1, 1, −1
    s = b_i
    pentru j = n, i + 1, −1
        s = s − a_ij · x_j
    x_i = s / a_ii
întoarce 0
```

Funcţia are parametrii:

- de intrare:

  n = dimensiunea sistemului;
  $a(n, n)$ = matricea sistemului;
  $b(n)$ = vectorul termenilor liberi.

- de ieşire:

  $x(n)$ = vectorul soluţiei.

Funcţia întoarce:

0 - dacă algoritmul s-a executat integral;
1 - dacă matricea sistemului este singulară.

3.2.4 Analiza complexităţii

Efort de calcul

Dacă se neglijează timpul consumat datorită permutărilor de linii, efortul de calcul la procedura Gauss cu pivotare este de ordinul $O(2n^3/3)$, ca şi la cea fără pivotare.

Necesarul de memorie

Pentru memorarea matricei **A** şi a vectorilor **b** şi **x** sunt necesare:

$$n^2 + n + n = n(n + 2)$$

locaţii, fiecare locaţie de memorie fiind rezervată unui număr real.

3.2.5 Analiza erorilor

Erorile se calculează la fel ca la algoritmul Gauss fără pivotare. Deşi stabilitatea numerică a algoritmului cu pivotare este în general mai bună, totuşi, erorile de rotunjire pot afecta grav soluţia în cazul sistemelor slab condiţionate.

3.3 Chestiuni de studiat

1. Rezolvarea unor sisteme algebrice liniare;
2. Analiza experimentală a algoritmilor;
3. Implementarea algoritmilor într-un limbaj de programare;
4. Căutare de informaţii legate de această temă.

3.4 Modul de lucru

Se selectează lucrarea *Rezolvarea sistemelor cu metoda Gauss* din meniul principal. Aceasta are ca efect lansarea următorului meniu:

- Rezolvare sisteme de ecuaţii
- Analiza algoritmilor

3.4.1 Rezolvarea unor sisteme algebrice liniare

Se selectează opţiunea *Rezolvare sisteme de ecuaţii*.

Se vor introduce parametrii necesari pentru rezolvarea următoarelor sisteme:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 + x_5 = 9 \\ 2x_1 + x_2 + x_3 + 3x_4 + x_5 = 12 \\ x_1 + x_2 + 3x_3 + x_4 + 2x_5 = 17 \\ 2x_1 + 3x_2 + x_3 + 2x_4 + x_5 = 15 \\ x_1 + x_2 + x_3 + x_4 + 4x_5 = 15 \end{cases} \quad R: \begin{array}{l} x_1 = 1 \\ x_2 = 2 \\ x_3 = 3 \\ x_4 = 1 \\ x_5 = 2 \end{array}$$

$$\begin{cases} x_1 + x_2 - 2x_3 = -3 \\ 3x_1 + x_3 = 10 \\ x_1 + 2x_2 + x_3 = 12 \end{cases} \quad R: \begin{array}{l} x_1 = 2 \\ x_2 = 3 \\ x_3 = 4 \end{array}$$

$$\begin{cases} y + z = 4 \\ 2x + y + 2z = 11 \\ x + 2y + z = 7 \end{cases}$$

$$\begin{cases} x + 2y + z = 7 \\ x + 2y + 2z = 10 \\ 2x + 4y + 3z = 11 \end{cases}$$

$$\begin{cases} mx_1 + 1000x_2 = 1000 \\ 1000x_1 = 1000 \end{cases}$$

unde m ia valorile: 1000, 100, 10, 1, 10^{-1}, 10^{-4}, 10^{-5}, 10^{-6}.

Se vor selecta pe rând metoda fără pivotare şi apoi cu pivotare. Se vor comenta rezultatele obţinute.

3.4.2 Analiza experimentală a algoritmului

Se selectează optiunea *Analiza algoritmilor*.

Astfel, se apelează un program care rezolvă sisteme de ecuaţii liniare de diferite mărimi prin metoda Gauss fără pivotare, Gauss cu pivotare parţială şi Gauss cu pivotare totală.

Parametrul de intrare al programului este dimensiunea sistemului, n. După preluarea acestui parametru programul generează aleator un sistem de n ecuaţii şi n necunoscute, îl rezolvă şi afişează: eroarea absolută, eroarea relativă şi reziduul rezultatului, calculate fiecare cu formula normei Euclidiene, sau a normei Cebîşev (în total 6 valori). Se afişează de asemenea timpul de calcul.

Pentru a simplifica operaţia de culegere a datelor numerice, programul cere valoarea iniţială a lui n, valoarea finală şi pasul. Se recomandă folosirea următoarelor valori: valoare iniţială 10, valoare finală 100, pas 10, ceea ce corespunde şirului de valori $n = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$.

Rezultatele numerice sunt afişate în consola Scilab şi reprezentate grafic.

Se vor nota pentru fiecare n şi fiecare metodă eroarea relativă (norma Euclidiană şi norma Cebîsev) şi reziduul relativ (norma euclidiană), precum şi timpul de calcul.

n		10	20	·····	90	100
timp [s]	G.f.p G.p.p G.p.t					
er.rel (Euclid)	G.f.p G.p.p.					
er.rel (Cebîşev)	G.f.p G.p.p.					
reziduu rel. (Euclid)	G.f.p G.p.p.					

Se vor reprezenta pe hârtie milimetrică:

- curba timpului de calcul în funcţie de dimensiunea sistemului n;
- curbele erorilor de calcul în funcţie de n.

3.4.3 Implementarea algoritmilor într-un limbaj de programare

Se va implementa o procedură proprie de rezolvare a unui sistem de ecuaţii prin metoda Gauss fără pivotare. Se va compila procedura eliminîndu-se eventualele erori.

Se va scrie pseudocodul şi se va implementa pe calculator un program care apelează procedura de la punctul anterior şi rezolvă sistemul de ecuaţii:

$$\begin{cases} 2x + y + z = 7 \\ x + y + z = 6 \\ 2x + y + 3z = 13 \end{cases} \qquad R: \begin{array}{l} x = 1 \\ y = 2 \\ z = 3 \end{array}$$

Acest program va permite introducerea datelor de la consolă şi afişarea pe ecran a soluţiei. Se vor nota şi comenta rezultatele obţinute, eventualele dificultăţi apărute pe parcursul lucrului.

3.4.4 Căutare de informaţii

Căutaţi pe Internet informaţii (coduri) legate de metoda Gauss pentru rezolvarea directă a sistemelor de ecuaţii liniare. Exemple de cuvinte cheie: *Gauss elimination and back substitution.*

3.5 Exemple

3.5.1 Exemple rezolvate

1. Fie sistemul de ecuaţii:

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

Să se rezolve sistemul de ecuaţii prin metoda Gauss.

Rezolvare:

Deoarece matricea coeficienţilor sistemului de ecuaţii este pătrată şi nesingulară, metoda Gauss se poate aplica pentru rezolvarea sistemului de ecuaţii.

Metoda Gauss constă în etapa de eliminare şi etapa de retrosubstituţie.

În **etapa de eliminare** (sau triangularizare) matricea coeficienţilor sistemului de ecuaţii se aduce la forma superior triunghiulară.

Pentru triangularizarea matricii, prima ecuaţie (linie, notată L_1 în c ele ce urmează) se înmulţeşte cu $-(2/1)$ şi se adună la a doua ecuaţie (linie, L_2) pentru a se elimina termenul în x din a doua ecuaţie. De notat că, după acest pas, prima ecuaţie nu se modifică, doar a două ecuaţie suferă modificări, conform relaţiei $L_2' = L_2 + (-2) \cdot L_1$.

$$\begin{array}{l} L_1 \\ L_2 \\ L_3 \end{array} \left\{ \begin{array}{rcrcrcll} x & + & 2y & - & z & = & 4 & \mid \cdot (-\frac{2}{1}) \\ 2x & + & 2y & - & 3z & = & 2 & \\ -3x & + & 2y & + & 2z & = & 2 & \end{array} \right. \quad \overset{L_2' = L_2 + (-2) \cdot L_1}{\Longrightarrow}$$

$$\begin{array}{l} L_1 \\ L_2' \\ L_3 \end{array} \left\{ \begin{array}{rcrcrcr} x & + & 2y & - & z & = & 4 \\ & - & 2y & - & z & = & -6 \\ -3x & + & 2y & + & 2z & = & 2 \end{array} \right.$$

În continuare, prima ecuaţie (L_1) se înmulţeşte cu $-[(-3)/1]$ şi se adună la a treia ecuaţie (L_3), astfel eliminându-se termenul în x şi din a treia ecuaţie ($L_3' = L_3 + 3 \cdot L_1$).

$$\begin{array}{l} L_1 \\ L_2' \\ L_3 \end{array} \left\{ \begin{array}{rcrcrcrl} x & + & 2y & - & z & = & 4 & \left| \cdot (-\frac{-3}{1}) \right. \\ & - & 2y & - & z & = & -6 & \\ -3x & + & 2y & + & 2z & = & 2 & \end{array} \right. \overset{L_3'=L_3+3\cdot L_1}{\Longrightarrow}$$

$$\begin{array}{l} L_1 \\ L_2' \\ L_3' \end{array} \left\{ \begin{array}{rcrcrcr} x & + & 2y & - & z & = & 4 \\ & - & 2y & - & z & = & -6 \\ & + & 8y & - & z & = & 14 \end{array} \right.$$

Se observă că elementele de sub elementul diagonal din prima ecuaţie sunt nuli.

Mai departe, se urmăreşte anularea elementelor de sub elementul diagonal din a doua ecuaţie.

Astfel, a doua ecuaţie (L_2') se înmulţeşte cu $-[8/(-2)]$ şi se adună la a treia ecuaţie (L_3'), eliminându-se termenul în y din a treia ecuaţie ($L_3'' = L_3' + 4 \cdot L_2'$). De notat că ecuaţia a doua (L_2') nu suferă modificări.

$$\begin{array}{l} L_1 \\ L_2' \\ L_3' \end{array} \left\{ \begin{array}{rcrcrcrl} x & + & 2y & - & z & = & 4 & \\ & - & 2y & - & z & = & -6 & \left| \cdot (-\frac{8}{-2}) \right. \\ & + & 8y & - & z & = & 14 & \end{array} \right. \overset{L_3''=L_3'+4\cdot L_2'}{\Longrightarrow}$$

$$\begin{array}{l} L_1 \\ L_2' \\ L_3'' \end{array} \left\{ \begin{array}{rcrcrcr} x & + & 2y & - & z & = & 4 \\ & - & 2y & - & z & = & -6 \\ & & & - & 5z & = & -10 \end{array} \right.$$

Se observă că matricea coeficienţilor ultimului sistem de ecuaţii este superior triunghiulară.

Necunoscutele sistemului se determină in **etapa de retrosubstituţie** (substituţie regresivă).

În ultima ecuaţie apare o singură necunoscută, z, care se calculează astfel:

$$z = \frac{-10}{-5} = 2.$$

În a doua ecuaţie, avem tot o singură necunoscută, y, deoarece z este deja calculat la pasul precedent. Astfel:

$$y = \frac{-6+z}{-2} = \frac{-6+2}{-2} = 2.$$

În sfârşit, în prima ecuaţie, singura necunoscuta este acum x, iar y şi z fiind calculate la paşii anteriori. Rezultă:

$$x = \frac{4-2y+z}{1} = 4-4+2 = 2.$$

Soluţia sistemului de ecuaţii este (2, 2, 2).

2. Fie sistemul de ecuaţii:

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

Să se rezolve sistemul de ecuaţii prin metoda Gauss.

Rezolvare:

Etapa de eliminare:

• Prima ecuaţie (L_1) se înmulţeşte cu $-[3/(-1)]$ şi se adună la a doua ecuaţie (L_2), astfel se elimină termenul în x din a doua ecuaţie. Doar a doua ecuaţie se modifică ($L_2' = L_2 + 3 \cdot L_1$).

• Prima ecuaţie (L_1) se înmulţeşte cu $-[(-2)/(-1)]$ şi se adună la a treia ecuaţie (L_3), astfel se elimină termenul în x şi din a treia ecuaţie ($L_3' = L_3 + (-2) \cdot L_1$).

De precizat că prima ecuaţie (L_1) nu se modifică.

$$\begin{array}{ll} L_1 & \left\{ \begin{array}{rcrcrcrl} -x & + & 2y & + & 4z & = & 5 & \mid \cdot(-\frac{3}{-1}) \quad \mid \cdot(-\frac{-2}{-1}) \\ 3x & - & y & + & z & = & 3 & \\ -2x & - & y & + & z & = & -2 & \end{array} \right. \\ L_2 & \\ L_3 & \end{array} \quad \begin{array}{l} \overset{L_2'=L_2+3\cdot L_1}{\Longrightarrow} \\ \overset{L_3'=L_3+(-2)\cdot L_1}{\Longrightarrow} \end{array}$$

$$\begin{array}{ll} L_1 & \left\{ \begin{array}{rcrcrcr} -x & + & 2y & + & 4z & = & 5 \\ & & 5y & + & 13z & = & 18 \\ & - & 5y & - & 7z & = & -12 \end{array} \right. \\ L_2' & \\ L_3' & \end{array}$$

• A doua ecuaţie (L_2') se înmulţeşte cu $-[(-5)/5]$ şi se adună la a treia ecuaţie (L_3'), eliminându-se y din a treia ecuaţie ($L_3'' = L_3' + 1 \cdot L_2'$).

De notat că ecuaţia a doua (L_2') nu suferă modificări.

$$\begin{array}{ll} L_1 & \left\{ \begin{array}{rcrcrcrl} -x & + & 2y & + & 4z & = & 5 & \\ & & 5y & + & 13z & = & 18 & \mid \cdot(-\frac{-5}{5}) \\ & - & 5y & - & 7z & = & -12 & \end{array} \right. \\ L_2' & \\ L_3' & \end{array} \quad \overset{L_3''=L_3'+L_2'}{\Longrightarrow}$$

$$\begin{array}{ll} L_1 & \left\{ \begin{array}{rcrcrcr} -x & + & 2y & + & 4z & = & 5 \\ & & 5y & + & 13z & = & 18 \\ & & & & 6z & = & 6 \end{array} \right. \\ L_2' & \\ L_3'' & \end{array}$$

Se observă că matricea coeficienţilor ultimului sistem de ecuaţii este superior triunghiulară.

Etapa de retrosubstituţie:

• Necunoscutele sistemului se determină astfel:

$$\begin{aligned} z &= \tfrac{6}{6} = 1 \\ y &= \tfrac{18-13z}{5} = \tfrac{18-13}{-5} = 1 \\ x &= \tfrac{5-2y-4z}{-1} = \tfrac{5-2-4}{-1} = 1. \end{aligned}$$

Soluţia sistemului de ecuaţii este (1, 1, 1).

3. Fie sistemul de ecuaţii:

$$\begin{cases} -2x + y - z = -3 \\ -4x + 4y + z = 7 \\ 6x + 2y + 3z = 19 \end{cases}$$

Să se rezolve sistemul de ecuaţii prin metoda Gauss.

Rezolvare:

Etapa de eliminare:

$$\begin{array}{ll} L_1 \\ L_2 \\ L_3 \end{array} \left\{ \begin{array}{rcrcrcrl} -2x & + & y & - & z & = & -3 & \left|\cdot\left(-\frac{-4}{-2}\right)\right. \quad \left|\cdot\left(-\frac{6}{-2}\right)\right. \\ -4x & + & 4y & + & z & = & 7 & \\ 6x & + & 2y & + & 3z & = & 19 & \end{array} \right. \begin{array}{l} \\ \overset{L_2'=L_2+(-2)\cdot L_1}{\Longrightarrow} \\ \overset{L_3'=L_3+3\cdot L_1}{\Longrightarrow} \end{array}$$

$$\begin{array}{ll} L_1 \\ L_2' \\ L_3' \end{array} \left\{ \begin{array}{rcrcrcrl} -2x & + & y & - & z & = & -3 & \\ & & 2y & + & 3z & = & 13 & \left|\cdot\left(-\frac{5}{2}\right)\right. \\ & & 5y & & & = & 10 & \end{array} \right. \overset{L_3''=L_3'+(-5/2)\cdot L_2'}{\Longrightarrow}$$

$$\begin{array}{ll} L_1 \\ L_2' \\ L_3'' \end{array} \left\{ \begin{array}{rcrcrcr} -2x & + & y & - & z & = & -3 \\ & & 2y & + & 3z & = & 13 \\ & & & - & \frac{15}{2}z & = & -\frac{45}{2} \end{array} \right.$$

Matricea coeficienţilor ultimului sistem de ecuaţii este superior triunghiulară.

Se observă că în ultima ecuaţie a penultimului sistem de ecuaţii de mai sus există o singură necunoscută, y. Determinarea lui y din această ecuaţie este corectă din punct de vedere matematic, dar nu este conform metodei Gauss. La metoda Gauss, necunoscutele sistemului se determină când matricea sistemului are o formă superior triunghiulară, ceea ce nu este cazul matricii penultimului sistem de ecuaţii.

Etapa de retrosubstituţie:

$$\begin{aligned} z &= \frac{-\frac{45}{2}}{-\frac{15}{2}} = 3 \\ y &= \frac{13-3z}{2} = \frac{13-9}{2} = 2 \\ x &= \frac{-3-y+z}{-2} = \frac{-3-2+3}{-2} = 1. \end{aligned}$$

Soluţia sistemului de ecuaţii este (1, 2, 3).

4. Fie sistemul de ecuaţii:

$$\begin{cases} 2y - 3z = -1 \\ x + y - z = 1 \\ -2x + 3y + z = 2 \end{cases}$$

Să se rezolve sistemul de ecuaţii prin metoda Gauss.

Rezolvare:

Etapa de eliminare:

În prima ecuaţie coeficientul diagonal (pivotul, termenul în x) este nul, ceea ce ar duce la o împărţire la zero în etapa de eliminare. Din acest motiv, trebuie să se aplice o tehnică de pivotare. Se va utiliza pivotarea parţială (permutarea a două linii), fiind cea mai uşor de aplicat.

Pentru a obţine erori de rotunjire minime, se permută prima ecuaţie (cu pivot nul) cu ecuaţia care are termenul în x maxim în modul (coeficientul de sub pivotul nul). Se observă că a treia ecuaţie îndeplineşte această condiţie ($|-2| > |1|$).

Astfel, prima ecuaţie se permută cu a treia ecuaţie, iar sistemul de ecuaţii devine:

$$\begin{array}{ll} L_1' & \left\{ \begin{array}{rcrcrcll} -2x & + & 3y & + & z & = & 2 & |\cdot(-\frac{1}{-2}) \\ x & + & y & - & z & = & 1 & \\ & & 2y & - & 3z & = & -1 & \end{array} \right. \\ L_2 & \\ L_3' & \end{array} \quad \overset{L_2'=L_2+(1/2)\cdot L_1'}{\Longrightarrow}$$

$$\begin{array}{ll} L_1' & \left\{ \begin{array}{rcrcrcll} -2x & + & 3y & + & z & = & 2 & \\ & & \frac{5}{2}y & - & \frac{1}{2}z & = & 2 & |\cdot(-\frac{4}{5}) \\ & & 2y & - & 3z & = & -1 & \end{array} \right. \\ L_2' & \\ L_3' & \end{array} \quad \overset{L_3''=L_3'+(-4/5)\cdot L_2'}{\Longrightarrow}$$

$$\begin{array}{ll} L_1' & \left\{ \begin{array}{rcrcrcl} -2x & + & 3y & + & z & = & 2 \\ & & \frac{5}{2}y & - & \frac{1}{2}z & = & 2 \\ & & & - & \frac{13}{5}z & = & -\frac{13}{5} \end{array} \right. \\ L_2' & \\ L_3'' & \end{array}$$

Matricea coeficienţilor ultimului sistem de ecuaţii este superior triunghiulară.

Etapa de retrosubstituţie:

$$\begin{aligned} z &= \frac{-\frac{13}{5}}{-\frac{13}{5}} = 1 \\ y &= \frac{2+\frac{1}{2}z}{\frac{5}{2}} = \frac{2+\frac{1}{2}}{\frac{5}{2}} = 1 \\ x &= \frac{2-3y-z}{-2} = \frac{2-3-1}{-2} = 1. \end{aligned}$$

Soluţia sistemului de ecuaţii este (1, 1, 1).

5. Fie sistemul de ecuaţii:

$$\begin{cases} -x + y + z + w = 2 \\ 2x - 2y - z + w = 0 \\ -3x + 2y + z - w = -1 \\ x - 3y + z + 2w = 1 \end{cases}$$

Să se rezolve sistemul de ecuaţii prin metoda Gauss.

Rezolvare:

Etapa de eliminare:

$$\begin{array}{l} L_1 \\ L_2 \\ L_3 \\ L_4 \end{array} \left\{ \begin{array}{rcrcrcrcrl} -x & + & y & + & z & + & w & = & 2 & \mid \cdot (-\frac{2}{-1}) \\ 2x & - & 2y & - & z & + & w & = & 0 & \mid \cdot (-\frac{-3}{-1}) \\ -3x & + & 2y & + & z & - & w & = & -1 & \mid \cdot (-\frac{1}{-1}) \\ x & - & 3y & + & z & + & 2w & = & 1 & \end{array} \right. \begin{array}{l} \\ L_2'=L_2+2\cdot L_1 \\ \Longrightarrow \\ L_3'=L_3+(-3)\cdot L_1 \\ \Longrightarrow \\ L_4'=L_4+L_1 \\ \Longrightarrow \end{array}$$

$$\begin{array}{l} L_1 \\ L_2' \\ L_3' \\ L_4' \end{array} \left\{ \begin{array}{rcrcrcrcr} -x & + & y & + & z & + & w & = & 2 \\ & & & & z & + & 3w & = & 4 \\ & - & y & - & 2z & - & 4w & = & -7 \\ & - & 2y & + & 2z & + & 3w & = & 3 \end{array} \right.$$

În a doua ecuaţie coeficientul diagonal (pivotul, termenul în y) este nul, fiind necesară o pivotare parţială.

Pentru a obţine erori de rotunjire minime, se permută a doua ecuaţie (cu pivot nul) cu ecuaţia care are termenul în y maxim în modul (coeficientul de sub pivotul nul). Se observă că a patra ecuaţie îndeplineşte această condiţie ($|-2| > |-1|$).

Astfel, a doua ecuaţie se permută cu a patra ecuaţie, iar sistemul de ecuaţii devine:

$$\begin{array}{l} L_1 \\ L_2'' \\ L_3' \\ L_4'' \end{array} \left\{ \begin{array}{rcrcrcrcrl} -x & + & y & + & z & + & w & = & 2 & \\ & - & 2y & + & 2z & + & 3w & = & 3 & |\cdot(-\frac{-1}{-2}) \\ & - & y & - & 2z & - & 4w & = & -7 & \\ & & & & z & + & 3w & = & 4 & \end{array} \right. \overset{L_3''=L_3'+(-1/2)\cdot L_2''}{\Longrightarrow}$$

$$\begin{array}{l} L_1 \\ L_2'' \\ L_3'' \\ L_4'' \end{array} \left\{ \begin{array}{rcrcrcrcrl} -x & + & y & + & z & + & w & = & 2 & \\ & - & 2y & + & 2z & + & 3w & = & 3 & \\ & & & - & 3z & - & \frac{11}{2}w & = & -\frac{17}{2} & |\cdot(-\frac{1}{-3}) \\ & & & & z & + & 3w & = & 4 & \end{array} \right. \overset{L_4'''=L_4''+(1/3)\cdot L_3''}{\Longrightarrow}$$

$$\begin{array}{l} L_1 \\ L_2'' \\ L_3'' \\ L_4''' \end{array} \left\{ \begin{array}{rcrcrcrcr} -x & + & y & + & z & + & w & = & 2 \\ & - & 2y & + & 2z & + & 3w & = & 3 \\ & & & - & 3z & - & \frac{11}{2}w & = & -\frac{17}{2} \\ & & & & & & \frac{7}{6}w & = & \frac{7}{6} \end{array} \right.$$

Matricea coeficienţilor ultimului sistem de ecuaţii este superior triunghiulară.

Etapa de retrosubstituţie:

$$\begin{array}{rcl} w & = & \frac{\frac{7}{6}}{\frac{7}{6}} = 1 \\ z & = & \frac{-\frac{17}{2}+\frac{11}{2}w}{-3} = \frac{-\frac{17}{2}+\frac{11}{2}}{-3} = 1 \\ y & = & \frac{3-2z-3w}{-2} = \frac{3-2-3}{-2} = 1 \\ x & = & \frac{2-y-z-w}{-1} = \frac{2-1-1-1}{-1} = 1. \end{array}$$

Soluţia sistemului de ecuaţii este (1, 1, 1, 1).

3.5.2 Exemple propuse

1. Fie sistemul de ecuaţii:

$$\left\{ \begin{array}{rcrcrcr} 3x & - & y & + & z & = & 3 \\ 3x & - & 6y & + & z & = & -2 \\ -x & + & 2y & + & 4z & = & 5 \end{array} \right.$$

   Să se rezolve sistemul de ecuaţii prin metoda directă Gauss.

2. Fie sistemul de ecuaţii:

$$\left\{ \begin{array}{rcrcrcr} x & + & y & - & 3z & = & -1 \\ 2x & + & y & - & z & = & 2 \\ x & - & y & - & z & = & -1 \end{array} \right.$$

   Să se rezolve sistemul de ecuaţii prin metoda directă Gauss.

3. Fie sistemul de ecuaţii:

$$\begin{cases} x + 2y - z = 2 \\ 2x - y - 2z = -1 \\ -x + y + z = 1 \end{cases}$$

Să se rezolve sistemul de ecuaţii prin metoda Gauss.

4. Fie sistemul de ecuaţii:

$$\begin{cases} y + z = 3 \\ 2x + y - z = 1 \\ -x - y + 2z = 2 \end{cases}$$

Să se rezolve sistemul de ecuaţii prin metoda Gauss.

5. Fie sistemul de ecuaţii:

$$\begin{cases} -x + y + z = 1 \\ 2x - 2y - z = -1 \\ -3x + y + z = -1 \end{cases}$$

Să se rezolve sistemul de ecuaţii prin metoda Gauss.

3.6 Întrebări

1. Cum se pot verifica soluţiile obţinute în urma rezolvării unui sistem de ecuaţii? Cum s-ar putea evalua eroarea cu care s-au determinat necunoscutele?

2. Daţi exemplu de un sistem, cu matricea A nesingulară, care nu poate fi rezolvat prin algoritmul Gauss fără pivotare.

3. Care este ordinul de complexitate al algoritmului Gauss? Să se compare valoarea teoretică cu valorile obţinute în urma experimentărilor.

4. Să se evalueze durata medie a unei operaţii matematice elementare.

5. Cum variază erorile de calcul introduse de algoritmul Gauss cu dimensiunea sistemului? Care este explicaţia?

6. Cât de bine este evaluată eroarea adevărată prin norma reziduului?

7. Evidenţiaţi utilitatea acestei proceduri în electrotehnică.

8. Ce modificări trebuie aduse procedurii pentru a rezolva sisteme cu coeficienţi complecşi?

9. Ce alte îmbunătăţiri se pot aduce procedurii?

10. Ce limitări are algoritmul prezentat în capitolul 3?

11. Cum variază timpul de calcul la rezolvarea de sisteme liniare prin metodele Gauss fără pivotare şi cu pivotare? Care este explicaţia?

12. Care sunt avantajele şi dezavantajele metodelor cu pivotare?

13. Cum variază erorile de calcul introduse de algoritmul Gauss pentru metodele: fără pivotare, cu pivotare parţială, cu pivotare totală?

14. Să se facă o analiză a celor trei metode, luând în consideraţie erorile şi timpul de calcul. Care este metoda optimă?

15. Ce modificări trebuie aduse algoritmului pentru a rezolva sistemul cu coeficienţi complecşi?

16. Ce alte îmbunătăţiri se pot aduce algoritmului?

17. Documentaţi-vă în literatura de specialitate şi pe internet cum poate fi evaluată eroarea soluţiei numerice a unui sistem de ecuaţii algebrice liniare, atunci când termenul liber este perturbat. Ce se întelege printr-un sistem liniar bine condiţionat şi, respectiv, slab condiţionat? Cum se defineste "numărul de condiţionare" şi cum se foloseşte acesta pentru a stabili gradul de condiţionare al unei probleme?

18. Cum se rezolvă sistemele algebrice liniare prin metode directe în mediul MATLAB/SCILAB?

19. Identificaţi în biblioteca matematică descrisă in [2] funcţii avansate pentru rezolvarea cu metode directe a sistemelor de ecuaţii algebrice liniare. Ce aduc nou aceste funcţii faţă de cea folosită in lucrare?

Johann Carl Friedrich Gauss (1777, Brunswick - 1855,Hanover (acum Germania)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Gauss.html
http://scienceworld.wolfram.com/biography/Gauss.html

Pafnuty Lvovich Chebyshev (1821, Rusia - 1894, Rusia)

http://scienceworld.wolfram.com/biography/Chebyshev.html

Euclid of Alexandria (aprox. 325 BC - aprox. 265 BC, Alexandria, Egipt)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Euclid.html
http://scienceworld.wolfram.com/biography/Euclid.html

Lucrarea 4

Metode iterative de rezolvare a sistemelor algebrice liniare

4.1 Caracterizarea metodelor

Metodele iterative sunt metode care permit obţinerea soluţiei numerice a unui sistem de ecuaţii, prin generarea unui şir care tinde către soluţia exactă. Elementele acestui şir de iteraţii se calculează recursiv, iar procesul se opreşte după un număr finit de paşi, la îndeplinirea criteriului de eroare.

Chiar dacă soluţia obţinută prin metode iterative este afectată de erori de trunchiere, erori care nu apar în cazul metodelor directe, este totuşi posibil ca soluţia iterativă să fie mai precisă decât cea obţinută prin metode directe. Pentru o anumită clasă de sisteme, metodele iterative sunt superioare atât din punctul de vedere al erorii cât şi din cel al efortului de cacul.

4.2 Principiul metodei

În lucrare se prezintă cele mai simple metode iterative pentru rezolvarea sistemelor algebrice liniare şi se descrie clasa sistemelor pentru care acestea pot fi aplicate.

Metodele iterative de rezolvare a sistemelor de ecuaţii liniare sunt metodele în care termenul $\mathbf{x}^{(k)}$ al şirului soluţiilor se obţine din termenul anterior, $\mathbf{x}^{(k-1)}$.

Soluţia exactă se obţine teoretic după un număr infinit de iteraţii. În practică, prin efectuarea unui număr finit de iteraţii, se poate ajunge la o aproximare suficient de bună a soluţiei exacte. Dacă şirul iteraţiilor este convergent, cu cât se efectează mai multe iteraţii, cu atât soluţia numerică este mai precis determinată, erorile, atât cele de trunchiere cât şi cele de rotunjire, devenind tot mai mici.

În metodele iterative, se porneşte de la o iniţializare arbitrară pentru vectorul soluţie numerică $\mathbf{x}^0$. Pentru a determina noua valoare a soluţiei numerice, se rescrie ecuaţia sub forma:

$$\mathbf{x} = F(\mathbf{x}), \tag{4.1}$$

în care F se numeşte *aplicaţie cu punct fix*, iar la fiecare pas k al algoritmului, se determină noua soluţie numerică din relaţia:

$$\mathbf{x}^k = F(\mathbf{x}^{k-1}). \tag{4.2}$$

Pentru a aduce sistemul de rezolvat

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{4.3}$$

la forma unei aplicaţii cu punct fix, în general se caută o descompunere a matricei $\mathbf{A}$ într-o diferenţă de două matrice:

$$\mathbf{A} = \mathbf{B} - \mathbf{C}, \tag{4.4}$$

sistemul putând fi astfel adus la forma:

$$\mathbf{x} = \mathbf{B}^{-1}(\mathbf{b} + \mathbf{C}\mathbf{x}), \tag{4.5}$$

soluţia la pasul k fiind:

$$\mathbf{x}^k = \mathbf{B}^{-1}(\mathbf{C}\mathbf{x}^{k-1} + \mathbf{b}), \tag{4.6}$$

sau

$$\mathbf{x}^k = \mathbf{M}\mathbf{x}^{k-1} + \mathbf{u}, \tag{4.7}$$

în care $\mathbf{M} = \mathbf{B}^{-1}\mathbf{C}$ se numeşte *matrice de iteraţie*, iar $\mathbf{u} = \mathbf{B}^{-1}\mathbf{b}$.

Una din problemele metodelor iterative este *convergenţa şirului de iteraţii*. Se demonstrează că o condiţie suficientă pentru ca metoda să fie convergentă este ca valorile proprii ale matricei de iteraţie $\mathbf{M} = \mathbf{B}^{-1}\mathbf{C}$ să fie toate, în modul, mai mici decât 1. Definind *raza de convergenţă* a matricei $\mathbf{M}$, $\rho(\mathbf{M})$, ca fiind modulul celei mai mari valori proprii, condiţia de convergenţă se scrie:

$$\rho(\mathbf{M}) < 1 \tag{4.8}$$

Această condiţie de convergenţă este corelată cu norma matricei de iteraţie $\mathbf{M}$. Se demonstrează că, pentru orice matrice, există următoarea relaţie între norma şi raza sa de convergenţă:

$$\rho(\mathbf{M}) \leq \|\mathbf{M}\| \tag{4.9}$$

Prin urmare, dacă matricea $\mathbf{M}$ are norma subunitară ($\|\mathbf{M}\| < 1$), raza sa de convergenţă va fi şi ea mai mică decât 1, iar metoda iterativă va fi în acest caz convergentă.

Metodele iterative cele mai cunoscute sunt:

- metoda deplasărilor simultane (Jacobi),

- metoda deplasărilor succesive (Gauss-Seidel),
- metoda suprarelaxărilor succesive(Frankel-Young),
- metoda direcţiilor alternante (Peaceman-Rachford),
- metoda iteraţiilor bloc,
- metoda factorizării incomplete,
- metoda Southwell.

Metoda Jacobi a deplasărilor simultane constă în alegerea partitiţiei matricei **A** astfel: **B** este matricea alcătuită din elementele diagonale ale lui **A**

$$\mathbf{B} = \mathbf{D}, \tag{4.10}$$

iar matricea **C** conţine restul elementelor din matricea **A**, luate cu semn schimbat:

$$\mathbf{C} = -(\mathbf{L} + \mathbf{U}), \tag{4.11}$$

în care s-au notat cu **L** şi **U** triunghiul inferior, respectiv cel superior din **A**. Cu această descompunere, vectorul soluţie la fiecare pas k va avea expresia:

$$\mathbf{x}^k = \mathbf{D}^{-1}(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}^{k-1}), \tag{4.12}$$

iar matricea de iteraţie **M** va fi:

$$\mathbf{M} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}). \tag{4.13}$$

Se consideră linia i a sistemului de ecuaţii care trebuie rezolvat:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ii}x_i + \ldots + a_{in}x_n = b_i \tag{4.14}$$

Partiţionarea aleasă pentru matricea **A** revine la a determina pe x_i la pasul curent k din ecuaţia i, cu relaţia:

$$x_i^k = \frac{b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{k-1}}{a_{ii}}, \tag{4.15}$$

în care x_i din membrul stâng reprezintă componenta i a noii soluţii, iar x_j din membrul drept sunt valorile obţinute la precedentul pas al iteraţiei. Se observă că, pentru determinarea noii soluţii, trebuie cunoscute, pe tot parcursul iteraţiei k, valorile soluţiei de la pasul anterior $k-1$.

În metoda *Gauss-Seidel*, a deplasărilor succesive, partiţionarea se alege astfel:

$$\mathbf{B} = \mathbf{D} + \mathbf{L}, \tag{4.16}$$

matricea $\mathbf{B}$ conţinând astfel partea inferior triunghiulară a matricei $\mathbf{A}$ inclusiv diagonala, iar

$$\mathbf{C} = -\mathbf{U} \tag{4.17}$$

matricea $\mathbf{C}$ conţinând partea superior triunghiulară a matricei. Matricea de iteraţie va avea forma

$$\mathbf{M} = -(\mathbf{D} + \mathbf{L})^{-1}\mathbf{U}. \tag{4.18}$$

Această partiţionare presupune că în membrul stâng al ecuaţiei i din sistem rămân termenii care conţin $x_j, j \leq i$, iar în membrul drept trec toţi ceilalţi termeni:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ii}x_i = b_i - a_{i,i+1}x_{i+1} - \ldots - a_{in}x_n. \tag{4.19}$$

Ca şi până acum, componentele vectorului soluţie aflate în membrul stâng reprezintă valori "noi", calculate la pasul curent k, pe când componentele din membrul drept sunt cele calculate la pasul anterior. Din această relaţie rezultă valoarea componentei x_i la pasul curent:

$$x_i^k = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}}{a_{ii}}. \tag{4.20}$$

Se observă că o componentă x_i a soluţiei la pasul k este calculată în funcţie de componentele precedente $1, \ldots, i-1$, deja calculate la pasul curent şi de următoarele componente $i+1, \ldots, n$, calculate la pasul precedent. Algoritmul nu necesită păstrarea vechii componente i, după ce cea nouă a fost calculată, de aceea x_i nou se poate plasa în memorie în aceeaşi locaţie ca şi vechea valoare. Astfel, algoritmul Gauss-Seidel nu necesită spaţiu pentru memorarea decât a unui vector soluţie, spre deosebire de algoritmul Jacobi, unde trebuiau memoraţi atât x nou cât şi x vechi. În metoda Gauss-Seidel, imediat ce o componentă a fost determinată, ea este folosită în calculele următoare, înlocuind valoarea veche care se pierde, idee cunoscută sub numele de *principiul lui Seidel*.

Una din problemele care apar la rezolvarea sistemelor de ecuaţii liniare prin metode iterative este *alegerea criteriului de oprire a procesului iterativ*. O metodă de a rezolva această problemă a criteriului de oprire constă în evaluarea, după fiecare iteraţie, a erorii Cauchy

$$e = \|\mathbf{x}^{nou} - \mathbf{x}^{vechi}\| \tag{4.21}$$

şi întreruperea calculelor atunci când această valoare devine mai mică decât eroarea impusă, ε.

În ceea ce priveşte *convergenţa* metodelor, se demonstrează că la metodele Jacobi şi Gauss-Seidel, o condiţie suficientă ca metodele să fie convergente este ca matricea $\mathbf{A}$ a sistemului să fie diagonal dominantă, adică

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \forall i. \tag{4.22}$$

Desigur, aşa cum s-a arătat, condiţia de mai sus este echivalentă cu impunerea condiţiei ca norma matricei de iteraţie să fie subunitară, procesul iterativ fiind cu atât mai rapid convergent cu cât norma matricei de iteraţie este mai mică. În cazul matricelor simetrice şi pozitiv definite, metoda Gauss-Seidel este de aproximativ 2 ori mai rapidă decât metoda Jacobi. Acest avantaj, corelat şi cu necesitatea memorării unui singur vector soluţie, face ca metoda Gauss-Seidel să fie preferabilă metodei Jacobi din toate punctele de vedere.

4.3 Pseudocodul algoritmilor

Următoarea procedură permite rezolvarea sistemelor de ecuatii liniare prin metoda *Jacobi*.

```
procedura Jacobi (n,a,b,x,nrit,eps)
        tablou real a(n,n),b(n),x(n)
        întreg nrit
        real eps
        tablou real xn(N)                          ; x nou
        ; inițializări
        k = 0                                      ; contor iterații
        pentru i=1,n
                x_i = 0                            ; inițializarea soluției
        ; iterații
        repetă                                     ; parcurge iterațiile
                err= 0;                            ; eroarea la pasul curent
                pentru i= 1,n                      ; parcurge ecuațiile
                        s = b(i)
                        pentru j=1,n               ; parcurge ecuația i
                                s = s - a(i,j)x(j)
                        s = s + a(i,i)x(i)
                        xn(i) = s/a(i,i)           ; x nou
                        s = |xn(i) - x(i)|
                        dacă err < s atunci err = s
                pentru i= 1,n
                        x(i) = xn(i)               ; înlocuiește x vechi cu x nou
                k = k + 1                          ; incrementează contor iterații
        până când (err < eps) sau (k > nrit)
retur
```

Următoarea procedură permite rezolvarea sistemelor de ecuaţii liniare prin metoda *Gauss-Seidel*

```
procedura Gauss-Seidel(n,a,b,x,nrit,eps)
        tablou real a(n,n), b(n), x(n)
        întreg nrit
        real eps
        inițializări
        k = 0                                   ; contor iterații
        pentru i=1,n
                x(i) = 0                        ; inițializarea soluției
                ; iterații
                repetă                          ; parcurge iterațiile
                err = 0                         ; eroarea la pasul curent
                pentru i=1,n                    ; parcurge ecuațiile
                        s = b(i)
                        pentru j=1,n            ; parcurge ecuația i
                                s = s - a(i,j)x(j)
                        s = (s + a(i,i)x(i))/a(i,i)
                        err = err + (s - x(i))
                        x(i) = s                ; x nou
                k = k + 1                       ; incrementează contor iterații
                err = sqrt(err)
        până când (err < eps) sau (k > nrit)
retur
```

Procedurile Jacobi şi Gauss-Seidel au parametrii:

- de intrare
  - `n` = dimensiunea sistemului;
  - `a(n,n)`= matricea sistemului;
  - `b(n)` = vectorul termenilor liberi;
  - `nrit` = numărul maxim de iteraţii;
  - `eps` = eroarea admisă;
- de ieşire `x(n)` = vectorul soluţie.

Pentru a demonstra modul diferit în care se poate evalua eroarea, în algoritmul Jacobi s-a folosit norma Cebîşev a erorii, iar în algoritmul Gauss-Seidel norma Euclidiană.

4.4 Analiza algoritmilor

Efort de calcul

Pentru o iteraţie, ordinul de complexitate al metodelor Jacobi şi Gauss-Seidel este $O(n(n+2))$. Efortul de calcul pentru rezolvarea întregului sistem de ecuaţii liniare prin metode iterative este de ordinul $O(mn^2)$, în care numărul total de iteraţii m care vor fi efectuate nu este în general cunoscut dinainte. Efortul de calcul depinde de norma matricei de iteraţie, fiind cu atât mai mic cu cât norma este mai mică.

Necesar de memorie

Pentru memorarea matricei sistemului, a vectorilor termen liber şi soluţie sunt necesare n^2+2n locaţii de memorie. În plus, la algoritmul Jacobi mai sunt necesare n locaţii pentru memorarea soluţiei obţinute la pasul curent.

Dacă matricea sistemului este o matrice rară, atunci metodele iterative se dovedesc extrem de eficiente din punctul de vedere al memoriei, ele negenerând umpleri.

Analiza erorilor

Spre deosebire de metodele directe, la care singurele erori care apar sunt cele de rotunjire, la metodele iterative apar şi erori de trunchiere prin reţinerea din şirul convergent către soluţia exactă, a unui număr finit de termeni. Datorită convergenţei lor, metodele iterative au proprietatea remarcabilă de a corecta erorile de rotunjire apărute pe parcurs.

Eroarea absolută la iterţia k este de cel puţin $\|M\|$ ori mai mică decât eroarea de la pasul anterior:

$$e_k = \|\mathbf{x}^k - \mathbf{x}\| \leq \|\mathbf{M}\|\|\mathbf{x}^{k-1} - \mathbf{x}\| = \|\mathbf{M}\|e_{k-1} \leq \|\mathbf{M}\|^k e_0. \tag{4.23}$$

Se constată că eroarea finală depinde de eroarea iniţială, de numărul de iteraţii efectuate şi de norma matricei de iteraţie, care determină viteza de convergenţă. Aceeaşi relaţie este valabilă şi pentru reziduul $\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\|$.

4.5 Chestiuni de studiat

- Rezolvarea unor sisteme de ecuaţii liniare prin metodele Jacobi şi Gauss-Seidel;
- Analiza experimentală a erorilor şi a efortului de calcul, la metodele Jacobi şi Gauss-Seidel;
- Implementarea algoritmilor;
- Căutare de informaţii pe Internet.

4.6 Mod de lucru

Pentru desfăşurarea lucrării se selectează lucrarea *Metode iterative de rezolvare a sistemelor algebrice liniare* din meniul general de lucrări.

Aceasta are ca efect afişarea meniului:

1. Rezolvare de sisteme cu metodele Jacobi/Gauss-Seidel
2. Analiza algoritmilor

4.6.1 Rezolvarea unor sisteme de ecuaţii liniare prin metodele Jacobi/Gauss-Seidel

Programul lansat permite rezolvarea unor sisteme liniare prin metode iterative.

Se vor introduce:

- numărul de ecuaţii;
- elementele $a(i, j)$ ale matricei sistemului;
- termenii liberi $b(i)$;
- eroarea admisă;
- numărul maxim de iteraţii admis.

Programul afişează în consola Scilab informaţii despre procesul iterativ: dacă a fost convergent sau nu, norma matricei de iteraţie, în cazul în care procesul a fost convergent, câte iteraţii au fost necesare şi care este soluţia.

Se vor introduce parametrii necesari pentru rezolvarea următoarelor sisteme:

$$\begin{cases} x_1 + x_2 = 5 \\ 2x_1 + 3x_2 = 13 \end{cases} \quad : R \quad \begin{array}{l} x_1 = 2 \\ x_2 = 3 \end{array}$$

Se va inversa ordinea ecuaţiilor şi se va comenta efectul asupra soluţiei.

$$\begin{cases} 8x_1 + 2x_2 + x_3 = 15 \\ 10x_1 + 4x_2 + x_3 = 21 \\ 50x_1 + 25x_2 + 8x_3 = 124 \end{cases} \quad R: \quad \begin{array}{l} x_1 = 1 \\ x_2 = 2 \\ x_3 = 3 \end{array}$$

$$\begin{cases} 2x_1 + x_2 + x_3 = 7 \\ x_1 + 2x_2 + x_3 = 8 \\ x_1 + x_2 + x_3 = 6 \end{cases} \quad R: \begin{array}{l} x_1 = 1 \\ x_2 = 2 \\ x_3 = 3 \end{array}$$

$$\begin{cases} 3x_1 + x_3 = 10 \\ x_1 + 2x_2 + x_3 = 12 \\ x_1 + x_2 + 2x_3 = 13 \end{cases} \quad R: \begin{array}{l} x_1 = 2 \\ x_2 = 3 \\ x_3 = 4 \end{array}$$

Se va inversa ordinea a două ecuaţii din sistem şi se va comenta efectul asupra soluţiei.

Se vor comenta rezultatele obţinute, convergenţa metodelor pentru fiecare sistem rezolvat.

4.6.2 Analiza experimentală a algoritmilor

Se selectează opţiunea *Analiza algoritmilor*. Aceasta are ca efect afişarea unui meniu cu opţiunile:

- Eroarea în funcţie de numărul de iteraţii;
- Numărul de iteraţii în funcţie de norma matricei de iteraţie.

La selectarea opţiunii *Eroarea în funcţie de numărul de iteraţii*, utilizatorul introduce: norma matricei de iteraţie Jacobi şi numărul de ecuaţii (valoare iniţială, valoare finală, pas). Astfel se apelează un program care rezolvă, pe baza celor două metode iterative, sisteme de ecuaţii liniare generate aleator. Programul afişează rezultatele numerice în consola Scilab şi reprezintă grafic variaţia erorii Cauchy în funcţie de iteraţie. Valorile recomandate sunt: norma matricei de iteraţie Jacobi 0.5, dimensiunile sistemelor 20, 40, 60. Datele se pot nota într-un tabel de tipul:

iteraţii		1	5	9	13	17
n = 20	eroare J					
	eroare GS					
n = 40	eroare J					
	eroare GS					
n = 60	eroare J					
	eroare GS					

Se vor reprezenta pe acelaşi grafic erorile în funcţie de numărul de iteraţii. Se vor comenta rezultatele obţinute.

La selectarea opţiunii *Numărul de iteraţii în funcţie de norma matricei de iteraţie* utilizatorul alege dimensiunea n a sistemului şi eroarea de oprire. Programul afişează numărul de iteraţii necesar celor două metode pentru sisteme generate aleator, de dimensiune n, având norma matricei de iteraţie 0.1, 0.2, ..., 0.9. Datele se pot nota într-un tabel de tipul:

norma matricei Jacobi	0.1	0.2	...	0.9
nr. iteraţii Jacobi				
nr. iteraţii Gauss-Seidel				

Se vor reprezenta pe acelaşi grafic numărul de iteraţii în funcţie de norma matricei. Se vor comenta rezultatele obţinute.

4.6.3 Implementarea algoritmilor

Se va implementa pe calculator, în limbajul C, o procedură proprie de rezolvare a unui sistem de ecuaţii liniare, prin una din cele două metode iterative. Se va compila şi testa procedura, eliminându-se eventualele erori.

Se va scrie pseudocodul şi se va implementa pe calculator un program principal, care apelează procedura anterioară şi rezolvă sistemul de ecuaţii:

$$\begin{cases} 2x + z = 5 \\ x + y + z = 6 \\ y + 3z = 11 \end{cases} \qquad R: \begin{array}{l} x = 1 \\ y = 2 \\ z = 3 \end{array}$$

Acest program va permite introducerea datelor de la consolă şi afişarea pe ecran a soluţiei. Se vor nota şi comenta rezultatele obţinute şi eventualele dificultăţi apărute pe parcursul lucrului.

4.6.4 Căutare de informaţii pe Internet

Se vor căuta pe Internet informaţii (coduri) legate de rezolvarea iterativă a sistemelor algebrice de ecuaţii prin metode iterative. Cuvinte cheie recomandate: *Iterative methods for algebraic systems*, *Jacobi*, *Gauss-Seidel*.

4.7 Exemple

4.7.1 Exemple rezolvate

1. Fie sistemul de ecuaţii:

$$\begin{cases} x - 2y = -2 \\ -3x + 2y = -6 \end{cases}$$

   (a) să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodei Jacobi, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = 0$;

   (b) să se ilustreze grafic procesul iterativ şi să se comenteze convergenţa lui;

   (c) să se comenteze rezultatele obţinute atunci când se schimbă ordinea ecuaţiilor.

   Rezolvare:

   (a) Iteraţiile metodei Jacobi se calculează conform formulelor:

$$\begin{cases} x^{(k+1)} = -2 + 2y^{(k)} \\ 2y^{(k+1)} = -6 + 3x^{(k)} \end{cases}$$

   Soluţia la prima iteraţie este:

$$\begin{cases} x^{(1)} = -2 + 2y^{(0)} = -2 \\ y^{(1)} = \frac{-6+3x^{(0)}}{2} = -3, \end{cases}$$

   iar la a doua iteraţie:

$$\begin{cases} x^{(2)} = -2 + 2y^{(1)} = -2 - 6 = -8 \\ y^{(2)} = \frac{-6+3x^{(1)}}{2} = \frac{-6-6}{2} = -6. \end{cases}$$

   (b) Din punct de vedere geometric, rezolvarea sistemului este echivalentă cu găsirea punctului de intersecţie al dreptelor ecuaţiilor, D_1 şi D_2:

$$\begin{aligned} D_1 &: \quad x - 2y + 2 = 0 \\ D_2 &: -3x + 2y + 6 = 0 \end{aligned}$$

   Vom reprezenta grafic aceste drepte. Dreapta D_1 taie axele în punctele de coordonate $(0, 1)$ şi $(-2, 0)$. Dreapta D_2 taie axele în punctele de coordonate $(0, -3)$ şi $(2, 0)$. Dreptele D_1 şi D_2 sunt concurente în punctul $(4, 3)$, soluţia sistemului de ecuaţii.

   Procesul iterativ este ilustrat în figura 1 şi se observă că este divergent. Deşi problema este bine formulată matematic (soluţia există şi este unică), metoda Jacobi eşuează.

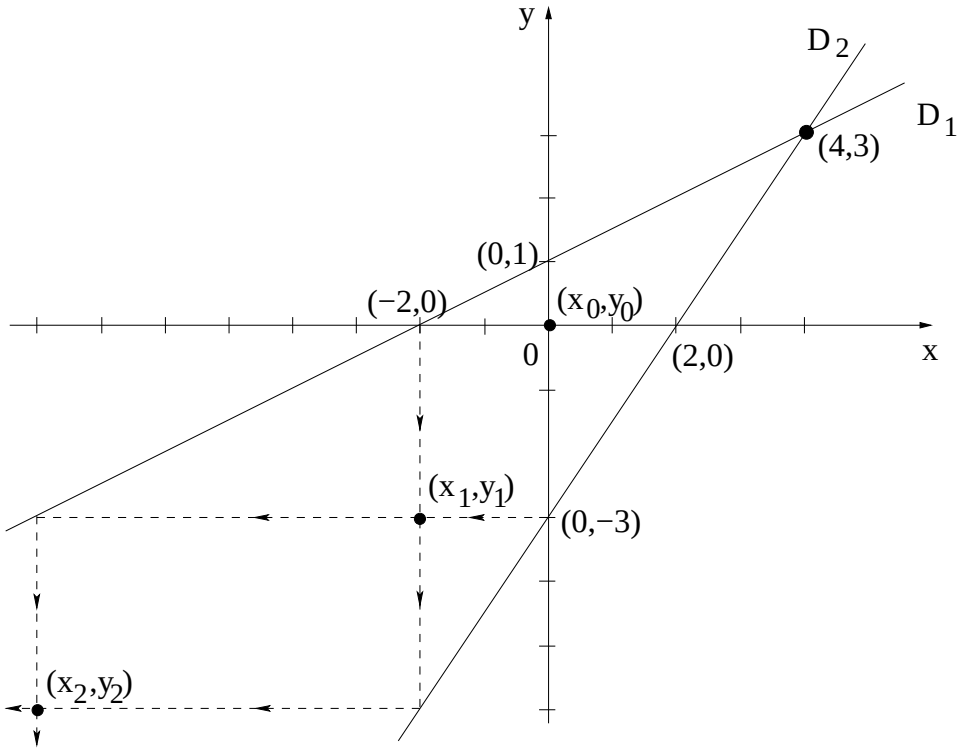

Fig. 1. Proces iterativ divergent al metodei Jacobi

Convergenţa metodei depinde de proprietăţile matricei de iteraţie. Dacă notăm $A = D + L + U$, atunci $M = -D^{-1}(L + U)$ este matricea de iteraţie în cazul metodei Jacobi.

În cazul problemei considerate:

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}, \quad L = \begin{pmatrix} 0 & 0 \\ -3 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0 & -2 \\ 0 & 0 \end{pmatrix}.$$

$$M = -\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 0 & -2 \\ -3 & 0 \end{pmatrix} = -\begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} 0 & -2 \\ -3 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ \frac{3}{2} & 0 \end{pmatrix}.$$

Raza spectrală (de convergenţă) a matricei de iteraţie M este: $\rho(M) = \max\limits_{i=1,2} |\lambda_i|$, unde λ_i sunt valorile proprii, care reprezintă soluţiile ecuaţiei $\det(M - \lambda I) = 0$.

În cazul studiat:

$$M - \lambda I = \begin{pmatrix} -\lambda & 2 \\ \frac{3}{2} & -\lambda \end{pmatrix} \Longrightarrow \det(M - \lambda I) = \lambda^2 - 3 = 0.$$

Valorile proprii sunt $\lambda_{1,2} = \pm\sqrt{3}$, iar $\rho(M) = \max\limits_{i=1,2} |\lambda_i| = \sqrt{3}$.

Deoarece $\rho(M) > 1$, procesul iterativ este divergent.

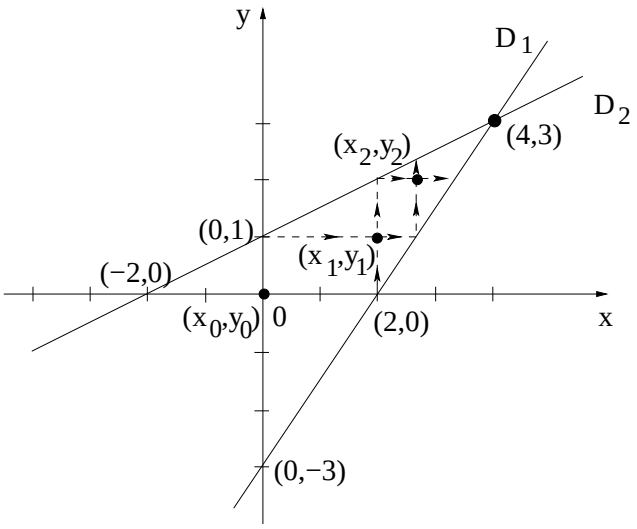

Fig. 2. Proces iterativ convergent al metodei Jacobi

(c) În cazul în care se schimbă ordinea ecuaţiilor, sistemul de rezolvat este:

$$\left\{ \begin{array}{rcrcl} -3x & + & 2y & = & -6 \\ x & - & 2y & = & -2 \end{array} \right.$$

Soluţia la prima iteraţie se calculează astfel:

$$\left\{ \begin{array}{rcl} -3x^{(1)} & = & -6 - 2y^{(0)} \\ -2y^{(1)} & = & -2 - x^{(0)} \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} x^{(1)} = 2 \\ y^{(1)} = 1 \end{array} \right.$$

Iar soluţia la a doua iteraţie este:

$$\left\{ \begin{array}{rcl} -3x^{(2)} & = & -6 - 2y^{(1)} \\ -2y^{(2)} & = & -2 - x^{(1)} \end{array} \right. \Longrightarrow \left\{ \begin{array}{l} x^{(2)} = \frac{8}{3} \\ y^{(2)} = 2 \end{array} \right.$$

Procedeul iterativ este convergent aşa cum se observă în figura 2.
Într-adevăr, pentru acest sistem, matricea de iteraţie este:

$$M = -\begin{pmatrix} 3 & 0 \\ 0 & -2 \end{pmatrix}^{-1} \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} = -\begin{pmatrix} -\frac{1}{3} & 0 \\ 0 & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} 0 & 2 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \frac{2}{3} \\ \frac{1}{2} & 0 \end{pmatrix},$$

iar valorile proprii sunt soluţiile ecuaţiei:

$$\det(M - \lambda I) = 0 \Longrightarrow \begin{vmatrix} -\lambda & \frac{2}{3} \\ \frac{1}{2} & -\lambda \end{vmatrix} = 0.$$

Deci:

$$\lambda^2 - \frac{1}{3} = 0 \Longrightarrow \lambda_{1,2} = \pm\frac{1}{\sqrt{3}} \Longrightarrow \rho(M) = \max_{i=1,2} |\lambda_i| = \frac{1}{\sqrt{3}}.$$

Deoarece $\rho(M) < 1$, procesul iterativ este convergent.

Aceeaşi concluzie se poate obţine dacă se observă că matricea sistemului este diagonal dominantă (vezi relaţia (4.22)). Aceasta este o condiţie suficientă de convergenţă.

Inegalitatea (4.22) este adevărată pentru cele două ecuaţii ale sistemului:

$$|-3| > |2|, \quad |-2| > |1|,$$

deci matricea sistemului este diagonal dominantă, ceea ce este echivalent cu $||M|| < 1$. Pentru orice matrice avem $\rho(M) \leq ||M||$, astfel că $\rho(M) < 1$.

2. Fie sistemul de ecuaţii de la exerciţiul anterior:

$$\begin{cases} x - 2y = -2 \\ -3x + 2y = -6 \end{cases}$$

(a) să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodei Gauss-Seidel, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = 0$;

(b) să se ilustreze grafic procesul iterativ şi să se comenteze convergenţa lui;

(c) să se comenteze rezultatele obţinute atunci când se schimbă ordinea ecuaţiilor.

Rezolvare:

(a) Iteraţiile metodei Gauss-Seidel se calculează conform formulelor:

$$\begin{cases} x^{(k+1)} = -2 + 2y^{(k)} \\ 2y^{(k+1)} = -6 + 3x^{(k+1)} \end{cases}$$

Soluţia la prima iteraţie este:

$$\begin{cases} x^{(1)} = -2 + 2y^{(0)} = -2 \\ y^{(1)} = \frac{-6+3x^{(1)}}{2} = \frac{-6-6}{2} = -6, \end{cases}$$

iar la a doua iteraţie:

$$\begin{cases} x^{(2)} = -2 + 2y^{(1)} = -2 - 12 = -14 \\ y^{(2)} = \frac{-6+3x^{(2)}}{2} = \frac{-6-42}{2} = -24. \end{cases}$$

(b) Procesul iterativ este ilustrat în figura 3 şi se observă că este divergent. Deşi problema este bine formulată matematic (soluţia există şi este unică), metoda Gauss-Seidel eşuează.

Matricea de iteraţie în cazul metodei Gauss-Seidel este $M = -(D + L)^{-1}U$.

În cazul problemei considerate:

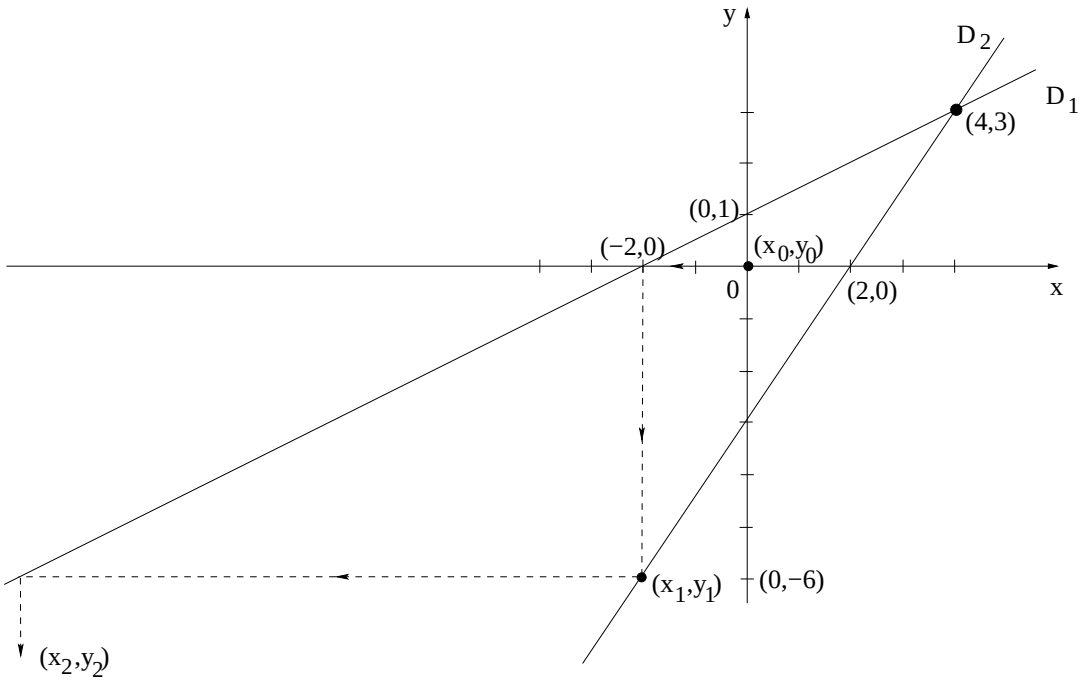

Fig. 3. Proces iterativ divergent al metodei Gauss-Seidel

$$M = -\begin{pmatrix} 1 & 0 \\ -3 & 2 \end{pmatrix}^{-1}\begin{pmatrix} 0 & -2 \\ 0 & 0 \end{pmatrix} = -\begin{pmatrix} 1 & 0 \\ \frac{3}{2} & \frac{1}{2} \end{pmatrix}\begin{pmatrix} 0 & -2 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 0 & 3 \end{pmatrix}.$$

Raza spectrală (de convergenţă) a matricei de iteraţie M este: $\rho(M) = \max\limits_{i=1,2}|\lambda_i|$, unde λ_i sunt valorile proprii, care sunt soluţiile ecuaţiei $\det(M - \lambda I) = 0$.
În cazul studiat:

$$M - \lambda I = \begin{pmatrix} -\lambda & 2 \\ 0 & 3-\lambda \end{pmatrix} \Longrightarrow \det(M - \lambda I) = -\lambda(3-\lambda) = 0.$$

Valorile proprii sunt $\lambda_1 = 0$, $\lambda_2 = 3$, iar $\rho(M) = \max\limits_{i=1,2}|\lambda_i| = 3$.

Deoarece $\rho(M) > 1$, procesul iterativ este divergent.

(c) În cazul în care se schimbă ordinea ecuaţiilor, sistemul de rezolvat este:

$$\begin{cases} -3x & + & 2y & = & -6 \\ x & - & 2y & = & -2 \end{cases}$$

Soluţia la prima iteraţie se calculează astfel:

$$\begin{cases} -3x^{(1)} & = & -6 - 2y^{(0)} \\ -2y^{(1)} & = & -2 - x^{(1)} \end{cases} \Longrightarrow \begin{cases} x^{(1)} = 2 \\ y^{(1)} = 2 \end{cases}$$

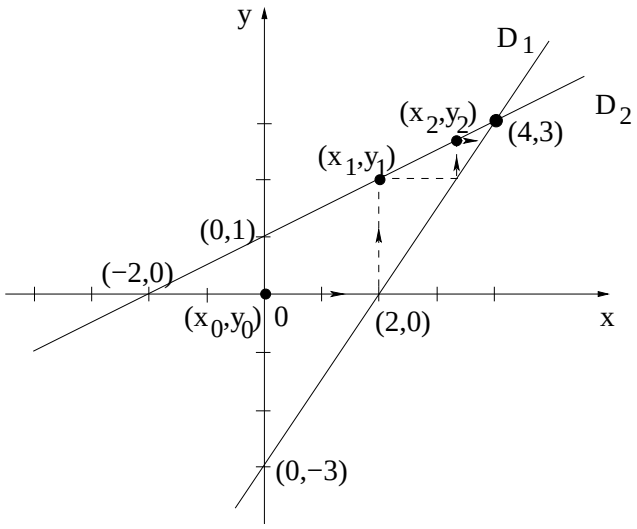

Fig. 4. Proces iterativ convergent al metodei Gauss-Seidel

Iar soluţia la a doua iteraţie este:

$$\left\{\begin{array}{rcl} -3x^{(2)} & = & -6-2y^{(1)} \\ -2y^{(2)} & = & -2-x^{(2)} \end{array}\right. \Longrightarrow \left\{\begin{array}{l} x^{(2)}=\frac{10}{3} \\ y^{(2)}=\frac{8}{3} \end{array}\right.$$

Procedeul iterativ este convergent aşa cum se observă în figura 4.
Într-adevăr, pentru acest sistem, matricea de iteraţie este:

$$M=-\begin{pmatrix} -3 & 0 \\ 1 & -2 \end{pmatrix}^{-1}\begin{pmatrix} 0 & 2 \\ 0 & 0 \end{pmatrix}=-\begin{pmatrix} -\frac{1}{3} & 0 \\ -\frac{1}{6} & -\frac{1}{2} \end{pmatrix}\begin{pmatrix} 0 & 2 \\ 0 & 0 \end{pmatrix}=\begin{pmatrix} 0 & \frac{2}{3} \\ 0 & \frac{1}{3} \end{pmatrix},$$

iar valorile proprii sunt soluţiile ecuaţiei:

$$\det(M-\lambda I)=0 \Longrightarrow \begin{vmatrix} -\lambda & \frac{2}{3} \\ 0 & \frac{1}{3}-\lambda \end{vmatrix}=0.$$

Deci:

$$-\lambda(\frac{1}{3}-\lambda)=0 \Longrightarrow \lambda_1=0,\ \lambda_2=\frac{1}{3} \Longrightarrow \rho(M)=\max_{i=1,2}|\lambda_i|=\frac{1}{3}.$$

Deoarece $\rho(M)<1$, procesul iterativ este convergent.

3. Fie sistemul de ecuaţii:

$$\left\{\begin{array}{rcrcrcr} 3x & - & y & + & z & = & 3 \\ 3x & - & 6y & + & z & = & -2 \\ -x & + & 2y & + & 4z & = & 5 \end{array}\right.$$

Să se comenteze convergenţa metodelor iterative şi să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodelor Jacobi şi Gauss-Seidel, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = z^{(0)} = 0$.

Rezolvare:

Deoarece matricea coeficienţilor sistemului este diagonal dominantă, metodele iterative Jacobi şi Gauss-Seidel sunt convergente.

De exemplu, pentru prima ecuaţie a unui sistem, conform relaţiei (4.22), trebuie să fie adevărată inegalitatea: $|a_{11}| > |a_{12}| + |a_{13}|$.

Pentru sistemul de ecuaţii considerat, inegalitatea (4.22) este adevărată pentru toate cele trei ecuaţii:

$$|3| > |-1| + |1|, \quad |-6| > |3| + |1|, \quad |4| > |-1| + |2|,$$

deci matricea sistemului este diagonal dominantă, ceea ce este echivalent cu $||M|| < 1$. Pentru orice matrice avem $\rho(M) \leq ||M||$, astfel că $\rho(M) < 1$, ceea ce înseamnă că metodele iterative sunt convergente pentru acest sistem.

Metoda Jacobi

Soluţia la o anumită iteraţie se determină în funcţie de soluţia calculată la iteraţia precedentă.

Conform relaţiei (4.15), din prima ecuaţie a sistemului se determină:

$$x^{(1)} = \frac{3 + y^{(0)} - z^{(0)}}{3} = \frac{3 + 0 - 0}{3} = 1,$$

iar din ecuaţiile a doua şi a treia rezultă următoarele componente ale soluţiei la prima iteraţie:

$$y^{(1)} = \frac{-2 - 3x^{(0)} - z^{(0)}}{-6} = \frac{-2 - 0 - 0}{-6} = \frac{1}{3},$$

$$z^{(1)} = \frac{5 + x^{(0)} - 2y^{(0)}}{4} = \frac{5 + 0 - 0}{4} = \frac{5}{4}.$$

La a doua iteraţie a metodei Jacobi soluţia sistemului este:

$$x^{(2)} = \frac{3 + y^{(1)} - z^{(1)}}{3} = \frac{3 + \frac{1}{3} - \frac{5}{4}}{3} = \frac{25}{36},$$

$$y^{(2)} = \frac{-2 - 3x^{(1)} - z^{(1)}}{-6} = \frac{-2 - 3 - \frac{5}{4}}{-6} = \frac{25}{24},$$

$$z^{(2)} = \frac{5 + x^{(1)} - 2y^{(1)}}{4} = \frac{5 + 1 - \frac{2}{3}}{4} = \frac{4}{3}.$$

Metoda Gauss-Seidel

Soluţia la o anumită iteraţie se determină în funcţie de componentele soluţiei deja calculate la iteraţia curentă şi restul componentelor soluţiei calculate la iteraţia precedentă.

Conform relaţiei (4.20), din prima ecuaţie a sistemului se determină:

$$x^{(1)} = \frac{3 + y^{(0)} - z^{(0)}}{3} = \frac{3 + 0 - 0}{3} = 1,$$

iar din ecuaţiile a doua şi a treia rezultă următoarele componente ale soluţiei la prima iteraţie:

$$y^{(1)} = \frac{-2 - 3x^{(1)} - z^{(0)}}{-6} = \frac{-2 - 3 - 0}{-6} = \frac{5}{6},$$

$$z^{(1)} = \frac{5 + x^{(1)} - 2y^{(1)}}{4} = \frac{5 + 1 - \frac{5}{3}}{4} = \frac{13}{12}.$$

La a doua iteraţie a metodei Gauss-Seidel soluţia sistemului este:

$$x^{(2)} = \frac{3 + y^{(1)} - z^{(1)}}{3} = \frac{3 + \frac{5}{6} - \frac{13}{12}}{3} = \frac{11}{12},$$

$$y^{(2)} = \frac{-2 - 3x^{(2)} - z^{(1)}}{-6} = \frac{-2 - \frac{33}{12} - \frac{13}{12}}{-6} = \frac{35}{36},$$

$$z^{(2)} = \frac{5 + x^{(2)} - 2y^{(2)}}{4} = \frac{5 + \frac{11}{12} - \frac{35}{18}}{4} = \frac{143}{144}.$$

4. Fie sistemul de ecuaţii:

$$\begin{cases} x - 2y + 2z = 2 \\ 2x - y + 2z = 6 \\ x + 2y + z = 8 \end{cases}$$

Să se comenteze convergenţa metodelor iterative şi să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodelor Jacobi şi Gauss-Seidel, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = z^{(0)} = 1$.

Rezolvare:

Inegalitatea (4.22) nu este adevărată pentru cele trei ecuaţii ale sistemului considerat:

$$|1| < |-2| + |2|, \quad |-1| < |2| + |2|, \quad |1| > |1| + |2|.$$

Totuşi, deşi matricea nu este diagonal dominantă, nu putem concluziona că metodele iterative nu sunt convergente.

Condiţia necesară şi suficientă de convergenţă este ca raza de convergenţă sa fie subunitară (4.8).

Pentru sistemul considerat, deoarece matricea sistemului nu este diagonal dominantă, norma matricei de iteraţie este supraunitară. Se cunoaşte că raza de convergenţă este mai mică decât norma matricei de iteraţie (4.9), însă nu putem afirma că raza de convergenţă este supraunitară.

Raza de convergenţă trebuie calculată pentru a preciza convergenţa şirului de soluţii.

În cazul metodei Jacobi, matricea de iteraţie este:

$$M = -\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & -2 & 2 \\ 2 & 0 & 2 \\ 1 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 & -2 \\ 2 & 0 & 2 \\ -1 & -2 & 0 \end{pmatrix},$$

iar valorile proprii sunt soluţiile ecuaţiei:

$$\det(M - \lambda I) = 0 \Longrightarrow \begin{vmatrix} -\lambda & 2 & -2 \\ 2 & -\lambda & 2 \\ -1 & -2 & -\lambda \end{vmatrix} = 0.$$

Deci:

$$-\lambda^3 + 2\lambda + 4 = 0 \Longrightarrow \rho(M) = \max_{i=1,2,3} |\lambda_i| = 2 > 1.$$

În cazul metodei Gauss-Seidel, matricea de iteraţie este:

$$M = -\begin{pmatrix} 1 & 0 & 0 \\ 2 & -1 & 0 \\ 1 & 2 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 & -2 & 2 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 & -2 \\ 0 & 4 & -2 \\ 0 & -10 & 6 \end{pmatrix},$$

iar valorile proprii sunt soluţiile ecuaţiei:

$$\det(M - \lambda I) = 0 \Longrightarrow \begin{vmatrix} -\lambda & 2 & -2 \\ 0 & 4 - \lambda & -2 \\ 0 & -10 & 6 - \lambda \end{vmatrix} = 0.$$

Deci:

$$\lambda(-\lambda^2 + 10\lambda - 4) = 0 \Longrightarrow \rho(M) = \max_{i=1,2,3} |\lambda_i| = 9.58 > 1.$$

Pentru ambele metode raza de convergenţă este supraunitară, deci metodele iterative nu sunt convergente pentru acest sistem.

Metoda Jacobi

La prima iteraţie a metodei Jacobi soluţia sistemului este:

$$x^{(1)} = \frac{2 + 2y^{(0)} - 2z^{(0)}}{1} = 2 + 2 - 2 = 2,$$

$$y^{(1)} = \frac{6 - 2x^{(0)} - 2z^{(0)}}{-1} = -(6 - 2 - 2) = -2,$$
$$z^{(1)} = \frac{8 - x^{(0)} - 2y^{(0)}}{1} = 8 - 1 - 2 = 5.$$

La a doua iteraţie a metodei Jacobi soluţia sistemului este:

$$x^{(2)} = \frac{2 + 2y^{(1)} - 2z^{(1)}}{1} = 2 - 4 - 10 = -12,$$
$$y^{(2)} = \frac{6 - 2x^{(1)} - 2z^{(1)}}{-1} = -(6 - 4 - 10) = 8,$$
$$z^{(2)} = \frac{8 - x^{(1)} - 2y^{(1)}}{1} = 8 - 2 + 4 = 10.$$

Metoda Gauss-Seidel

La prima iteraţie a metodei Gauss-Seidel soluţia sistemului este:

$$x^{(1)} = \frac{2 + 2y^{(0)} - 2z^{(0)}}{1} = 2 + 2 - 2 = 2,$$
$$y^{(1)} = \frac{6 - 2x^{(1)} - 2z^{(0)}}{-1} = -(6 - 4 - 2) = 0,$$
$$z^{(1)} = \frac{8 - x^{(1)} - 2y^{(1)}}{1} = 8 - 2 + 0 = 6.$$

La a doua iteraţie a metodei Gauss-Seidel soluţia sistemului este:

$$x^{(2)} = \frac{2 + 2y^{(1)} - 2z^{(1)}}{1} = 2 + 0 - 12 = -10,$$
$$y^{(2)} = \frac{6 - 2x^{(2)} - 2z^{(1)}}{-1} = -(6 + 20 - 12) = -14,$$
$$z^{(2)} = \frac{8 - x^{(2)} - 2y^{(2)}}{1} = 8 + 10 + 28 = 46.$$

4.7.2 Exemple propuse

1. Fie sistemul de ecuaţii:
$$\begin{cases} 2x + y = -4 \\ 3x + 5y = 15 \end{cases}$$

   (a) să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodelor Jacobi şi Gauss-Seidel, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = 0$;

   (b) să se ilustreze grafic procesul iterativ şi să se comenteze convergenţa lui;

(c) să se comenteze rezultatele obţinute atunci când se schimbă ordinea ecuaţiilor.

2. Fie sistemul de ecuaţii:

$$\begin{cases} 3x + 5y = 15 \\ 2x + y = -4 \end{cases}$$

(a) să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodelor Jacobi şi Gauss-Seidel, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = 0$;

(b) să se ilustreze grafic procesul iterativ şi să se comenteze convergenţa lui;

(c) să se comenteze rezultatele obţinute atunci când se schimbă ordinea ecuaţiilor.

3. Fie sistemul de ecuaţii:

$$\begin{cases} x + y - 3z = -1 \\ 2x + y - z = 2 \\ x - y - z = -1 \end{cases}$$

Să se comenteze convergenţa metodelor iterative şi să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodelor Jacobi şi Gauss-Seidel, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = z^{(0)} = 0$.

4. Fie sistemul de ecuaţii:

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

Să se comenteze convergenţa metodelor iterative şi să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodelor Jacobi şi Gauss-Seidel, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = z^{(0)} = 0$.

5. Fie sistemul de ecuaţii:

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

Să se comenteze convergenţa metodelor iterative şi să se determine soluţia sistemului de ecuaţii pentru primele două iteraţii ale metodelor Jacobi şi Gauss-Seidel, cunoscând soluţia iniţială $x^{(0)} = y^{(0)} = z^{(0)} = 0$.

4.8 Întrebări şi probleme

1. Daţi o explicaţie calitativă pentru convergenţa mai rapidă a metodei Gauss-Seidel faţa de metoda Jacobi.

2. Există sisteme de ecuaţii, nedominant diagonale, care se pot rezolva totuşi prin metoda Gauss-Seidel, deşi metoda Jacobi nu este convergentă?

3. Care este explicaţia că inversând ordinea a 2 ecuaţii în sistemele propuse pentru rezolvare cu primul program demonstrativ, metodele iterative nu conduc la obţinerea soluţiei?

4. Cum scade eroarea în funcţie de numărul de iteraţii, la metodele Jacobi şi Gauss-Seidel? Cum este corelată scăderea erorii cu norma matricei de iteraţie?

5. Scrieţi pseudocodul unei proceduri care să genereze, cu ajutorul unui generator de numere aleatoare, o matrice a cărei normă este dată.

6. Care sunt matricele $\mathbf{B}$ şi $\mathbf{C}$ în care s-a partiţionat matricea sistemului la o metodă iterativă care ar folosi următoarea expresie pentru determinare a componentei i a soluţiei :
$$x_i^k = \frac{b_i - \sum_{j=i+1}^{n} a_{ij} x_j^k - \sum_{j=1}^{i-1} a_{ij} x_j^{k-1}}{a_{ii}}.$$

7. Scrieţi pseudocodul unui polialgoritm de rezolvare a unui sistem de ecuaţii liniare prin metode directe a cărui soluţie este rafinată ulterior prin metode iterative, în vederea eliminării erorilor de rotunjire.

8. Analizaţi teoretic modul în care depinde numărul de iteraţii m, necesare atingerii unei precizii dorite, în funcţie de norma matricei $\mathbf{M}$.

9. Modificaţi pseudocodul algoritmului Gauss-Seidel prin adoptarea unui criteriu de eroare relativ la norma reziduului $\|\mathbf{Ax} - \mathbf{b}\|$.

10. Scrieţi un algoritm de rezolvare iterativă a unui sistem cu matrice tridiagonală.

11. Scrieţi un algoritm de rezolvare iterativă a sistemului $\mathbf{Ax} = \mathbf{b}$ bazat pe partiţia:
$$\mathbf{A} = \mathbf{B} - \mathbf{C} = \mathbf{L} + \mathbf{D} - (-\mathbf{U}),$$
cu $\mathbf{D}$ matrice bloc diagonală.

12. Documentaţi-vă în literatura de specialitate şi pe internet ce inseamnă preconditionarea? Cum influenţeaza această operaţie numărul de condiţionare şi viteza de convergenţă a rezolvării sistemelor liniare prin metode iterative?

13. Rezolvaţi în mediul MATLAB/SCILAB, un sistem de ecuaţii liniare algebrice folosind o metodă iterativă. Evaluaţi viteza de convergenţă a procesului iterativ.

14. Identificaţi în biblioteca matematică descrisa în [2] funcţii avansate pentru rezolvarea cu metode directe a sistemelor de ecuaţii algebrice liniare cu matrice rare. Ce aduc nou aceste funcţii faţă de cea folosită în lucrare?

Carl Gustav Jacob Jacobi (1804, Prusia - 1851, Germania)
http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Jacobi.html
http://scienceworld.wolfram.com/biography/Jacobi.html

Philipp Ludwig von Seidel (1821, Germania - 1896, Germania)
http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Seidel.html

Augustin Louis Cauchy (1789,Franta - 1857, Franta)
http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Cauchy.html
http://scienceworld.wolfram.com/biography/Cauchy.html

Lucrarea 5

Analiza numerică a circuitelor electrice liniare în regim permanent

5.1 Caracterizarea lucrării

Cele mai simple circuite electrice sunt circuitele rezistive liniare. Studiul acestor circuite este o problemă foarte importantă, deoarece se constată că şi analiza altor categorii de circuite, cum sunt cele neliniare sau cele în regim tranzitoriu, se reduce în final la această problemă. Circuitele electrice rezistive liniare sunt caracterizate de sisteme de ecuaţii algebrice liniare formate din ecuaţiile lui Kirchhoff şi din relaţiile constitutive ale elementelor. Din punct de vedere matematic, analiza unui astfel de circuit se reduce la rezolvarea unui sistem liniar.

Analiza asistată de calculator a circuitelor electrice presupune nu numai rezolvarea numerică a ecuaţiilor asociate acestor circuite ci şi generarea automată, cu ajutorul calculatorului a acestor ecuaţii. Ecuaţiile unui circuit electric pot avea diferite forme echivalente. Cea mai eficientă tehnică de scriere automată a ecuaţiilor s-a dovedit a fi tehnica nodală, bazată pe metoda potenţialelor la noduri.

În lucrare se prezintă tehnica nodală aplicată în analiza circuitelor electrice liniare, de curent continuu si alternativ. Pentru descrierea circuitelor a fost ales un limbaj de maximă simplitate, orientat pe laturi.

5.2 Principiul metodei

Se consideră un circuit electric cu N noduri şi L laturi. Pe fiecare latură k a acestui circuit se află un rezistor liniar cu rezistenţa R_k, eventual înseriat cu o sursă cu t.e.m. E_k,

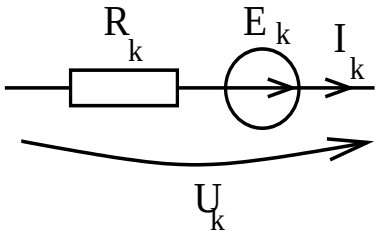

Figura 5.1: Latura standard considerată pentru analiza circuitelor rezistive liniare de curent continuu.

astfel încât:

$$u_k = R_k i_k - E_k, \quad k = 1, 2, \ldots, L, \tag{5.1}$$

unde u_k este tensiunea la bornele laturii k, iar i_k este curentul ce străbate latura (figura 5.1).

Relaţia (5.1) se poate scrie şi sub forma:

$$i_k = \frac{u_k + E_k}{R_k} = G_k u_k + J_k, \tag{5.2}$$

în care $G_k = 1/R_k$ este conductanţa laturii, presupusă mărginită, iar $J_k = E_k/R_k$ este curentul electromotor al generatorului de curent Norton echivalent laturii. S-a presupus că nici o latură nu are rezistenţa nulă.

Dacă se notează cu $\mathbf{i} = [i_1, i_2, \ldots, i_L]^T$ vectorul intensităţilor curenţilor din laturile circuitului, cu $\mathbf{u} = [u_1, u_2, \ldots, u_L]^T$ vectorul tensiunilor la bornele laturilor şi cu $\mathbf{v} = [v_1, v_2, \ldots, v_{N-1}]^T$ vectorul potenţialelor nodurilor, atunci teoremele lui Kirchhoff capătă următoarea formă matriceală:

$$\mathbf{Ai} = \mathbf{0}, \tag{5.3}$$

$$\mathbf{u} = \mathbf{A}^T \mathbf{v}, \tag{5.4}$$

în care s-a notat cu $\mathbf{A} \in \mathbb{R}^{(N-1)\times L}$ matricea redusă a incidenţelor laturi-noduri.

Problema fundamentală a analizei circutelor electrice rezistive liniare constă în determinarea vectorilor $\mathbf{u}$, $\mathbf{i}$ şi $\mathbf{v}$, atunci când se cunosc parametrii R_k, E_k, pentru $k = 1, \ldots, L$ şi topologia circuitului descrisă, de exemplu, prin matricea $\mathbf{A}$. Soluţia acestei probleme se determină prin rezolvarea sistemului de ecuaţii algebrice liniare (5.1), (5.3), (5.4).

Scriind şi ecuaţiile constitutive ale laturilor (5.1) sub forma matriceală:

$$\mathbf{u} = \mathbf{Ri} - \mathbf{E}, \tag{5.5}$$

unde $\mathbf{R} = diag(R_1, R_2, \ldots, R_L) \in \mathbb{R}^{L\times L}$ este matricea rezistenţelor laturilor, iar $\mathbf{E} = [E_1, E_2, \ldots, E_L]^T$ este matricea t.e.m., rezultă, prin eliminarea vectorilor $\mathbf{u}$ şi $\mathbf{i}$ din (5.3), (5.4), şi (5.5):

$$\mathbf{Ai} = \mathbf{AR}^{-1}(\mathbf{u} + \mathbf{E}) = \mathbf{AR}^{-1}\mathbf{A}^T + \mathbf{AR}^{-1}\mathbf{E} = \mathbf{0}. \tag{5.6}$$

Dacă se notează cu $\mathbf{G} = \mathbf{A}\mathbf{R}^{-1}\mathbf{A}^T$ matricea conductanţelor nodale şi cu $\mathbf{i_S} = -\mathbf{A}\mathbf{R}^{-1}\mathbf{E}$ matricea injecţiilor de curent în noduri (suma curenţilor de scurtcircuit) rezultă sistemul de ecuaţii algebrice liniare

$$\mathbf{G}\mathbf{v} = \mathbf{i_S} \tag{5.7}$$

specific tehnicii nodale, matricea $\mathbf{G}$ fiind o matrice pătrată de dimensiune $N-1$. Prin rezolvarea acestui sistem, rezultă potenţialele nodurilor $\mathbf{v} = \mathbf{G}^{-1}\mathbf{i_S}$, din care se calculează cu (5.4) tensiunile şi cu (5.5) curenţii.

Se constată că fiecare latură k, conectată între nodurile iniţial ni şi final nf, contribuie la patru termeni ai matricei $\mathbf{G}$ şi la doi termeni ai vectorului $\mathbf{i_S}$:

	Matricea $\mathbf{G}$		vectorul $\mathbf{i}_S$
	coloana ni	coloana nf	
linia ni	$1/R_k$	$-1/R_k$	$-E_k/R_k$
linia nf	$-1/R_k$	$1/R_k$	E_k/R_k

Această observaţie permite generarea automată a matricelor $\mathbf{G}$, $\mathbf{i_S}$, prin parcurgerea laturilor şi adunarea contribuţiei fiecărei laturi, la aceste matrice. Se constată că matricea $\mathbf{G}$ este diagonal dominantă, ceea ce permite rezolvarea sistemului liniar atât cu metode iterative cât şi cu metode directe, fără pivotare.

Analiza circuitelor electrice liniare în curent alternativ prin tehnica nodală presupune **reprezentarea circuitului în complex**. Dacă se notează cu $\mathbf{I} \in \mathbb{C}^L$, $\mathbf{U} \in \mathbb{C}^L$, $\mathbf{V} \in \mathbb{C}^{N-1}$, vectorii curenţilor, tensiunilor şi respectiv potenţialelor complexe, ecuaţiile circuitului au forma matriceală:

$$\begin{aligned} \mathbf{A}\mathbf{I} &= \mathbf{0}, \\ \mathbf{U} &= \mathbf{A}^T\mathbf{V}, \\ \mathbf{U} &= \mathbf{Z}\mathbf{I} - \mathbf{E}, \end{aligned} \tag{5.8}$$

în care $\mathbf{Z} = \operatorname{diag}(\underline{Z}_1, \underline{Z}_2, \ldots, \underline{Z}_L) \in \mathbb{C}^{L\times L}$ este matricea impedanţelor complexe ale laturilor, iar $\mathbf{E} = \operatorname{diag}(\underline{E}_1, \underline{E}_2, \ldots, \underline{E}_L)^T$ este matricea tensiunilor electromotoare complexe. Impedanţele complexe ale elementelor ideale sunt: la rezistor $\underline{Z}_R = R$, la bobină $\underline{Z}_L = j\omega L$, iar la condensator $\underline{Z}_C = \frac{1}{j\omega C}$, unde ω este pulsaţia (frecvenţa unghiulară) în radiani/s.

Prin eliminări succesive din (5.8) se obţine:

$$\mathbf{Y}\mathbf{V} = \mathbf{I_S}, \tag{5.9}$$

în care $\mathbf{Y} = \mathbf{Z}^{-1}\mathbf{A}^T$ este matricea admitanţelor nodale, iar $\mathbf{I_S} = -\mathbf{A}\mathbf{Z}^{-1}\mathbf{E}$ este matricea injecţiilor de curent din noduri. Prin rezolvarea sistemului algebric liniar (5.9) rezultă potenţialele $\mathbf{V} = \mathbf{Y}^{-1}\mathbf{I_S}$.

În consecinţă, analiza de curent alternativ se desfăşoară similar cu cea de curent continuu, cu deosebirea că în acest caz se operează cu numere complexe.

5.3 Pseudocodul metodei

Următorul pseudocod descrie analiza unui **circuit rezistiv liniar, de curent continuu**.

; Introducerea datelor de descriere a circuitului
citește N ; numărul de noduri
citește L ; numărul de laturi
pentru $k = 1, L$; parcurge laturi
    citește ni_k, nf_k ; nodurile inițial și final
    citește R_k, E_k ; rezistența și t.e.m

; Rezolvă circuitul prin tehnica nodală și determină vectorul v al potențialelor nodal (N, L, ni, nf, R, E, v)

; Calculează și afișează soluția
$pg = 0$; puterea generată
$pc = 0$; puterea consumată
pentru $k = 1, L$; parcurge laturile
    $u = v_{ni(k)} - v_{nf(k)}$; tensiunea la bornele laturii
    $i = (u + E_k)/R_k$; curentul din latură
    $pg = pg + E_k i$
    $pc = pc + R_k i^2$
    scrie k, u, i
scrie pc, pg ; bilanțul de puteri

Acest program apelează procedura nodal, care determină potenţialele nodurilor:

procedura nodal (N, L, ni, nf, R, E, v)

; analizează circuitul rezistiv liniar cu tehnica nodală

întreg N ; număr de noduri
întreg L ; număr de laturi
tablou întreg $ni(L)$; noduri inițiale
tablou întreg $nf(L)$; noduri finale
tablou real $R(L)$; rezistețele laturilor
tablou real $E(L)$; t.e.m. ale laturilor
tablou real $v(N)$; potențialele nodurilor
; (date de ieșire)
tablou real $G(N, N)$; matricea conductanțelor nodale

tablou real $is(L)$; vectorul injecţiilor de curent
pentru $i = 1, N$; iniţializează matricea sistemului
$is(i) = 0$
pentru $j = 1, N$
$G(i, j) = 0$
pentru $k = 1, L$; parcurge laturi
$n1 = ni(k)$
$n2 = nf(k)$
$G(n1, n1) = G(n1, n1) + 1/R_k$
$G(n2, n2) = G(n2, n2) + 1/R_k$
$G(n1, n2) = G(n1, n2) - 1/R_k$
$G(n2, n1) = G(n2, n1) - 1/R_k$
$is(n1) = is(n1) - E_k/R_k$
$is(n2) = is(n2) + E_k/R_k$
Gauss (N-1, G, is, v) ; rezolvă sistem liniar cu N-1 ecuaţii
; cu matricea G şi termenul liber is
$v(N) = 0$
retur

În procedura nodal se apelează procedura Gauss, care determină vectorul **v** al potenţialelor, prin rezolvarea sistemului liniar $\mathbf{Gv} = \mathbf{i_S}$ cu metoda eliminării gaussiene. Pentru a simplifica algoritmii, matricea **G** generată are dimensiunea $N \times N$, dar la rezolvarea sistemului aceasta se consideră de dimensiuni $(N-1)\times(N-1)$, ceea ce corespunde alegerii ultimului nod, ca nod de referinţă. Algoritmul prezentat poate fi extins fără dificultăţi pentru a permite analiza circuitelor care conţin şi surse de curent.

Dacă se operează cu variabile complexe şi nu cu variabile reale, procedura nodal se poate aplica la analiza circuitelor de curent alternativ. Se va nota cu `nodal-cx` varianta acestei proceduri, în care declaraţia **real** se înlocuieşte cu **complex**.

Următorul pseudocod descrie **algoritmul de analiză a circuitelor de curent alternativ**.

; Introducerea datelor de descriere
citeşte N ; număr de noduri
citeşte L ; număr de laturi
citeşte f ; frecvenţa în Hz
pentru $k = 1, L$
citeşte ni_k, nf_k
citeşte tip_k ; tipul elementului R, L sau C
citeşte p_k ; parametrul elementului pasiv
citeşte e_k ; valoarea efectivă

citește fi_k ; faza inițială
$w = 2 \cdot \pi \cdot f$; pulsația
pentru $k = 1, L$
dacă $tip_k = L$ **atunci**
$Z_k = complex(0, wp_k)$
altfel dacă $tip_k = C$ **atunci**
$Z_k = complex(0, -1/(wp_k))$
altfel
$Z_k = complex(p_k, 0)$
$E_k = complex(e_k cos(fi_k), e_k sin(fi_k))$

; Determină potențialele complexe ale nodurilor
nodal-cx (N, L, ni, nf, Z, E, V)

; Calculează și afișează soluția
$Sg = complex(0, 0)$; puterea complexă generată
$Sc = complex(0, 0)$; puterea complexă consumată
pentru $k = 1, L$
$U = V_{ni(k)} - V_{nf(k)}$
$I = (U + E_k)/Z_k$
$Sg = Sg + E_k \cdot conjugat(I)$
$Sc = Sc + Z_k \cdot |I|^2$
scrie k ; latura
scrie $|I|$; valoarea efectivă
scrie $arg(I)$; faza curentului

scrie $Re(Sg)$; puterea generată activă P_g
scrie $Im(Sg)$; puterea generată reactivă Q_g
scrie $Re(Sc)$; puterea consumată activă P_c
scrie $Im(Sc)$; puterea consumată reactivă Q_g

Dacă partea de analiză a acestui program este repetată ciclic pentru diferite frecvenţe se obţin caracteristicile de frecvenţă ale circuitului.

5.4 Analiza algoritmilor

Necesarul de memorie

Memoria necesară în analiza circuitelor electrice rezistive liniare este ocupată de:

- 2 vectori întregi **ni**, **nf** de dimensiune L;

- 2 vectori reali $\mathbf{R}$, $\mathbf{E}$ de dimensiune L;
- 1 vector real $\mathbf{i_S}$ de dimensiune N;
- 1 matrice $\mathbf{G}$ de dimensiune $N \times N$.

Se constată că necesarul de memorie depinde pătratic de dimensiunea circuitului, caracterizată în principal prin numărul de noduri N.

Efort de calcul

Timpul de calcul necesar analizei este folosit în special pentru rezolvarea sistemului liniar de dimensiune N, deci ordinul de complexitate al algoritmului de analiză este cubic $\mathrm{O}(2N^3/3)$.

Analiza erorilor

Erorile numerice apărute în analiza circuitelor sunt:

- erori inerente, în datele de intrare;
- erori de rotunjire, datorate reprezentării finite.

Erorile inerente şi de rotunjire se propagă în procesul de calcul şi pot genera instabilităţi numerice, cu atât mai mari, cu cât sistemul de ecuaţii este mai slab condiţionat.

Dacă circuitul analizat are rezistenţe foarte diferite, atunci instabilităţile numerice pot deveni importante. Aşa se întâmplă, dacă de exemplu, una din rezistenţele laturilor tinde către zero (conductanţa tinde către infinit, ceea ce determină valori foarte mari pentru unele din elementele matricei $\mathbf{G}$).

5.5 Chestiuni de studiat

1. Analiza numerică a unui circuit rezistiv liniar în regim staţionar (curent continuu);
2. Analiza numerică a unui circuit liniar în regim sinusoidal (curent alternativ);
3. Implementarea unui algoritm de analiză numerică a unui circuit electric liniar;
4. Căutarea de informaţii pe Internet.

5.6 Modul de lucru

Pentru desfăşurarea lucrării se selectează lucrarea *Analiza numerică a circuitelor liniare în regim permanent* din meniul principal de lucrări.

Aceasta are ca efect lansarea unui meniu cu următoarele opţiuni:

- Circuite de curent continuu;
- Circuite de curent alternativ,

din care utilizatorul selectează opţiunea dorită.

5.6.1 Analiza numerică a unui circuit rezistiv liniar

Prin selectarea opţiunii *Circuite de curent continuu* din meniul principal se lansează un program de analiză a circuitelor electrice de tip R, E.

Programul solicită din partea utilizatorului următoarele informaţii de descriere a circuitului:

- numărul de noduri;
- numărul de laturi;
- pentru fiecare latură de circuit:
  - nodul iniţial al laturii;
  - nodul final al laturii;
  - rezistenţa din latură;
  - t.e.m. a sursei de tensiune.

După introducerea datelor, programul analizează circuitul introdus cu metoda potenţialelor la noduri şi sunt afişate pentru fiecare latură valoarea tensiunii şi cea a intensităţii curentului. Sunt calculate şi afişate valorile puterii consumate şi puterii generate.

Pentru a analiza un circuit electric cu ajutorul acestui program, acesta trebuie "pregătit" în felul următor:

- se numerotează nodurile de la 1 la N;
- se numerotează laturile de la 1 la L;

- se alege pentru fiecare latură un sens de parcurs, orientat de la borna minus la borna plus a sursei de tensiune, dacă aceasta există, şi arbitrar în caz contrar.

Se recomandă analiza unui circuit electric simplu, de exemplu cu $L = 3$ şi $N = 2$, a cărei soluţie este cunoscută. Se va studia soluţia numerică obţinută, pentru valori extreme (foarte mari şi foarte mici) ale uneia din rezistenţele circuitului.

5.6.2 Analiza unui circuit de curent alternativ

Prin selectarea opţiunii *Circuite de curent alternativ* se lansează un program, care permite analiza unui circuit electric în regim sinusoidal.

Pentru descrierea circuitului, în afară de numărul de laturi, numărul de noduri şi a frecvenţei de lucru (exprimată în Hz), pentru fiecare latură se descrie:

- nodul iniţial şi final;
- tipul elementului pasiv din latură (R, L sau C);
- valoarea parametrului elementului pasiv (R, L sau C);
- valoarea efectivă a t.e.m. a sursei de tensiune din latură;
- faza iniţială a t.e.m. a sursei de tensiune (în grade).

După introducerea datelor se analizează circuitul cu metoda nodală şi se afişază, pentru fiecare latură:

- valoarea efectivă a curentului;
- faza iniţială a curentului.

În final se afişează puterile active şi reactive, consumate şi generate. Se recomandă să se analizeze numeric funcţionarea unui circuit electric stabilizator de curent de tip Boucherot pentru diferite valori ale rezistenţei de sarcină. Reamintim că un circuit de tip Boucherot este un circuit de tipul celui din figura 5.2.

Pentru acest circuit curentul $\underline{I}_3$ prin impedanţa $\underline{Z}_3$ nu depinde de valoarea acestei impedanţe dacă $\underline{Z}_1 + \underline{Z}_2 = 0$. Pentru a justifica acest lucru este suficient să se calculeze curentul $\underline{I}_3$:

$$\underline{I}_3 = \frac{\underline{E}}{\underline{Z}_1 + \frac{\underline{Z}_2\underline{Z}_3}{\underline{Z}_2+\underline{Z}_3}} \frac{\underline{Z}_2}{\underline{Z}_2 + \underline{Z}_3} = \frac{\underline{E}\underline{Z}_2}{\underline{Z}_1\underline{Z}_2 + (\underline{Z}_1 + \underline{Z}_2)\underline{Z}_3}.$$

Este evident că, dacă $\underline{Z}_1 + \underline{Z}_2 = 0$ curentul $\underline{I}_3$ nu depinde de $\underline{Z}_3$, şi anume el este

$$\underline{I}_3 = \frac{\underline{E}}{\underline{Z}_1}.$$

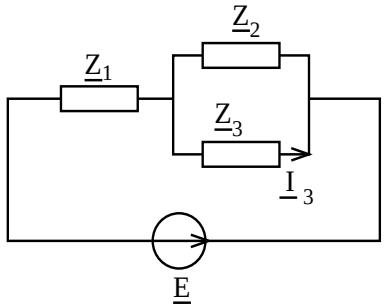

Figura 5.2: Circuit de tip Boucherot. ($\underline{Z}_1 + \underline{Z}_2 = 0$ şi $\underline{I}_3$ nu depinde de $\underline{Z}_3$)

5.6.3 Implementarea algoritmilor

Se va implementa în limbajul C, algoritmul de analiză a circuitelor electrice rezistive liniare. Programul va fi editat, compilat, executat şi testat pentru un circuit simplu.

5.6.4 Căutarea de informaţii pe Internet

Căutaţi pe Internet informaţii (coduri) legate de rezolvarea circuitelor electrice. Exemple de cuvinte cheie: *electric circuits simulation*.

5.7 Exemple

5.7.1 Exemple rezolvate

1. Fie circuitul de curent continuu din figura 5. Se cunosc: $R_1 = 1\Omega$, $R_2 = 2\Omega$, $R_3 = 3\Omega$, $R_4 = 4\Omega$, $R_5 = 5\Omega$, $R_6 = 6\Omega$, $E_1 = 10\text{V}$, $E_3 = 30\text{V}$, $E_5 = 50\text{V}$.

   Se cer:

   - Să se precizeze numărul de noduri, numărul de laturi şi, pentru fiecare latură, nodul iniţial, nodul final, rezistenţa şi t.e.m. a sursei de tensiune.
   - Ce dimensiuni au matricea $\mathbf{G}$ şi vectorul $\mathbf{i_S}$ asamblate de algoritm?
   - Să se determine contribuţiile laturilor 3 şi 4 la matricea $\mathbf{G}$ şi vectorul $\mathbf{i_S}$.
   - Care sunt matricea $\mathbf{G}$ şi vectorul $\mathbf{i_S}$ asamblate?
   - Care este sistemul de ecuaţii de rezolvat?

   Rezolvare:

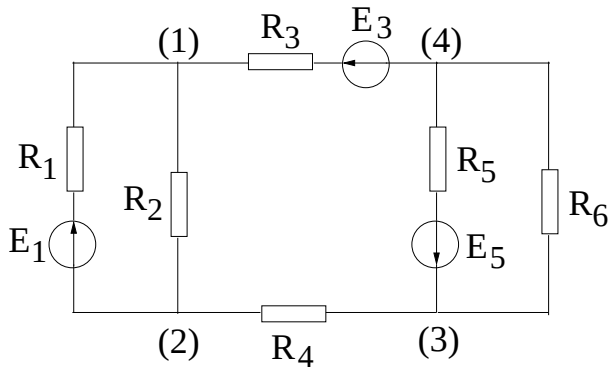

Fig. 5. Circuit rezistiv liniar de curent continuu

Pentru circuitul din figura 5, numărul de noduri este $N = 4$, iar numărul de laturi este $L = 6$.

Pentru laturile cu surse de tensiune, sensul laturii este sensul intern al sursei de tensiune, iar pentru restul laturilor, sensul laturii este ales de utilizator.

Latura	Nodul iniţial	Nodul final	$R\,[\Omega]$	$E\,[\mathrm{V}]$
1	2	1	1	10
2	1	2	2	0
3	4	1	3	30
4	2	3	4	0
5	4	3	5	50
6	4	3	6	0

Deoarece numărul total de noduri este 4, matricea $\mathbf{G}$ asamblată de algoritm are dimensiunea $(4, 4)$, iar vectorul $\mathbf{i_S}$ are dimensiunea 4.

Contribuţiile laturii 3 la matricea $\mathbf{G}$ şi vectorul $\mathbf{i_S}$:

$\mathbf{G}$				$\mathbf{i_S}$
$\frac{1}{3}$			$-\frac{1}{3}$	$\frac{30}{3}$
$-\frac{1}{3}$			$\frac{1}{3}$	$-\frac{30}{3}$

Contribuţiile laturii 4 la matricea $\mathbf{G}$ şi vectorul $\mathbf{i_S}$:

G				$\mathbf{i_S}$
	$\frac{1}{4}$	$-\frac{1}{4}$		0
	$-\frac{1}{4}$	$\frac{1}{4}$		0

Matricea **G** şi vectorul $\mathbf{i_S}$ asamblate:

G				$\mathbf{i_S}$
$1+\frac{1}{2}+\frac{1}{3}$	$-1-\frac{1}{2}$		$-\frac{1}{3}$	$10+\frac{30}{3}$
$-1-\frac{1}{2}$	$1+\frac{1}{2}+\frac{1}{4}$	$-\frac{1}{4}$		-10
	$-\frac{1}{4}$	$\frac{1}{4}+\frac{1}{5}+\frac{1}{6}$	$-\frac{1}{5}-\frac{1}{6}$	$\frac{50}{5}$
$-\frac{1}{3}$		$-\frac{1}{5}-\frac{1}{6}$	$\frac{1}{3}+\frac{1}{5}+\frac{1}{6}$	$-\frac{30}{3}-\frac{50}{5}$

Nodul 4 este nod de referinţă.

Matricea **G** a sistemului de rezolvat se obţine din matricea **G** asamblată prin eliminarea ultimei linii şi a ultimei coloane. Vectorul $\mathbf{i_S}$ al sistemului de rezolvat se obţine din vectorul $\mathbf{i_S}$ asamblat prin eliminarea ultimei linii.

Matricea **G** şi vectorul $\mathbf{i_S}$ ale sistemului de ecuaţii de rezolvat:

G			$\mathbf{i_S}$
$1+\frac{1}{2}+\frac{1}{3}$	$-1-\frac{1}{2}$		$10+\frac{30}{3}$
$-1-\frac{1}{2}$	$1+\frac{1}{2}+\frac{1}{4}$	$-\frac{1}{4}$	-10
	$-\frac{1}{4}$	$\frac{1}{4}+\frac{1}{5}+\frac{1}{6}$	$\frac{50}{5}$

2. Fie circuitul de curent alternativ din figura 6: Se cunosc: frecvenţa $f = 50\text{Hz}$, $R_2 = 3\Omega$, $R_3 = 1\Omega$, $R_5 = 4\Omega$, $L_1 = \frac{2}{100\pi}\text{H}$, $L_4 = \frac{3}{100\pi}\text{H}$, $C_3 = \frac{1}{100\pi}\text{F}$, $C_6 = \frac{1}{400\pi}\text{F}$, $e_1(t) = 30\sqrt{2}\sin(\omega t)\text{V}$, $e_5(t) = 20\sin(\omega t + \frac{\pi}{2})\text{V}$.

   Să se precizeze numărul de noduri, numărul de laturi şi, pentru fiecare latură, nodul iniţial, nodul final, impedanţa şi valoarea complexă a t.e.m. a sursei de tensiune.

   Rezolvare:

   Pentru circuitul din figura 6, numărul de noduri este $N = 4$, iar numărul de laturi este $L = 6$.

   Pentru laturile cu surse de tensiune, sensul laturii este sensul intern al sursei de tensiune, iar pentru restul laturilor, sensul laturii este ales de utilizator.

   Transformarea în complex simplificat a unei valori instantanee a t.e.m. este:

$$e(t) = E\sqrt{2}\sin(\omega t + \varphi) \longrightarrow \underline{E} = E\mathrm{e}^{j\varphi} = E[\cos(\varphi) + j\sin(\varphi)].$$

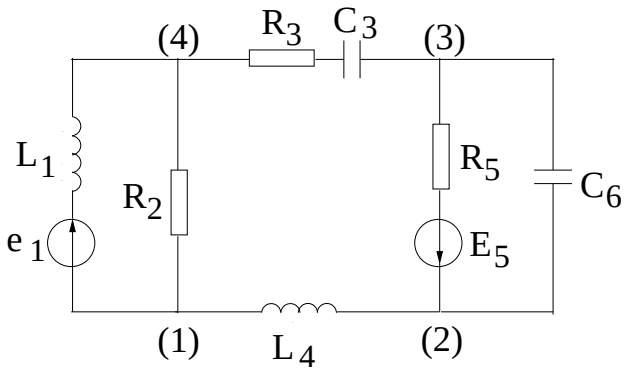

Fig. 6. Circuit rezistiv liniar de curent alternativ

Astfel:

$$\underline{E}_1 = 30\mathrm{e}^{j0} = 30, \quad \underline{E}_5 = \frac{20}{\sqrt{2}}\mathrm{e}^{j\frac{\pi}{2}} = \frac{20}{\sqrt{2}}j.$$

Impedanţele rezistorului, bobinei şi condesatorului sunt:

$$\underline{Z}_R = R, \quad \underline{Z}_L = j\omega L, \quad \underline{Z}_C = \frac{1}{j\omega C} = -j\frac{1}{\omega C},$$

unde $\omega = 2\pi f = 100\pi$ este pulsaţia.

De exemplu, impedanţele laturilor 1 şi 3 sunt:

$$\underline{Z}_1 = j\omega L_1 = 2j, \quad \underline{Z}_3 = R_3 - j\frac{1}{\omega C_3} = 1 - j.$$

Latura	Nodul iniţial	Nodul final	$\underline{Z}$	$\underline{E}$
1	1	4	$2j$	30
2	1	4	3	0
3	4	3	$1-j$	0
4	2	1	$3j$	0
5	3	2	4	$\frac{20}{\sqrt{2}}j$
6	2	3	$-4j$	0

5.7.2 Exemple propuse

1. Fie circuitul de curent continuu din figura 7: Se cunosc: $R_1 = 1\Omega$, $R_2 = 2\Omega$, $R_3 = 1\Omega$, $R_4 = 3\Omega$, $R_5 = 2\Omega$, $R_6 = 4\Omega$, $E_3 = 10\mathrm{V}$, $E_4 = 20\mathrm{V}$, $E_5 = 10\mathrm{V}$, $E_6 = 30\mathrm{V}$.

   Se cer:

   - Să se precizeze numărul de noduri, numărul de laturi şi, pentru fiecare latură, nodul iniţial, nodul final, rezistenţa şi t.e.m. a sursei de tensiune.

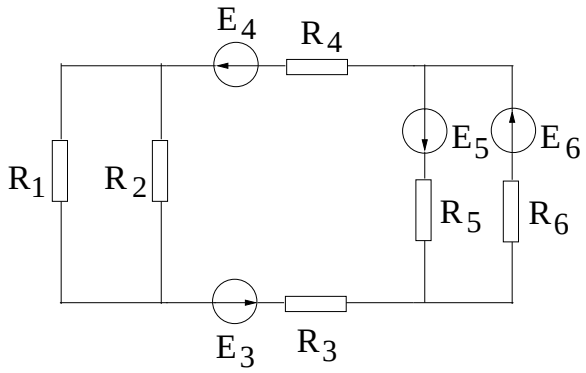

Fig. 7. Circuit rezistiv liniar de curent continuu

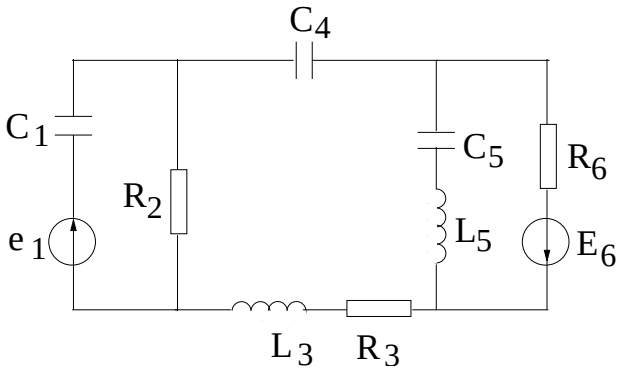

Fig. 8. Circuit rezistiv liniar de curent alternativ

- Ce dimensiuni au matricea **G** şi vectorul $\mathbf{i_S}$ asamblate de algoritm?
- Să se determine contribuţiile laturilor 2 şi 3 la matricea **G** şi vectorul $\mathbf{i_S}$.
- Care sunt matricea **G** şi vectorul $\mathbf{i_S}$ asamblate?
- Care este sistemul de ecuaţii de rezolvat?

2. Fie circuitul de curent alternativ din figura 8: Se cunosc: frecvenţa $f = 50$Hz, $R_2 = 5\Omega$, $R_3 = 2\Omega$, $R_6 = 3\Omega$, $L_3 = \frac{2}{100\pi}$H, $L_5 = \frac{4}{100\pi}$H, $C_1 = \frac{1}{300\pi}$F, $C_4 = \frac{1}{100\pi}$F, $C_5 = \frac{1}{200\pi}$F, $e_1(t) = 20\sqrt{2}\sin(\omega t + \pi)$V, $e_6(t) = 10\sin(\omega t + \frac{3\pi}{2})$V.

   Să se precizeze numărul de noduri, numărul de laturi şi, pentru fiecare latură, nodul iniţial, nodul final, impedanţa şi valoarea complexă a t.e.m.

5.8 Întrebări şi probleme

1. Cum trebuie modificat algoritmul de analiză a circuitelor electrice rezistive liniare, pentru a admite şi surse ideale de curent, conectate în paralel cu fiecare latură?

2. Extindeţi algoritmul de analiză a circuitelor electrice rezistive liniare pentru a admite şi surse comandate liniar în tensiune.

3. Modificaţi algoritmul de analiză nodală, astfel încât să admită şi elemente comandate în curent (surse de tensiune sau surse comandate liniar în curent).

4. Modificaţi algoritmul de analiză în curent alternativ, astfel încât aceasta să admită şi bobine cuplate mutual.

5. Analizaţi comparativ cele două metode de descriere a unui circuit: cea orientată pe elemente (se dau nodurile fiecărui element) şi cea orientată pe noduri (se dau elementele, care concură la fiecare nod).

6. Analizaţi modul în care se pot folosi tehnicile de matrice rare la analiza circuitelor electrice liniare.

7. Generaţi un algoritm de analiză a circuitelor de curent alternativ, pentru un sistem care nu admite operaţii cu numere complexe.

8. Generaţi un algoritm de analiză a datelor de descriere a unui circuit, care permite simbolizarea nodurilor prin nume alfanumerice şi care testează corectitudinea descrierii.

9. Comparaţi diferitele metode de scriere ale ecuaţiilor unui circuit electric rezistiv liniar.

Gustav Robert Kirchhoff (1824, Prusia - 1887, Germania)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Kirchhoff.html
http://scienceworld.wolfram.com/biography/Kirchhoff.html

Lucrarea 6

Interpolarea polinomială a funcţiilor reale

6.1 Caracterizarea metodei

Funcţiile numerice se pot reprezenta în sistemele de calcul prin două metode principial diferite:

- prin cod, indicând algoritmul care permite evaluarea funcţiei în orice punct al domeniului de definiţie;

- prin date, indicând valorile funcţiei numai într-o reţea de puncte din domeniul de definiţie, numite noduri.

Evaluarea unei *funcţii reprezentate tabelar (prin date)* presupune aproximarea ei (interpolarea) în intervalele dintre nodurile reţelei, în orice punct al domeniului de definiţie. Una din cele mai simple metode de interpolare constă în aproximarea funcţiei cu un polinom. În acest caz evaluarea funcţiei se reduce la operaţii aritmetice elementare (adunări şi înmulţiri).

Scopul lucrării este de a evidenţia cele mai eficiente metode de determinare a polinomului de interpolare. În lucrare se studiază efortul de calcul, eroarea introdusă prin interpolare polinomială şi limitele acestei metode de aproximare.

6.2 Principiul metodei

Se consideră funcţia reală de variabilă reală $f : [a, b] \to \mathbb{R}$ ale cărei valori sunt cunoscute într-o reţea de noduri $a = x_0 < x_1 < ... < x_n = b$:

$$\begin{array}{lcccc} x: & x_0 & x_1 & \dots & x_n \\ y: & y_0 & y_1 & \dots & y_n \end{array}$$

în care $y_k = f(x_k)$.

Problema fundamentală a interpolării constă în determinarea unei funcţii $g : [a, b] \to \mathbb{R}$, care aproximează funcţia f satisfăcând condiţiile: $g(x_0) = y_0, g(x_1) = y_1, ..., g(x_n) = y_n$. De obicei funcţia g este căutată de forma unei combinaţii liniare

$$g(x) = \sum_{k=0}^{n} c_k b_k(x) \tag{6.1}$$

de funcţii $b_k : [a, b] \to \mathbb{R}$, numite funcţii de bază. Dacă în general problema interpolării nu are soluţie unică, prin alegerea funcţiilor de bază problema este bine formulată şi are soluţie unică, cu condiţia ca aceste funcţii să fie liniar independente. În acest caz, problema interpolării se reduce la determinarea coeficienţilor $c_0, c_1, \dots, c_n$, care alcătuiesc vectorul $\mathbf{c} = [c_0, c_1, \dots, c_n]^T \in \mathbb{R}^{(n+1)}$.

Metoda clasică

Alegând funcţiile de bază de forma $1, x, x^2, \dots, x^n$, respectiv $b_k = x^k$, *funcţia de interpolare este un polinom*:

$$g(x) = \sum_{k=0}^{n} c_k x^k,$$

de gradul n care satisface *condiţiile de interpolare*:

$$g(x_k) = y_k, \quad k = 0, 1, 2, \dots, n.$$

Există deci o legătură strânsă între gradul polinomului de interpolare şi numărul de puncte ale tabelului de valori, şi anume gradul polinomului este cu 1 mai mic decât numărul de puncte din tabel (prin două puncte trece o dreaptă, prin trei puncte trece o parabolă, etc.).

În consecinţă, coeficienţii polinomului de interpolare satisfac sistemul de ecuaţii algebrice liniare:

$$\begin{array}{rcl} c_0 + c_1 x_0 + c_2 x_0^2 + \dots + c_n x_0^n &=& y_0 \\ c_0 + c_1 x_1 + c_2 x_1^2 + \dots + c_n x_1^n &=& y_1 \\ \vdots & & \\ c_0 + c_1 x_n + c_2 x_n^2 + \dots + c_n x_n^n &=& y_n \end{array} \tag{6.2}$$

sau sub forma matriceală:

$$\mathbf{Ac} = \mathbf{y},$$

în care $\mathbf{y} = [y_0, y_1, \ldots, y_n]^T \in \mathbb{R}^{(n+1)}$ iar

$$\mathbf{A} = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)} \tag{6.3}$$

este o matrice nesingulară, dacă $x_i \neq x_j$ pentru $i \neq j$.

În acest fel, problema interpolării presupune parcurgerea etapelor:

- determinarea coeficienţilor polinomului de interpolare prin rezolvarea unui sistem liniar de ecuaţii algebrice;
- evaluarea polinomului interpolant.

Această tehnică de interpolare poate fi aplicată doar pentru valori mici ale gradului ($n < 5$), deoarece are două mari dezavantaje:

1. efortul de calcul pentru determinarea coeficienţilor este relativ mare, ordinul de complexitate al celui mai eficient algoritm de rezolvare a unui sistem liniar general fiind $O(2n^3/3)$;
2. erorile soluţiei sunt mari, deoarece sistemul este slab condiţionat pentru valori mari ale gradului n.

Metoda Lagrange

O metodă care evită aceste dezavantaje este metoda Lagrange, în care funcţiile de bază se aleg de forma:

$$\begin{aligned} b_0(x) &= (x - x_1)(x - x_2)\cdots(x - x_n) \\ b_1(x) &= (x - x_0)(x - x_2)\cdots(x - x_n) \\ &\vdots \\ b_n(x) &= (x - x_0)(x - x_1)\cdots(x - x_{n-1}) \end{aligned} \tag{6.4}$$

sau în general:

$$b_k(x) = \prod_{i=0, i\neq k}^{n} (x - x_i) \tag{6.5}$$

Impunând polinomului

$$g(x) = \sum_{k=0}^{n} c_k \prod_{i=0, i\neq k}^{n} (x - x_i) \tag{6.6}$$

condiţiile de interpolare, rezultă sistemul de ecuaţii algebrice liniare:

$$\begin{aligned} g(x_0) &= c_0(x_0 - x_1)(x_0 - x_2)\cdots(x_0 - x_n) = y_0 \\ g(x_1) &= c_1(x_1 - x_0)(x_1 - x_2)\cdots(x_1 - x_n) = y_1 \\ &\vdots \\ g(x_n) &= c_n(x_n - x_0)\cdot(x_n - x_1)\cdots(x_n - x_{n-1}) = y_n \end{aligned} \tag{6.7}$$

cu structura diagonală. Soluţia acestui sistem este:

$$c_k = \frac{y_k}{\prod_{i=0, i\neq k}^{n}(x_k - x_i)}, \tag{6.8}$$

iar polinomul de interpolare are expresia

$$g(x) = \sum_{k=0}^{n} y_k \prod_{i=0, i\neq k}^{n} \frac{(x - x_i)}{(x_k - x_i)} = \sum_{k=0}^{n} y_k l_k(x), \tag{6.9}$$

în care s-a notat cu $l_k(x)$ polinomul lui Lagrange.

Metoda Lagrange elimină dezavantajele metodei "clasice", în schimb timpul necesar evaluării polinomului de interpolare creşte de la ordinul liniar $O(n)$ la cel pătratic $O(n^2)$.

Metoda Newton

O altă metodă pentru determinarea polinomului de interpolare este metoda Newton, în care funcţiile de bază se aleg de forma:

$$\begin{aligned} b_0(x) &= 1 \\ b_1(x) &= (x - x_0) \\ b_2(x) &= (x - x_0)(x - x_1) \\ &\vdots \\ b_n(x) &= (x - x_0)(x - x_1)\cdots(x - x_{n-1}) \end{aligned} \tag{6.10}$$

sau în general:

$$b_k(x) = \prod_{i=0}^{k-1}(x - x_i).$$

Impunând polinomului

$$g(k) = \sum_{k=0}^{n} c_k \prod_{i=0}^{k-1}(x - x_i) \tag{6.11}$$

condiţiile de interpolare:

$$\begin{aligned} g(x_0) &= c_0 = y_0 \\ g(x_1) &= c_0 + c_1(x_1 - x_0) = y_1 \\ g(x_2) &= c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) = y_2 \\ &\vdots \\ g(x_n) &= c_0 + c_1(x_n - x_0) + c_2(x_n - x_0)(x_n - x_1) + \cdots = y_n \end{aligned} \tag{6.12}$$

rezultă un sistem algebric liniar cu structura triunghiular inferioară, a cărui rezolvare poate fi realizată prin metoda substituţiei progresive. Coeficienţii polinomului de interpolare Newton sunt:

$$\begin{aligned} c_0 &= y_0 \\ c_1 &= (y_1 - c_0)/(x_1 - x_0) = (y_1 - y_0)(x_1 - x_0) \\ c_2 &= (y_2 - c_0 - c_1(x_2 - x_0))/(x_2 - x_0)/(x_2 - x_1) \\ &\vdots \\ c_n &= (y_n - c_0 - c_1(x_n - x_0) - c_2(x_n - x_0)(x_n - x_1)\cdots)/(x_n - x_0)/\cdots/(x_n - x_{n-1}) \end{aligned} \tag{6.13}$$

Un concept aflat într-o strânsă relaţie cu polinomul de interpolare Newton este cel al *diferenţelor divizate de ordinul k ale unei funcţii f*, valori notate cu $f[\cdots]$ şi care se definesc recursiv prin:

$$f[x_0, x_1, \cdots, x_k] = \frac{f[x_1, \cdots, x_k] - f[x_0, x_1, \cdots, x_{k-1}]}{x_k - x_0}, \tag{6.14}$$

iar în particular, în cazul diferenţei divizate de ordinul 1:

$$f[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0}. \tag{6.15}$$

Se constată că valorile coeficienţilor c_k din polinomul de interpolare Newton sunt date de diferenţele divizate

$$c_k = f[x_0, x_1, \cdots, x_n]$$

şi:

$$g(x) = \sum_{k=0}^{n} c_k b_k(x) = \sum_{k=0}^{n} f[x_0, x_1, \cdots, x_k] \prod_{i=0}^{k-1} (x - x_i). \tag{6.16}$$

Determinarea coeficienţilor polinomului Newton este facilitată de utilizarea tabelei diferenţelor divizate, în care elementele se calculează recursiv:

x	y	ord.1	ord.2	ord.3	ord.4
x_0	y_0				
		$f[x_0, x_1]$			
x_1	y_1		$f[x_0, x_1, x_2]$		
		$f[x_1, x_2]$		$f[x_0, \ldots, x_3]$	
x_2	y_2		$f[x_1, x_2, x_3]$		$f[x_0, \ldots, x_4]$
		$f[x_2, x_3]$		$f[x_1, \ldots, x_4]$	
x_3	y_3		$f[x_2, x_3, x_4]$		
		$f[x_3, x_4]$			
x_4	y_4				
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$

Prima "linie" a acestui tabel conţine chiar coeficienţii polinomului de interpolare Newton. Diferenţele divizate de diferite ordine reprezintă aproximări ale derivatelor de ordin corespunzător ale funcţiei în sensul că dacă $f(x)$ are m derivate pe intervalul $[a, b]$ există un punct $a \leq z \leq b$ astfel încât:

$$f[x_0, x_1, \cdots, x_m] = \frac{f^{(m)}(z)}{m!} \tag{6.17}$$

în care $x_0, x_1, \ldots, x_m$ sunt puncte distincte din intervalul $[a, b]$.

Metoda de interpolare Newton are următoarele avantaje:

- realizează un compromis optim între efortul de construcţie şi cel de evaluare;
- algoritmul este relativ stabil din punct de vedere numeric, având erori numerice acceptabile ale rezultatelor;
- permite mărirea gradului polinomului de interpolare, prin adăugarea unui nod nou în reţeaua de interpolare, cu reutilizarea coeficienţilor de la gradul anterior, care nu se modifică, deci cu un efort minim de calcul;
- are posibilitatea de a controla eroarea de interpolare;
- prin mărirea succesivă a gradului polinomului de interpolare până la atingerea preciziei dorite, timpul de calcul este dependent de eroarea impusă, având valori mari doar în cazurile în care se doreşte o precizie ridicată;
- coeficienţii polinomului Newton reprezintă diferenţele divizate de interpolat, ceea ce facilitează calculul numeric al polinomului de interpolare.

Observaţie:

Cele trei metode prezentate (clasică, Lagrange, Newton) sunt metode de interpolare globală. Ele caută un polinom de grad n ce trece prin cele $n + 1$ puncte ale tabelului de

date. Deoarece acest polinom este unic (de exemplu, dacă $n = 1$, există o dreaptă unică ce trece prin cele două puncte, dacă $n = 2$, există o parabolă unică ce trece prin cele trei puncte, etc.), cele trei metode rulate pe un calculator ideal, de precizie infinită (în care nu există erori de rotunjire) ar da acelaşi rezultat.

6.3 Pseudocodul algoritmilor

Metoda clasică

Determinarea coeficienţilor polinomului de interpolare se poate face cu următoarea procedură, care apelează la rândul ei procedura Gauss, de rezolvare a sistemelor algebrice liniare.

```
procedura interp(n, x, y, c)      ; determină coeficienții polinomului de interpolare
    întreg n                      ; gradul polinomului de interpolare
    tablou real x(n)              ; abscisele punctelor de interpolare
    tablou real y(n)              ; ordonatele punctelor de interpolare
    tablou real c(n)              ; coeficienții polinomului (date de ieșire), indicii sunt de la 0 la n
    tablou real a(n, n)           ; matricea sistemului, cu indici de la 0 la n
    pentru i = 0, n
        a_i0 = 1
    pentru j = 1, n
        pentru i = 0, n
            a_ij = a_ij-1 x_i
    Gauss0(n+1, a, y, c)          ; rezolvă sistemul liniar ac = y cu n + 1 ecuații
retur
```

Atenţie: Tabloul de date are indici de la 0 la n. Procedura Gauss implementată în lucrarea 3 folosea indici de la 1 la n. Ea trebuie rescrisă pentru a trata matrice în care notaţia indicelor este de la 0 la n. De aceea pseudocodul de mai sus apelează o funcţie `Gauss0`.

Pentru evaluarea polinomului se poate folosi rutina evalp prezentată în lucrarea 1.

Metoda Lagrange fără pregătire

Procedura de interpolare Lagrange admite următoarea reprezentare în pseudocod:

```
funcția interp-L (n, x, y, xcrt)   ; evaluează polinomul de interpolare
                                   ; Lagrange în punctul xcrt
    întreg n                       ; gradul polinomului
    tablou real x(n), y(n)         ; rețeaua de interpolare, indici de la 0 la n
```

```
    real xcrt                    ; variabilă independentă
    real ycrt                    ; valoarea polinomului în xcrt
    real p                       ; variabilă intermediară
    ycrt = 0
    pentru k = 0, n
        p = 1
        pentru j = 0, n
            dacă j ≠ k atunci
                p = p · (x − x_j)/(x_k − x_j)
        ycrt = ycrt + y_k · p
întoarce ycrt
```

Metoda Lagrange cu pregătire

Dacă polinomul este evaluat într-un număr mare de puncte, diferite de nodurile reţelei de interpolare, atunci este avantajoasă următoarea formă a polinomului de interpolare (6.6):

$$g(x) = (x - x_0)(x - x_1) \cdots (x - x_n) \sum_{k=0}^{n} \left(\frac{c_k}{x - x_k} \right), \tag{6.18}$$

în care c_k este dat de relaţia (6.8).

Această metodă admite următorul pseudocod:

```
procedura prep-L(n, x, y, c)     ; pregătește datele pentru interpolarea Lagrange
    întreg n                     ; gradul polinomului
    tablou real x(n), y(n)       ; rețeaua de interpolare, indici de la 0 la n
        c(n)                     ; coeficienții polinomului de ieșire, indici de la 0 la n
    pentru k = 0, n
        c_k = y_k
        pentru j = 0, n
            dacă j ≠ k atunci
                c_k = c_k/(x_k − x_j)
retur
```

```
  funcția eval-L (n, x, y, xcrt)   ; evaluează polinomul Lagrange în punctul xcrt cu
                                   ; coeficienții c calculați cu prep-L
        întreg n                   ; gradul polinomului
        tablou real x(n),          ; abscisele punctelor de interpolare, indici de la 0 la n
        tablou real c(n)           ; coeficienții polinomului, indici de la 0 la n
        real xcrt                  ; varibilă independentă
        p = 1
```

pentru $k = 0, n$
$p = p(xcrt - x_k)$
dacă $p = 0$ **atunci întoarce** y_k
$ycrt = 0$
pentru $k = 0, n$
$ycrt = ycrt + c_k/(xcrt - x_k)$
$ycrt = p \cdot ycrt$
întoarce $ycrt$

Metoda Newton

Problema determinării coeficienţilor polinomului Newton are o soluţie dependentă de reprezentarea datelor.

Dacă se generează diferenţele divizate şi se memorează într-un tabel bidimensional cu elementele:

$$a_{ij} = f[x_i \cdots, x_{i+j}]$$

pentru $i = 0, 1, \ldots, n$ şi $j = 0, 1, \ldots, (n - i)$ şi $a_{ij} = 0$ pentru $j > n - i$, atunci pe prima linie a acestui tabel se obţin coeficienţii polinomului de interpolare Newton $c_k = a_{0k}$:

		$j=0$	$j=1$	$j=2$	$\cdots$	
$i=0:$	x_0	$f[x_0]$	$f[x_0,x_1]$	$f[x_0,x_1,x_2]$	$\cdots$	
$i=1:$	x_1	$f[x_1]$	$f[x_1,x_2]$	$f[x_1,x_2,x_3]$	$\cdots$	$=[a_{ij}]$
$i=2:$	x_2	$f[x_2]$	$f[x_2,x_3]$	$f[x_2,x_3,x_4]$	$\cdots$	
$\vdots$						
$i=n-1:$	x_{n-1}	$f[x_{n-1}]$	$f[x_{n-1},x_n]$	0	$\cdots$	
$i=n:$	x_n	$f[x_n]$	0	0	$\cdots$	

procedura prep-N (n, x, y, c) ; determină coeficienţii polinomului
; de interpolare Newton
întreg n ; gradul polinomului
tablou real $x(n)$, $y(n)$; reţeaua de interpolare, indici de la 0 la n
tablou real $c(n)$; coeficienţii polinomului, indici de la 0 la n
; (date de ieşire)
pentru $i = 0, n$
$a_{i0} = y_i$
pentru $j = 1, n$
pentru $i = 0, n - j$
$a_{ij} = (a_{i+j,j-1} - a_{i,j-1})/(x_{j+1} - x_i)$
pentru $i = 0, n$

```
        c_i = a_0 i
retur

  funcţia eval-N (n, x, y, xcrt)      ; evaluează polinomul Newton
        întreg n                      ; gradul polinomului
        tablou real x(n),             ; abscisele punctelor de interpolare, indici de la 0 la n
        tablou real c(n)              ; coeficienţii polinomului, indici de la 0 la n
        real xcrt                     ; varibilă independentă
        ycrt = c_n
        pentru k = n-1, 0, -1
              ycrt = c_k + (xcrt - x_k) ycrt
         întoarce ycrt
```

Evaluarea polinomului Newton se bazează pe observaţia ca acesta poate fi scris sub forma:

$$g(x) = c_0 + (x - x_0)(c_1 + (x - x_1)(c_2 + \cdots + (x - x_{n-2})(c_{n-1} + c_n(x - x_{n-1}) \cdots)$$

mult mai avantajoasă din punctul de vedere al efortului de calcul decât forma (6.11).

6.4 Analiza complexităţii algoritmilor

6.4.1 Efort de calcul

Metoda clasică de interpolare polinomială, bazată pe rezolvarea unui sistem de ecuaţii liniare necesită un efort de calcul de ordinul $O(2n^3/3)$ pentru determinarea coeficienţilor polinomului şi un efort liniar $O(2n)$ pentru evaluarea polinomului. Dacă se efectuează m evaluări, efortul global de calcul este $O(2n^3/3 + 2mn)$.

La *metoda Lagrange*, efortul de calcul pentru evaluarea polinomului Lagrange într-un punct cu relaţia (6.9) este de ordinul $O(4n^2)$. Dacă se utilizează relaţia (6.18), atunci pregătirea datelor (procedura prep-L) necesită un efort de calcul de ordinul $O(2n^2)$, iar evaluarea (funcţia eval-L) necesită un efort de calcul de ordinul $O(5n)$, deci un efort de calcul global cu ordinul $O(2n^2 + 5mn)$.

În cazul *metodei Newton* efortul de pregătire are ordinul $O(3n^2/2)$, iar efortul de evaluare are ordinul $O(2n)$, ceea ce corespunde unui efort de calcul global de ordinul $O(3n^2/2 + 2mn)$.

Următorul tabel sintetizează rezultatele obţinute şi evidenţiază eficienţa metodei Newton:

Metoda	Timp de pregătire	Timp de evaluare	Timp total
Clasică	$2n^3/3$	$2n$	$2n^3/3 + 2nm$
Lagrange (6.9)	–	$4n^2$	$4n^2m$
Lagrange (6.18)	$2n^2$	$5n$	$2n^2 + 5nm$
Newton	$3n^2/2$	$2n$	$3n^2/2 + 2nm$

6.4.2 Necesar de memorie

Din punctul de vedere al eficienţei spaţiale metoda clasică necesită un spaţiu de memorie de ordine $O(n^2)$ pe când în cazul metodei Lagrange spaţiul de memorie necesar are ordinul liniar $O(2n)$ pentru varianta (6.9) şi $O(3n)$ pentru varianta (6.18).

Metoda Newton, aşa cum a fost descrisă în procedura `prep-N`, necesită un spaţiu de memorie $O(n^2)$ pentru determinarea coeficienţilor. Deoarece jumătate din acest spaţiu este neutilizat, structura de date poate fi modificată prin memorarea diferenţelor divizate într-un tablou unidimensional, ceea ce conduce la un necesar de memorie de ordinul $O(n^2/2)$.

6.5 Eroarea de interpolare

Problema generală a interpolării nu are soluţie unică, existând o infinitate de funcţii $g(x)$, care interpolează funcţia $f(x)$ pe o reţea dată. Mai mult, abaterea dintre funcţia "exactă" $f(x)$ şi aproximarea sa $g(x)$ poate fi făcută oricât de mare, chiar dacă $g(x_k) = f(x_k)$ pentru $k = 0, 1, \cdots, n$. Dacă în schimb, funcţia $f(x)$ are derivate de ordin superior pe intervalul $[a, b]$, atunci eroarea de aproximare a ei cu un polinom interpolat $g(x)$ este mărginită.

Dacă se notează cu:

$$e(x) = f(x) - g(x),$$

eroarea absolută, în care $g(x)$ este polinomul de interpolare pe reţeaua $x_0, x_1, \ldots, x_n$, atunci:

$$e(x) = f[x_0, x_1, \ldots, x_n, x]. \tag{6.19}$$

Folosind relaţiile (6.16) şi (6.17) rezultă că există un punct $z \in [a, b]$ astfel încât:

$$e(x) = \frac{f^{(n+1)}(z)}{(n+1)!} \prod_{j=0}^{n} (x - x_j). \tag{6.20}$$

Dacă $|f^{(n+1)}(x)| \leq M$, pentru orice $a \leq x \leq b$, atunci eroarea de interpolare satisface inegalitatea:

$$|e(x)| \leq \frac{M}{(n+1)!} \prod_{j=0}^{n} |x - x_j|. \tag{6.21}$$

În particular, în cazul interpolării liniare pe reţeaua cu două puncte x_0, x_1 eroarea de interpolare satisface inegalitatea:

$$|e(x)| \leq \frac{M_2}{2}|x - x_0||x - x_1|,$$

în care $M_2 > |f''(z)|$ cu $x_0 < z < x_1$. Se constată că eroarea de interpolare se anulează în nodurile reţelei x_0, x_1 şi că nu poate depăşi valoarea:

$$|e(x)| \leq \frac{M_2}{2}h^2,$$

în care s-a notat cu $h = x_1 - x_0$ lungimea intervalului.

Interpolarea parabolică (cu polinom de gradul doi) va avea o eroare care depinde de valoarea maximă a modulului derivatei a treia a funcţiei de interpolat. În general, dacă $h = b - a$, rezultă inegalitatea:

$$|e(x)| \leq \frac{M_{n+1}}{(n+1)!}h^{n+1}, \tag{6.22}$$

care evidenţiază dependenţa erorii de lungimea h a intervalului de interpolat.

Dacă marginile $M_2, M_3, M_4, \ldots$ ale derivatelor alcătuiesc un şir monoton descrescător ($M_{k+1} \leq M_k$), atunci relaţiile (6.19) şi (6.21) evidenţiează faptul că ultimul termen din polinomul interpolat Newton reprezintă o aproximare a erorii de interpolare pentru polinomul de grad $n - 1$.

Din rezultatele prezentate s-ar părea că mărind gradul polinomului de interpolare, eroarea de interpolare trebuie să scadă. În realitate, această afirmaţie nu este exactă. Un exemplu celebru este cel dat de Runge, care a arătat că pentru funcţia $f : [-5, 5] \to \mathbb{R}$, definită prin:

$$f(x) = \frac{1}{1 + x^2},$$

care este o funcţie suficient de netedă, interpolarea pe o reţea uniformă de noduri este numeric instabilă. Pe măsură ce creşte numărul de noduri (şi implicit gradul polinomului) creşte şi eroarea de interpolare, datorită oscilaţiei polinomului interpolat între punctele reţelei. Această lipsă de convergenţă a polinomului de grad *ni* către $f(x)$, când gradul n tinde către infinit este cunoscută sub numele de *efectul Runge*.

Pentru a elimina efectul Runge se recomandă alegerea nodurilor de interpolare în poziţia corespunzătoare rădăcinilor polinomului Cebîşev:

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{(n-k)\pi}{n}\right), \tag{6.23}$$

unde $k = 0, 1, \ldots, n$

Erorile prezentate sunt în fond *erori de trunchiere*, dar într-o interpolare apar şi erori inerente sau de rotunjire.

Erorile inerente se datorează abaterilor datelor de intrare de la valoarea exactă. Interpolarea polinomială având un caracter global, o abatere într-un nod x_k se face simţită nu numai local, în vecinătatea nodului x_k ci pe întreg domeniu $[a, b]$. Dacă de exemplu $y_k^* = y_k + e_y$ este afectată de eroarea e_y, atunci rezultatul interpolării $g(x)^* = g(x) + e_g$ are o eroare:

$$e_g = e_y \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j},$$

care se anulează în nodurile x_j cu j diferit de k, deci este chiar polinomul Lagrange $l_k(x)$ multiplicat cu e_y. Acest polinom poate lua în intervalul $[a, b]$ valori mult mai mari ca e_y. Alegând nodurile conform rădăcinilor polinomului Cebîşev, atunci polinomul Lagrange $l_k(x)$ se identifică cu polinomul Cebîşev de grad k, care are oscilaţie minimă în intervalul $[a, b]$. În consecinţă, interpolarea Cebîşev asigură nu numai erori de trunchiere ci şi erori de rotunjire minime.

Spre deosebire de erorile de trunchiere, *erorile de rotunjire* depind de metoda adoptată. Deoarece polinomul de interpolare al unei reţele de date este unic (nu depinde de metoda prin care a fost calculat), erorile de aproximare în cazul operării cu un sistem de calcul ideal (fără erori de rotunjire) sunt aceleaşi indiferent dacă polinomul a fost determinat prin metoda clasică, Lagrange sau Newton.

Modul în care se propagă erorile de rotunjire depinde de stabilitatea numerică a metodei. Metoda Lagrange este foarte stabilă numeric, deoarece elementele bazei sunt ortogonale şi sistemul de rezolvat (cu structură diagonală) este foarte bine condiţionat. Metoda "clasică" este foarte instabilă deoarece sistemul care se rezolvă este slab condiţionat, mai ales pentru polinoame de grad mare. Metoda Newton prezintă o stabilitate numerică acceptabilă.

Erorile prezentate limitează drastic *utilizarea interpolării polinomiale globale*. Nu este recomandabilă această metodă pentru interpolarea datelor experimentale, succeptibile de erori mari. Ea poate fi aplicată cu succes la interpolarea funcţiilor evaluate algoritmic, care sunt suficient de netede (de preferinţă analitice), fără modificări prea rapide ale pantei (de preferinţă la funcţii monotone). Dacă nodurile reţelei de interpolare pot fi alese de utilizator, atunci se recomandă ca ele să fie mai dese spre marginile intervalului, de preferinţă conform relaţiei (6.23). Din păcate, nici alegerea nodurilor conform acestei relaţii nu rezolvă complet problema interpolării polinomiale. Faber a demonstrat o teoremă care afirmă că *pentru orice şir de reţele de discretizare cu un număr de noduri n tinzând către infinit există cel puţin o funcţie continuă pentru care procedura de interpolare este divergentă, în sensul că eroarea de interpolare tinde la infinit.* Această teoremă evidenţiează limitarea majoră a metodei interpolării polinomiale globale.

6.6 Chestiuni de studiat

1. Interpolarea polinomială a funcţiilor pe reţele uniforme sau neuniforme (conform radăcinilor polinoamelor Cebîşev);
2. Analiza experimentală a erorilor;
3. Analiza experimentală a timpului de calcul;
4. Implementarea şi testarea unor algoritmi de interpolare polinomială;
5. Căutare de informaţii pe Internet.

6.7 Mod de lucru

Pentru desfăşurarea lucrării selectaţi opţiunea *Interpolarea polinomială a funcţiilor reale* din meniul principal de lucrări. Aceasta are ca urmare lansarea următorului meniu:

- Interpolarea polinomială uniformă /Cebîşev
- Analiza algoritmilor - erori
- Analiza algoritmilor - timpi de calcul

6.7.1 Interpolarea polinomială a funcţiilor pe reţele uniforme/Cebîşev

Se selectează din meniul principal opţiunea *Interpolare polinomială*, care lansează un program de interpolare mai întâi pe o reţea cu pas constant şi apoi pe o reţea cu pas neuniform, în concordanţă cu rădăcinile polinoamelor Cebîşev. Programul afişează lista funcţiilor $f(x)$ care pot fi interpolate:

- $\sin(x)$
- $\exp(x)$
- $\ln(|x|)$
- $\mathrm{th}(x)$
- Runge: $1/(1+x^2)$
- $|x|$

- $\sqrt{|x|}$

După selectarea funcţiei programul solicită limitele domeniului de definiţie $a \leq x \leq b$ şi gradul polinomului de interpolare. Se reprezintă grafic funcţia exactă $f(x)$ şi interpolarea sa polinomială $g(x)$ în ambele cazuri (uniform/neuniform). În consola Scilab se afişează eroarea de interpolare $\max_{a \leq x \leq b} |f(x) - g(x)|$.

Se vor studia interpolările polinomiale pentru cele 7 funcţii predefinite, cu diferite valori ale gradului polinomului. Pentru una din funcţii (în afara funcţiei sin) se va reprezenta grafic modul de variaţie a erorii în funcţie de numărul nodurilor de interpolare (de la 2 la 20). Se recomandă să se aleagă o funcţie pentru care se remarcă efectul Runge. Se vor comenta rezultatele obţinute.

6.7.2 Analiza experimentală a erorilor de interpolare

Se selectează opţiunea *Analiza algoritmilor - erori* din meniul principal al lucrării. Programul lansat determină eroarea de interpolare a funcţiei $\sin(x)$ pe o perioadă, în cazul folosirii metodelor de interpolare: clasică, Lagrange, Newton. Programul solicită valoarea iniţială, valoarea finală şi pasul pentru gradul polinomului de interpolare. Valorile recomandate pentru gradul polinomului sunt 1,2,3,4,5,6,7,8 (valoarea iniţială 1, valoarea finală 8, pas 1).

Se vor nota erorile şi se vor comenta rezultatele. Se va reprezenta grafic eroarea în funcţie de gradul polinomului de interpolare.

6.7.3 Analiza experimentală a timpului de calcul necesar interpolării polinomiale

Se selectează opţiunea *Analiza algoritmilor - timpi de calcul* din meniul principal al lucrării. Programul lansat determină timpul de calcul necesar interpolării funcţiei $\sin(x)$ pe o perioadă, în cazul folosirii metodelor de interpolare: clasică, Lagrange, Newton.

Singura dată de intrare este numărul de noduri n din reţeaua de interpolare. Se va reprezenta grafic modul de variţie a timpului de calcul (separat pentru pregătire şi interpolare) n, pentru n variind de la 20 la 100, cu pas 20.

6.7.4 Implementarea unor algoritmi de interpolare polinomială

Se va implementa în limbajul C algoritmul Lagrange de interpolare. Se va scrie un program de testare, care va permite introducerea datelor n - gradul polinomului de interpolare, x, y

- tabelele de date (cu $n+1$ puncte) şi afişarea rezultatelor interpolării la mijlocul distanţei dintre nodurile de interpolare x.

6.7.5 Căutare de informaţii pe Internet

Căutaţi pe Internet informaţii legate de interpolarea funcţiilor. Cuvinte cheie recomandate: *interpolation*, *fitting*.

6.8 Exemple

6.8.1 Exemple rezolvate

1. Fie funcţia $y = f(x)$ definită tabelar:

x	1	2	4
y	2	1	3

Se cer:

(a) Cât este gradul polinomului de interpolare globală?

(b) Care sunt condiţiile de interpolare?

(c) Care este sistemul de ecuaţii asamblat în metoda clasică de interpolare?

(d) Care sunt polinoamele Lagrange asociate diviziunii [1 2 4]?

(e) Care este polinomul de interpolare Lagrange?

(f) Care este tabelul de diferenţe divizate?

(g) Care este polinomul de interpolare Newton?

(h) Care este valoarea funcţiei în punctul $x = 3$?

Rezolvare:

(a) Gradul polinomului de interpolare globală $g(x)$ este cu o unitate mai mic decât numărul de noduri ale reţelei de discretizare în care este cunoscută funcţia $f(x)$. Pentru exemplul dat, funcţia $y = f(x)$ are valori date în 3 noduri, şi în consecinţă gradul polinomului de interpolare este $n = 2$.

(b) Condiţiile de interpolare sunt $g(x_k) = y_k$, $k = 0, n$. Aceste condiţii impun ca funcţia de interpolare $g(x)$ să aibă aceleaşi valori ca funcţia $f(x)$ în nodurile reţelei de discretizare:

$$\begin{aligned} g(x_0) = y_0 &\Longrightarrow g(1) = 2, \\ g(x_1) = y_1 &\Longrightarrow g(2) = 1, \\ g(x_2) = y_2 &\Longrightarrow g(4) = 3. \end{aligned} \tag{6.24}$$

(c) În metoda clasică de interpolare, funcţia de interpolare este $g(x) = \sum_{k=0}^{n} c_k x^k$.

În cazul acestei probleme, avem:

$$g(x) = c_0 + c_1 x + c_2 x^2. \tag{6.25}$$

Din cele $n+1$ condiţii de interpolare rezultă un sistem (6.2) de $n+1$ ecuaţii cu $n+1$ necunoscute (coeficienţii c_k, $k = 0, n$).

Conditiile de interpolare (6.24) scrise pentru funcţia $g(x)$ dată de (6.25) conduc la următorul sistem de ecuaţii:

$$\begin{cases} c_0 + c_1 + c_2 = 2 \\ c_0 + 2c_1 + 4c_2 = 1 \\ c_0 + 4c_1 + 16c_2 = 3 \end{cases}$$

(d) Polinomul Lagrange l_k asociat unei diviziuni x_i are următoarea formă:

$$l_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}.$$

Pentru exemplul considerat, diviziunea are 3 puncte, în consecinţă vor fi 3 polinoame Lagrange de grad 2:

$$l_0(x) = \prod_{i=0, i\neq 0}^{2} \frac{x - x_i}{x_k - x_i} = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} = \frac{(x-2)(x-4)}{(1-2)(1-4)} = \frac{x^2 - 6x + 8}{3},$$

$$l_1(x) = \prod_{i=0, i\neq 1}^{2} \frac{x - x_i}{x_k - x_i} = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} = \frac{(x-1)(x-4)}{(2-1)(2-4)} = \frac{x^2 - 5x + 4}{-2},$$

$$l_2(x) = \prod_{i=0, i\neq 2}^{2} \frac{x - x_i}{x_k - x_i} = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} = \frac{(x-1)(x-2)}{(4-1)(4-2)} = \frac{x^2 - 3x + 2}{6}.$$

(e) Polinomul de interpolare Lagrange este: $g(x) = \sum_{k=0}^{n} y_k l_k(x)$.

În cazul particular studiat:

$$g(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x),$$

şi înlocuind expresiile polinomului Lagrange de mai sus, rezultă:

$$g(x) = 2\frac{x^2 - 6x + 8}{3} + 1\frac{x^2 - 5x + 4}{-2} + 3\frac{x^2 - 3x + 2}{6} = \frac{2x^2 - 9x + 13}{3}.$$

(f) Tabelul de diferenţe divizate este:

x	y	ord.1	ord.2
$x_0 = 1$	$y_0 = 2$		
		$f[x_0, x_1] = -1$	
$x_1 = 2$	$y_1 = 1$		$f[x_0, x_1, x_2] = \frac{2}{3}$
		$f[x_1, x_2] = 1$	
$x_2 = 4$	$y_2 = 3$		

Diferenţele divizate de ordinul 1, respectiv ordinul 2 se calculează cu formula (6.14), astfel:

$$f[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0} = \frac{1-2}{2-1} = -1,$$

$$f[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1} = \frac{3-1}{4-2} = 1,$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{1-(-1)}{4-1} = \frac{2}{3}.$$

(g) Polinomul de interpolare Newton (6.16) este:

$$g(x) = \sum_{k=0}^{n} f[x_0, x_1, \cdots, x_k] \prod_{i=0}^{k-1} (x - x_i).$$

În cazul studiat:

$$g(x) = y_0 + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) =$$

$$= 2 + (-1)(x-1) + \frac{2}{3}(x-1)(x-2) = \frac{2x^2 - 9x + 13}{3}.$$

Observaţie: polinomul de interpolare globală $g(x)$ este unic, indiferent de metoda de interpolare utilizată. De aceea, rezultatul obţinut este identic cu rezultatul de la punctul (e).

(h) În punctul $x = 3$, polinomul de interpolare este:

$$g(3) = \frac{2 \cdot 3^2 - 9 \cdot 3 + 13}{3} = \frac{4}{3}.$$

2. Fie funcţia $y = f(x)$ definită tabelar:

x	2	3	5
y	5	7	11

Se cer:

(a) Cât este gradul polinomului de interpolare globală?

(b) Care sunt condiţiile de interpolare?

(c) Care este sistemul de ecuaţii asamblat în metoda clasică de interpolare?

(d) Care sunt polinoamele Lagrange asociate diviziunii [2 3 5]?

(e) Care este polinomul de interpolare Lagrange?

(f) Care este tabelul de diferenţe divizate?

(g) Care este polinomul de interpolare Newton?

(h) Care este valoarea funcţiei în punctul $x = 4$?

(i) Care este polinomul de interpolare în ipoteza adăugării punctului $(1, 0)$ în tabelul de valori?

Rezolvare:

(a) Deoarece reţeaua de discretizare are 3 noduri, gradul polinomului de interpolare este $n = 2$.

(b) Condiţiile de interpolare sunt:

$$\begin{aligned} g(2) &= 5, \\ g(3) &= 7, \\ g(5) &= 11. \end{aligned}$$

(c) Impunând condiţiile de interpolare expresiei $g(x) = c_0 + c_1 x + c_2 x^2$, rezultă sistemul de ecuaţii în metoda clasică de interpolare:

$$\left\{ \begin{array}{rcrcrcr} c_0 & + & 2c_1 & + & 4c_2 & = & 5 \\ c_0 & + & 3c_1 & + & 9c_2 & = & 7 \\ c_0 & + & 5c_1 & + & 25c_2 & = & 11 \end{array} \right.$$

(d) Polinoamele Lagrange sunt:

$$\begin{aligned} l_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-3)(x-5)}{(2-3)(2-5)} = \frac{x^2-8x+15}{3}, \\ l_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-2)(x-5)}{(3-2)(3-5)} = \frac{x^2-7x+10}{-2}, \\ l_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-2)(x-3)}{(5-2)(5-3)} = \frac{x^2-5x+6}{6}. \end{aligned}$$

(e) Polinomul de interpolare Lagrange este:

$$g(x) = y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x) = 5\frac{x^2-8x+15}{3} + 7\frac{x^2-7x+10}{-2} +$$

$$+11\frac{x^2-5x+6}{6} = \frac{12x+6}{6} = 2x+1.$$

Gradul polinomului de interpolare este 1, deşi avem 3 noduri în reţeaua de discretizare. Aceasta se explică prin faptul că cele trei noduri sunt situate pe dreapta $2x + 1 = y$.

(f) Tabelul de diferenţe divizate este:

x	y	ord.1	ord.2
$x_0 = 2$	$y_0 = 5$		
		$f[x_0, x_1] = 2$	
$x_1 = 3$	$y_1 = 7$		$f[x_0, x_1, x_2] = 0$
		$f[x_1, x_2] = 2$	
$x_2 = 5$	$y_2 = 11$		

Diferenţele divizate de ordinul 1, respectiv ordinul 2 sunt:

$$f[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0} = \frac{7 - 5}{3 - 2} = 2,$$

$$f[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1} = \frac{11 - 7}{5 - 3} = 2,$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{2 - 2}{5 - 2} = 0.$$

Ultima diferenţă divizată egală cu zero indică faptul că polinomul de interpolare nu este de grad 2.

(g) Polinomul de interpolare Newton este:

$$g(x) = y_0 + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) =$$

$$= 5 + 2(x - 2) + 0(x - 2)(x - 3) = 2x + 1.$$

(h) În punctul $x = 4$, polinomul de interpolare este:

$$g(4) = 2 \cdot 4 + 1 = 9.$$

(i) În situaţia în care tabelului de valori i se adaugă un punct, metoda Newton este cea mai avantajoasă metodă deoarece aceasta permite refolosirea calculelor deja efectuate.

Tabelul de diferenţe divizate se completează cu noua valoare astfel:

x	y	ord.1	ord.2	ord.3
2	5			
		2		
3	7		0	
		2		$\frac{3}{8}$
5	11		$-\frac{3}{8}$	
		$\frac{11}{4}$		
1	0			

Diferenţele divizate care apar prin adăugarea punctului $(1, 0)$ sunt:

$$f[x_2, x_3] = \frac{y_3 - y_2}{x_3 - x_2} = \frac{0 - 11}{1 - 5} = \frac{11}{4},$$

$$f[x_1, x_2, x_3] = \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1} = \frac{11/4 - 2}{1 - 3} = -\frac{3}{8},$$

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} = \frac{-3/8 - 0}{1 - 2} = \frac{3}{8}.$$

Polinomul de interpolare este:

$$h(x) = g(x) + \frac{3}{8}(x-2)(x-3)(x-5) = 2x + 1 + \frac{3}{8}(x^3 - 10x^2 + 31x - 30) =$$

$$= \frac{3}{8}x^3 - \frac{15}{4}x^2 + \frac{109}{8}x - \frac{41}{4}.$$

Se observă ca nu este nevoie ca punctele diviziunii să fie sortate. De asemenea, în cazul aplicării metodei clasice sau Lagrange, efortul de calcul efectuat anterior s-ar fi pierdut.

3. Fie funcţia $y = f(x)$ definită tabelar:

x	1	2	3	5
y	3	2	4	1

Se cer:

(a) Cât este gradul polinomului de interpolare globală?

(b) Care sunt condiţiile de interpolare?

(c) Care este sistemul de ecuaţii asamblat în metoda clasică de interpolare?

(d) Care sunt polinoamele Lagrange asociate diviziunii [1 2 3 5]?

(e) Care este polinomul de interpolare Lagrange?

(f) Care este tabelul de diferenţe divizate?

(g) Care este polinomul de interpolare Newton?

(h) Care este valoarea funcţiei în punctul $x = 4$?

Rezolvare:

(a) Deoarece reţeaua de discretizare are 4 noduri, gradul polinomului de interpolare este $n = 3$.

(b) Condiţiile de interpolare sunt;

$$\begin{aligned} g(1) &= 3, \\ g(2) &= 2, \\ g(3) &= 4, \\ g(5) &= 1. \end{aligned}$$

(c) Condiţiile de interpolare impuse polinomului $g(x) = c_0 + c_1x + c_2x^2 + c_3x^3$ conduc la sistemul de ecuaţii în metoda clasică de interpolare:

$$\left\{\begin{array}{rcl} c_0 + 1\cdot c_1 + 1^2\cdot c_2 + 1^3\cdot c_3 &=& 3 \\ c_0 + 2\cdot c_1 + 2^2\cdot c_2 + 2^3\cdot c_3 &=& 2 \\ c_0 + 3\cdot c_1 + 3^2\cdot c_2 + 3^3\cdot c_3 &=& 4 \\ c_0 + 5\cdot c_1 + 5^2\cdot c_2 + 5^3\cdot c_3 &=& 1 \end{array}\right. \Longrightarrow \left\{\begin{array}{rcrcrcrcl} c_0 &+& c_1 &+& c_2 &+& c_3 &=& 3 \\ c_0 &+& 2c_1 &+& 4c_2 &+& 8c_3 &=& 2 \\ c_0 &+& 3c_1 &+& 9c_2 &+& 27c_3 &=& 4 \\ c_0 &+& 5c_1 &+& 25c_2 &+& 125c_3 &=& 1 \end{array}\right.$$

(d) Polinoamele Lagrange sunt:

$$l_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-2)(x-3)(x-5)}{(1-2)(1-3)(1-5)} = \frac{x^3-10x^2+31x-30}{-8},$$

$$l_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x-1)(x-3)(x-5)}{(2-1)(2-3)(2-5)} = \frac{x^3-9x^2+23x-15}{3},$$

$$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x-1)(x-2)(x-5)}{(3-1)(3-2)(3-5)} = \frac{x^3-8x^2+17x-10}{-4},$$

$$l_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x-1)(x-2)(x-3)}{(5-1)(5-2)(5-3)} = \frac{x^3-6x^2+11x-6}{24}.$$

(e) Polinomul de interpolare Lagrange este:

$$g(x) = y_0l_0(x) + y_1l_1(x) + y_2l_2(x) = 3\frac{x^3-10x^2+31x-30}{-8} + 2\frac{x^3-9x^2+23x-15}{3} +$$
$$+4\frac{x^3-8x^2+17x-10}{-4} + 1\frac{x^3-6x^2+11x-6}{24} = \frac{-16x^3+132x^2-308x+264}{24} =$$
$$= -\frac{2}{3}x^3 + \frac{11}{2}x^2 - \frac{77}{6}x + 11.$$

(f) Tabelul de diferenţe divizate este:

x	y	ord.1	ord.2	ord.3
$x_0 = 1$	$y_0 = 3$			
		$f[x_0, x_1] = -1$		
$x_1 = 2$	$y_1 = 2$		$f[x_0, x_1, x_2] = \frac{3}{2}$	
		$f[x_1, x_2] = 2$		$f[x_0, x_1, x_2, x_3] = -\frac{2}{3}$
$x_2 = 3$	$y_2 = 4$		$f[x_1, x_2, x_3] = -\frac{7}{6}$	
		$f[x_2, x_3] = -\frac{3}{2}$		
$x_3 = 5$	$y_3 = 1$			

Diferenţele divizate de ordinele 1, 2 şi 3 sunt:

$$f[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0} = \frac{2-3}{2-1} = -1,$$

$$f[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1} = \frac{4-2}{3-2} = 2,$$

$$f[x_2, x_3] = \frac{y_3 - y_2}{x_3 - x_2} = \frac{1-4}{5-3} = -\frac{3}{2},$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} = \frac{2-(-1)}{3-1} = \frac{3}{2},$$

$$f[x_1, x_2, x_3] = \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1} = \frac{-\frac{3}{2} - 2}{5-2} = -\frac{7}{6},$$

$$f[x_0, x_1, x_2, x_3] = \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0} = \frac{-\frac{7}{6} - \frac{3}{2}}{5-1} = -\frac{2}{3}.$$

(g) Polinomul de interpolare Newton este:

$$g(x) = y_0 + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) +$$

$$+ f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2),$$

$$g(x) = 3 + (-1)(x-1) + \frac{3}{2}(x-1)(x-2) + \left(-\frac{2}{3}\right)(x-1)(x-2)(x-3) =$$

$$= -\frac{2}{3}x^3 + \frac{11}{2}x^2 - \frac{77}{6}x + 11.$$

(h) În punctul $x = 4$, polinomul de interpolare este:

$$g(4) = -\frac{2}{3}4^3 + \frac{11}{2}4^2 - \frac{77}{6}4 + 11 = 5.$$

6.8.2 Exemple propuse

1. Fie funcţia $y = f(x)$ definită tabelar:

x	2	4	5
y	1	3	2

Se cer:

(a) Cât este gradul polinomului de interpolare globală?

(b) Care sunt condiţiile de interpolare?

(c) Care este sistemul de ecuaţii asamblat în metoda clasică de interpolare?

(d) Care sunt polinoamele Lagrange asociate diviziunii [2 4 5]?

(e) Care este polinomul de interpolare Lagrange?

(f) Care este tabelul de diferenţe divizate?

(g) Care este polinomul de interpolare Newton?

(h) Care este valoarea funcţiei în punctul $x = 3$?

2. Fie funcţia $y = f(x)$ definită tabelar:

x	1	3	5
y	4	1	3

Se cer:

(a) Cât este gradul polinomului de interpolare globală?

(b) Care sunt condiţiile de interpolare?

(c) Care este sistemul de ecuaţii asamblat în metoda clasică de interpolare?

(d) Care sunt polinoamele Lagrange asociate diviziunii [1 3 5]?

(e) Care este polinomul de interpolare Lagrange?

(f) Care este tabelul de diferenţe divizate?

(g) Care este polinomul de interpolare Newton?

(h) Care este valoarea funcţiei în punctul $x = 2$?

(i) Care este polinomul de interpolare în ipoteza adăugării punctului $(4, 5)$ în tabelul de valori?

3. Fie funcţia $y = f(x)$ definită tabelar:

x	1	3	4	5
y	2	3	1	4

Se cer:

(a) Cât este gradul polinomului de interpolare globală?

(b) Care sunt condiţiile de interpolare?

(c) Care este sistemul de ecuaţii asamblat în metoda clasică de interpolare?

(d) Care sunt polinoamele Lagrange asociate diviziunii [1 3 4 5]?

(e) Care este polinomul de interpolare Lagrange?

(f) Care este tabelul de diferenţe divizate?

(g) Care este polinomul de interpolare Newton?

(h) Care este valoarea funcţiei în punctul $x = 2$?

6.9 Întrebări şi probleme

1. Evaluaţi stabilitatea numerică a metodei interpolării polinomiale.
2. Generaţi un algoritm de interpolare polinomială a unei funcţii de două variabile $f(x, y)$.
3. Implementaţi într-un limbaj de nivel înalt algoritmul de interpolare Newton.
4. Determinaţi o aproximare polinomială pentru caracteristica unei diode semiconductoare $i(u) = I_s(e^{u/vi} - 1)$, cu $I_s = 10^{-6}$ A, $v = 0,027$ V şi analizaţi eroarea introdusă prin interpolare.
5. Determinaţi o aproximare polinomială pentru caracteristica de magnetizare $B = \mu_0 H + B_s \text{th}(\mu H/B_s)$, cu $\mu_0 = 4\pi \cdot 10^{-7}$ H/m, $B_s = 1.7$ T, $\mu = 10^4 \mu_0$ şi analizaţi eroarea introdusă prin interpolare.
6. Analizaţi comportarea aproximării polinomiale în exteriorul domeniului $a \leq x \leq b$ (problema extrapolării).
7. Generaţi şi implementaţi un algoritm de interpolare a unei funcţii reale impare şi periodice printr-un polinom trigonometric de forma
$$g(x) = \sum_{k=1}^{n} a_k \sin(k\omega t).$$
8. Documentaţi-vă în literatura de specialitate şi pe internet ce înseamnă interpolarea polinomială pe portiuni (de tip spline).
9. Rezolvaţi problemele 4 şi 5 în mediul MATLAB/SCILAB.
10. Identificaţi în biblioteca matematică descrisa în [2] funcţii avansate pentru interpolări şi aproximari polinomiale, raţionale şi spline. Ce aduc nou aceste funcţii faţă de cea analizată în lucrare?

Joseph-Louis Lagrange (1736, Sardinia-Piedmont, acum Italia - 1813, Franţa

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Lagrange.html

Sir Isaac Newton (1643, Anglia - 1727, Anglia)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Newton.html

Carle David Tolme Runge (1856, Germania - 1927, Germania)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Runge.html

Lucrarea 7

Derivarea numerică a funcţiilor reale

7.1 Caracterizarea metodelor de derivare numerică

Metodele numerice de derivare au o largă aplicabilitate în inginerie, de exemplu atunci când expresia analitică a derivatei funcţiei este complicată şi cere un efort mare de evaluare sau atunci când funcţia are expresia necunoscută (valorile sunt determinate experimental sau rezultă dintr-un calcul numeric anterior). Aceste metode stau la baza rezolvării numerice a ecuaţiilor diferenţiale, foarte puţine din acestea putând fi rezolvate prin metode analitice.

Derivarea numerică se reduce la calcule aritmetice (adunări, scăderi, înmulţiri şi împărţiri), folosind formule relativ simple deduse prin aproximarea funcţiei de derivat printr-un polinom de interpolare. În acest fel, din punct de vedere teoretic, problema derivării numerice se reduce, practic, la problema interpolării funcţiilor.

În lucrare se studiază erorile de trunchiere şi rotunjire şi modul în care acestea depind de ordinul aproximării şi pasul reţelei de interpolare.

7.2 Principiile metodelor

Problema derivării numerice se formulează diferit în următoarele două cazuri:

- funcţia este definită prin cod (este cunoscută expresia ei analitică sau algoritmul de evaluare pentru orice punct din domeniul de definiţie);
- funcţia este definită prin date (este cunoscut tabelul valorilor funcţiei într-o reţea finită de puncte din domeniul de definiţie, numite noduri).

Fie o reţea de discretizare:

$$\begin{array}{lllll} x: & x_0, & x_1, & x_2, & \ldots, & x_n \\ f: & f_0, & f_1, & f_2, & \ldots, & f_n \end{array}$$

pe care sunt date valorile $f_k = f(x_k)$ ale unei funcţii reale $f : [x_0, x_n] \to \mathbb{R}$.

Dacă se notează cu $g : [x_0, x_n] \to \mathbb{R}$ o funcţie care interpolează datele anterioare (de exemplu, polinomul de interpolare de gradul n), atunci derivata funcţiei $f(x)$ în punctul x poate fi calculată aproximativ, evaluând $g'(x)$.

De exemplu, în cazul $n = 1$, polinomul de interpolare

$$g(x) = f_0 \frac{x - x_1}{x_0 - x_1} + f_1 \frac{x - x_0}{x_1 - x_0} \tag{7.1}$$

are derivata

$$g'(x) = \frac{f_1 - f_0}{x_1 - x_0} \tag{7.2}$$

iar în cazul $n = 2$:

$$\begin{aligned} g(x) &= f_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + f_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + f_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \\ g'(x) &= f_0 \frac{2x - (x_1 + x_2)}{(x_0 - x_1)(x_0 - x_2)} + f_1 \frac{2x - (x_0 + x_2)}{(x_1 - x_0)(x_1 - x_2)} + f_2 \frac{2x - (x_0 + x_1)}{(x_2 - x_0)(x_2 - x_1)} \end{aligned} \tag{7.3}$$

Calculele pot fi continuate pentru grade superioare, prin folosirea polinomului Newton de interpolare.

În cazul *reţelei de noduri echidistante* cu pasul

$$h = x_{i+1} - x_i, \tag{7.4}$$

aceste relaţii capătă o formă mult mai simplă:

$$g'(x) = \frac{f_1 - f_0}{h}, \tag{7.5}$$

pentru ordinul $n = 1$ şi

$$g'(x) = f_0 \frac{x - (x_0 + \frac{3h}{2})}{h^2} - 2f_1 \frac{x - (x_0 + h)}{h^2} + f_2 \frac{x - (x_0 + \frac{h}{2})}{h^2}, \tag{7.6}$$

pentru $n = 2$.

Polinomul de interpolare de grad doi devine, după derivare, o funcţie de gradul întâi care, în cele trei noduri ale reţelei are valorile:

$$g'(x_0) = \frac{1}{2h}(-3f_0 + 4f_1 - f_2); \tag{7.7}$$

$$g'(x_1) = \frac{1}{2h}(f_2 - f_0); \tag{7.8}$$

$$g'(x_2) = \frac{1}{2h}(f_0 - 4f_1 + 3f_2). \tag{7.9}$$

Aceste trei relaţii reprezintă aproximări de ordinul doi ale primei derivate, cunoscute sub numele de relaţii cu diferenţe finite *progresive*, *centrate* şi respectiv *regresive*. Aproximările de ordin doi oferă erori de trunchiere mai mici decât aproximările progresive şi regresive de ordinul întâi:

$$g'(x_0) = \frac{1}{h}(f_1 - f_0); \tag{7.10}$$

$$g'(x_1) = \frac{1}{h}(f_1 - f_0); \tag{7.11}$$

obţinute folosind polinomul de gradul $n = 1$.

Se constată că pentru aproximarea derivatei într-un nod al reţelei de discretizare se folosesc valorile funcţiei în nodurile vecine. Cu cât ordinul aproximării este mai mare, cu atât sunt utilizate mai multe noduri din vecinătatea iar calculul este mai complicat.

Derivatele de ordin superior se pot evalua numeric prin aplicarea recursivă a formulelor de calcul pentru prima derivată. De exemplu, folosind succesiv relaţia (7.8) a diferenţelor finite centrate de ordinul doi, pentru o discretizare locală cu pasul $h/2$ şi cu notaţiile: $f(x_i + h/2) = f_{i+1/2}$, $f(x_i - h/2) = f_{i-1/2}$, se obţine:

$$f''(x_i) = \frac{f'_{i+1/2} - f'_{i-1/2}}{2(h/2)} = \frac{\frac{f_{i+1}-f_i}{2(h/2)} - \frac{f_i-f_{i-1}}{2(h/2)}}{h} = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}, \tag{7.12}$$

rezultat care corespunde derivării directe a polinomului de interpolare $g(x)$, pentru aproximarea de ordinul doi. Acest rezultat are avantajul unei precizii maxime pentru reţeaua considerată şi pentru trei evaluări a funcţiei f.

În cazul unei *reţele neuniforme* cu: $x_1 - x_0 = h_1$, $x_2 - x_1 = h_2$, rezultă:

$$g''(x) = f_0\frac{2}{h_1(h_1 + h_2)} - f_1\frac{2}{h_1h_2} + f_2\frac{2}{(h_1 + h_2)h_2}. \tag{7.13}$$

Formulele de derivare numerică obţinute se pot aplica şi în calculul derivatelor parţiale.

Rezultatele obţinute pentru cazul funcţiilor definite tabelar se aplică şi în cazul funcţiilor definite prin cod, cu observaţia că: $f_i = f(x_i)$ şi $f_{i+1} = f(x_i + h)$, în care pasul h se alege astfel încât erorile numerice să fie cât mai mici.

7.3 Analiza algoritmilor

Analiza erorilor

Deoarece valoarea exactă a derivatei se obţine printr-un proces infinit de trecere la limită:

$$f'(x) = \lim_{h\to 0} \frac{f(x + h) - f(x)}{h}, \tag{7.14}$$

evaluarea numerică este afectată de o *eroare de trunchiere*. Teoretic, eroarea de trunchiere este cu atât mai mică cu cât ordinul aproximării este mai mare. În realitate, din cauza fenomenului Runge (evidenţiat la interpolarea polinomială), formulele de derivare de ordin superior sunt afectate, în anumite cazuri, de erori mari de trunchiere (mai ales la prelucrarea datelor experimentale). În acest caz se recomandă folosirea relaţiilor de ordin mic, cea mai indicată fiind relaţia cu diferenţe finite centrate de ordinul doi.

Erorile care apar prin folosirea formulelor de derivare numerică depind de doi factori:

- pasul de derivare h;
- gradul polinomului de interpolare n.

Pasul de derivare are o valoare optimă, finită, pentru care eroarea este minimă.

Pentru $h > h_{\text{optim}}$, predomină *eroarea de trunchiere*, datorată trunchierii seriei Taylor. În general, eroarea de trunchiere are ordinul $\mathrm{O}(h^n)$, deci scade cu micşorarea pasului h şi creşterea gradului n.

Pentru $h < h_{\text{optim}}$, predomină *eroarea de rotunjire* cauzată de faptul că sistemul nu poate reţine decât un număr limitat de zecimale. Această eroare creşte pe măsura scăderii pasului h, datorită fenomenului de "anulare prin scădere".

Pentru evaluarea erorii de trunchiere se foloseşte dezvoltarea în serie Taylor:

$$f(x_1) = f(x_0) + \frac{h}{1!}f'(x_0) + \frac{h^2}{2!}f''(z), \tag{7.15}$$

de unde rezultă:

$$f'(x_0) = \frac{f_1 - f_0}{h} - \frac{h}{2}f''(z) = g'(x_0) + \mathrm{O}(h), \tag{7.16}$$

deci o eroare de trunchiere $\mathrm{O}(h)$ pentru relaţia cu diferenţe finite de ordinul întâi.

Efort de calcul

Timpul de calcul creşte cu creşterea ordinului n al aproximării, fiind necesare $n + 1$ evaluări ale funcţiei f pentru calculul primei derivate.

Derivatele de ordin superior se pot calcula numeric prin aplicarea recursivă a formulei de derivare de ordinul întâi, dar erorile sunt tot mai mari pentru derivatele de ordin superior, generând importante *instabilităţi numerice*.

7.4 Pseudocodul metodei

Metodele de derivare numerică se transpun uşor în orice limbaj de programare, datorită simplităţii formulelor de calcul.

Ca date de intrare pentru calculul derivatei funcţiei f sunt necesare doar:

- punctul x_0 (în care se calculează derivata);
- pasul de derivare h (distanţa între două puncte ale reţelei de noduri).

Dacă funcţia este dată tabelar, datele de intrare trebuie să fie completate cu:

- tabelul nodurilor reţelei şi al valorilor funcţiei în aceste noduri.

În cazul *funcţiilor definite prin cod*, calculul primei derivate prin metoda diferenţelor finite progresive de ordinul 1 se poate face utilizând pseudocodul următor (care determină automat valoarea optimă a pasului de derivare):

```
funcţia derivare (x, h)          ; derivează funcţia f în punctul x, cu pasul iniţial h
    real x
    real h
    nrit = 10                    ; numărul maxim de iteraţii pentru
                                 ; determinarea pasului optim
    err = 1e-20                  ; eroarea relativă de rotunjire
    f0 = f(x)                    ; valoarea funcţiei în punctul de derivare
    e = err(|f0|+err)            ; eroarea absolută de rotunjire
    k = 0                        ; contor iteraţii
    repeta
        k = k+1
        f1 = f(x+h)
        f2 = f(x+2h)
        d2 = |f0-2f1+f2|/h^2     ; modulul dublei derivate
        dacă d2 = 0 atunci
            h0 = h
        altfel
            h0 = sqrt(2e/d2)     ; pasul optim
        r = h/h0                 ; rata de modificare a pasului
        h = h0                   ; actualizează pasul la valoarea optimă
    până când (k > nrit sau ( r < 2 şi r > 1/2))
    df = (f(x+h)-f0)/h           ; derivata progresivă de ordinul 1
    întoarce df
```

Aplicând funcţia `derivare` pentru valori negative ale parametrului h, valoarea întoarsă *df* va fi derivata regresivă de ordinul 1. Derivata centrată se poate calcula ca media aritmetică a valorilor regresivă şi progresivă.

În cazul funcţiilor tabelare, calculul derivatei admite următoarea reprezentare în pseudocod:

```
procedura derivtab (n, x0, h, f, df); calculează tabelul de derivare a funcției f,
                                    ; dat prin vectorii x și f, în n+1 puncte
    întreg n                        ; dimensiunea tabloului
    real x0                         ; nodul inițial
    real h                          ; pasul rețelei
    tablou real f(n)                ; tabloul valorilor funcției
    tablou real df(n)               ; tabloul valorilor derivatei (date de ieșire)
    df(0) = (-3f(0) + 4f(1) - f(2))/2h
    df(n) = (f(n-2) - 4f(n-1) + 3f(n))/2h
    pentru i = 1, n-1
        df(i) = (f(i+1) - f(i-1))/2h
retur
```

7.5 Chestiuni de studiat

1. Evaluarea numerică a primei derivate;
2. Analiza experimentală a erorii de derivare numerică;
3. Analiza derivării numerice de ordin superior;
4. Implementarea algoritmului;
5. Căutare de informaţii pe Internet.

7.6 Modul de lucru

Pentru desfăşurarea lucrării selectaţi opţiunea *Derivarea numerică a funcţiilor reale* din meniul principal de lucrări. Aceasta are ca urmare lansarea următorului meniu:

- Program demonstrativ
- Analiza erorilor
- Derivare de ordin superior

7.6.1 Evaluarea numerică a primei derivate

Se selectează din meniul principal opţiunea *Program demonstrativ*. Acest program calculează derivatele unor functii definite prin cod şi pune în evidenţă ordinul de mărime a erorilor ce apar, dependenţa acestora de modul de interpolare (progresivă, regresivă şi centrată), de pasul de derivare (valoare dată de utilizator şi valoare optimă, determinată de program), de funcţia de derivat şi de punctul în care se calculează derivata.

Utilizatorul alege:

- funcţia de derivat;
- punctul $x0$ în care se calculează derivata;
- pasul de derivare h.

Funcţia de derivat se poate alege din cele 6 funcţii afişate:

- x^2
- $\sqrt{x}$
- e^x
- $\ln(x)$
- $\sin(x)$
- $\mathrm{tg}(x)$

Programul afişează valoarea exactă a derivatei şi valorile aproximative calculate prin diferite metode şi erorile absolute (de rotunjire şi de trunchiere).

Se vor comenta rezultatele obţinute şi se va studia eficienţa algoritmului de optimizare a pasului.

7.6.2 Analiza experimentală a erorii de derivare numerică

Se selectează opţiunea *Analiza erorilor* din meniul principal.

Programul ilustrează dependenţa erorii de pasul de derivare folosit şi evidenţiază ponderile celor două tipuri de erori (de rotunjire şi de trunchiere).

Se folosesc succesiv formulele diferenţelor finite centrate de ordin 2, 4, respectiv 6, pentru calculul derivatei funcţiei sinus în punctul $\pi/4$. Se poate observa, astfel, şi dependenţa

ordinului de mărime al erorilor de ordinul polinomului de interpolare. Graficele sunt afişte în scară dublu-logaritmică, pentru a se putea vizualiza în amănunt tot intervalul "activ" al variaţiei erorilor.

După afişarea unui grafic, în consola Scilab se afişează eroarea minimă şi pasul pentru care este obţinută. Se va estima din grafic valorea pasului optim şi ordinul erorii de trunchiere în funcţie de pasul de derivare.

Se vor analiza şi comenta rezultatele obţinute.

7.6.3 Analiza derivării numerice de ordin superior

Se selectează opţiunea *Derivare de ordin superior*. Programul calculează şi reprezintă grafic derivatele succesive de ordin superior ale funcţiei exponenţiale $exp(x)$, definite pe intervalul $[Xmin, Xmax]$. Se afişează de fiecare dată şi graficul exact al derivatei respective, care este tot $exp(x)$.

Datele de intrare ale programului sunt:

- ordinul maxim al derivatei de reprezentat;
- X minim;
- X maxim;
- numărul de noduri.

Programul evaluează derivatele în punctele interioare ale reţelei cu metoda diferenţelor finite centrate de ordinul 2, iar pentru punctele periferice foloseşte metoda diferenţelor finite progresive respectiv regresive de ordinul 3.

Se recomandă analiza propagării erorilor la calculul până la ordinul 5, pe o reţea cu 100 de noduri ce discretizează intervalul [0, 10], precum şi intervalul [0, 1]. Se vor comenta comparativ rezultatele obţinute.

7.6.4 Implementarea algoritmului

Se va scrie, în limbajul de programare C, o funcţie care să permită calculul numeric al derivatei întâi, prin metoda diferenţelor finite centrate pentru

$$f(x) = \frac{x^2}{\sin(x)} \ln(x).$$

Folosind această funcţie, se va scrie apoi un program care să aibă ca date de intrare pasul de derivare şi punctul în care se calculează derivata şi care să afişeze valorile numerice ale derivatei.

7.6.5 Căutare de informaţii pe Internet

Căutaţi pe Internet informaţii şi coduri legate de derivarea numerică a funcţiilor. Cuvinte cheie recomandate: *numerical derivative*.

7.7 Exemple

7.7.1 Exemple rezolvate

1. Fie funcţia $y = f(x)$ definită tabelar:

x	1	2	3	4
y	2	1	3	5

Să se determine derivata funcţiei în nodurile reţelei de discretizare utilizând diferenţele finite astfel: formulele progresivă şi regresivă de ordinul întâi pentru primul şi respectiv ultimul nod, formula centrată de ordinul doi pentru nodurile interioare.

Rezolvare:

Reţeaua de discretizare este uniformă cu pasul $h = x_1 - x_0 = 2 - 1 = 1$.

Derivata progresivă (7.10) şi derivata regresivă (7.11) de ordinul întâi în primul nod şi respectiv ultimul nod sunt:

$$y_0' = \frac{y_1 - y_0}{h} = \frac{1-2}{1} = -1,$$

$$y_3' = \frac{y_3 - y_2}{h} = \frac{5-3}{1} = 2.$$

Derivatele centrate de ordinul doi (7.8) în nodurile interioare sunt:

$$y_1' = \frac{y_2 - y_0}{2h} = \frac{3-2}{2} = \frac{1}{2},$$

$$y_2' = \frac{y_3 - y_1}{2h} = \frac{5-1}{2} = 2.$$

2. Fie funcţia $y = f(x)$ definită tabelar:

x	1	3	5
y	4	7	10

Să se determine derivata funcţiei în nodurile reţelei de discretizare utilizând diferenţele finite astfel: formulele progresivă şi regresivă de ordinul întâi pentru primul şi respectiv ultimul nod, formula centrată de ordinul doi pentru nodul interior.

Rezolvare:

Reţeaua de discretizare este uniformă cu pasul $h = x_1 - x_0 = 3 - 1 = 2$.

Derivata progresivă (7.10) şi derivata regresivă (7.11) de ordinul întâi în primul nod şi respectiv ultimul nod sunt:

$$y_0' = \frac{y_1 - y_0}{h} = \frac{7-4}{2} = \frac{3}{2},$$
$$y_2' = \frac{y_2 - y_1}{h} = \frac{10-7}{2} = \frac{3}{2}.$$

Derivata centrată de ordinul doi (7.8) în nodul interior este:

$$y_1' = \frac{y_2 - y_0}{2h} = \frac{10-4}{4} = \frac{3}{2}.$$

Cele trei derivate sunt egale deoarece punctele din tabel sunt plasate pe o dreaptă.

3. Fie funcţia $y = f(x)$ definită tabelar:

x	1	3	5	7	9
y	3	1	2	4	1

Să se determine derivata funcţiei în nodurile reţelei de discretizare utilizând diferenţele finite astfel: formulele progresivă şi regresivă de ordinul doi pentru primul şi respectiv ultimul nod, formula centrată de ordinul doi pentru nodurile interioare.

Rezolvare:

Reţeaua de discretizare este uniformă cu pasul $h = x_1 - x_0 = 3 - 1 = 2$.

Derivata progresivă (7.7) şi derivata regresivă (7.9) de ordinul doi în primul nod şi respectiv ultimul nod sunt:

$$y_0' = \frac{-3y_0 + 4y_1 - y_2}{2h} = \frac{-9+4-2}{4} = -\frac{7}{4},$$
$$y_4' = \frac{y_2 - 4y_3 + 3y_4}{2h} = \frac{2-16+3}{4} = -\frac{11}{4}.$$

Derivatele centrate de ordinul doi (7.8) în nodurile interioare sunt:

$$y_1' = \frac{y_2 - y_0}{2h} = \frac{2-3}{4} = -\frac{1}{4},$$
$$y_2' = \frac{y_3 - y_1}{2h} = \frac{4-1}{4} = \frac{3}{4},$$
$$y_3' = \frac{y_4 - y_2}{2h} = \frac{1-2}{4} = -\frac{1}{4}.$$

4. Să se estimeze ordinul de mărime al pasului de derivare optim pentru evaluarea numerică a derivatei funcţiei $sin\, x$ cu o formulă de derivare progresivă de ordinul 1.

   Rezolvare:

   Formula de calcul a pasului optim este:

   $$h_{\text{optim}} = 2\sqrt{\frac{\varepsilon_0 M_0}{M_2}}.$$

   În cazul funcţiei sinus, $M_0 = M_2 = 1$. Considerând $\varepsilon_0 = 10^{-16}$, rezultă un pas de derivare optim de ordinul $h_{\text{optim}} = 2 \cdot 10^{-8}$.

7.7.2 Exemple propuse

1. Fie funcţia $y = f(x)$ definită tabelar:

x	1	3	5	7
y	1	2	4	1

   Să se determine derivata funcţiei în nodurile reţelei de discretizare utilizând diferenţele finite astfel: formulele progresivă şi regresivă de ordinul întâi pentru primul şi respectiv ultimul nod, formula centrată de ordinul doi pentru nodurile interioare.

2. Fie funcţia $y = f(x)$ definită tabelar:

x	1	2	3	4	5
y	1	4	2	3	5

   Să se determine derivata funcţiei în nodurile reţelei de discretizare utilizând diferenţele finite astfel: formulele progresivă şi regresivă de ordinul doi pentru primul şi respectiv ultimul nod, formula centrată de ordinul doi pentru nodurile interioare.

7.8 Probleme şi întrebări

1. Deduceţi formulele aproximative ale derivatelor centrate până la ordinul 4.

2. Determinaţi expresia Laplaceanului funcţiei $f(x, y)$, folosind diferenţele finite centrate de ordinul 2. Calculaţi valoarea Laplaceanului în punctul $x = 1, y = 1$, pentru funcţia:

   $$f(x, y) = x^3 y^3$$

   folosind formula obţinută şi evaluaţi eroarea.

3. Determinaţi expresiile aproximative centrate de ordinul 2 ale derivatelor întâi şi a doua, în cazul în care pasul de derivare nu este constant. Se vor folosi notaţiile:

$$\begin{aligned} x(i) - x(i-1) &= h \\ x(i+k) - x(i) &= kh \end{aligned}$$

   în care se presupun cunoscute: $h, k, f(x(i-1)), f(x(i)), f(x(i+k))$.

4. Deduceţi ordinul de mărime al erorii maxime în cazul aproximării centrate, progresive şi regresive de ordin n.

5. Evaluaţi ordinul erorii de rotunjire la metodele diferenţelor finite de ordinul 1 şi ordinul 2. Analizaţi dependenţa acestor erori de pasul h.

6. Argumentaţi algoritmul pentru determinarea pasului optim, folosit în funcţia `derivare`, prezentată în pseudocod.

7. Generaţi un algoritm pentru determinarea pasului optim la diferenţe centrate.

8. Generaţi un algoritm pentru derivarea funcţiilor definite tabelar pe reţele neuniforme.

Lucrarea 8

Integrarea numerică a funcţiilor reale

8.1 Caracterizarea metodelor de integrare numerică

Integrarea numerică se aplică funcţiilor care nu pot fi integrate analitic (sau ar necesita calcule complicate) sau în cazul funcţiilor date tabelar, de exemplu cele rezultate experimental.

În general, metodele de integrare numerică se bazează pe aproximarea funcţiei de integrat prin functii mai simple, a căror integrală se poate evalua uşor. O metodă de aproximare frecvent utilizată în cadrul integralelor este *metoda interpolării polinomiale pe porţiuni*.

Spre deosebire de derivarea numerică, integrarea este o operaţie relativ stabilă numeric, dar mai lentă, din cauza numărului sporit de calcule care trebuie efectuate.

8.2 Principiul metodei

Se consideră funcţia reală $f : [a, b] \to \mathbb{R}$, definită prin cod sau tabelar, într-o reţea de $n + 1$ noduri: $x_0, x_1, \ldots x_n$. Se formulează problema calculului integralei:

$$I = \int_a^b f(x)\,\mathrm{d}x. \tag{8.1}$$

Metoda trapezelor

Este una dintre cele mai simple metode de integrare numerică, cu rezultate bune în aplicaţiile inginereşti.

Pe o reţea de $n + 1$ noduri echidistante, pasul reţelei are valoarea

$$h = \frac{x_n - x_0}{n}. \tag{8.2}$$

Metoda se bazează pe aproximarea variaţiei funcţiei de integrat între două noduri succesive ale reţelei printr-un polinom de gradul întâi (o dreaptă). Aceasta echivalează cu aproximarea ariei subîntinse de graficul funcţiei între două noduri succesive (i şi $i+1$), cu aria trapezului "sprijinit" de axa Ox, determinat de abscisele x_i, x_{i+1} şi ordonatele $f(x_i)$ si $f(x_{i+1})$. Prin însumarea ariilor tuturor trapezelor de acest fel care se formează între punctele a şi b, se obţine valoarea aproximativă a integralei definite a lui f, între aceste puncte.

Aria unui trapez este

$$I_i = \frac{h(f(x_i) + f(x_{i+1}))}{2}. \tag{8.3}$$

Rezultă, prin însumare pentru $i = 0, \ldots, n-1$, formula de integrare prin metoda trapezelor:

$$I = \int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{2}(f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n)). \tag{8.4}$$

Metoda Simpson 1/3

Se aproximează variaţia funcţiei de integrat între trei noduri succesive $(i-1, i, i+1)$, printr-un polinom de interpolare de gradul doi: $P(x) = a_0 + a_1 x + a_2 x^2$ (o parabolă).

Presupunănd aceeaşi reţea de $n+1$ noduri echidistante, dar cu n par, şi considerând originea în punctul x_i, se poate scrie:

$$\begin{aligned} f(x_{i-1}) &= P(-h) = a_0 - a_1 h + a_2 h^2, \\ f(x_i) &= P(0) = a_0, \\ f(x_{i+1} &= P(h) = a_0 + a_1 h + a_2 h^2. \end{aligned} \tag{8.5}$$

Aşadar, se aproximează aria subîntinsă de funcţia de integrat între punctele x_{i-1} şi x_{i+1} cu aria subîntinsă de parabola P dintre punctele $-h$ şi h:

$$I_i = \int_{-h}^{h} P(x)\,\mathrm{d}x = \int_{-h}^{h} (a_0 + a_1 x + a_2 x^2)\,\mathrm{d}x = 2ha_0 + 2\frac{h^3}{3}a_2. \tag{8.6}$$

Din condiţiile de interpolare se calculează:

$$a_0 = f(x_i); \tag{8.7}$$

$$a_2 = \frac{f(x_{i-1}) - 2f(x_i) + f(x_{i+1})}{2h^2} \tag{8.8}$$

necesare în evaluarea integralei

$$I_i = \frac{h(f(x_{i-1}) + 4f(x_i) + f(x_{i+1}))}{3}. \tag{8.9}$$

Însumând toate ariile I_i, pentru $i = 1, 3, 5, \ldots, n-1$, rezultă

$$I = \int_a^b f(x)\,\mathrm{d}x \approx \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + f(x_n)), \tag{8.10}$$

cunoscută sub numele de *Formula lui Simpson 1/3*.

Integrarea Romberg

Valoarea numerică a integralei depinde de pasul de integrare h, eroarea de trunchiere scăzând cu micşorarea pasului h. Se notează cu I_0 valoarea exactă a integralei şi cu $I = I(h)$ valoarea ei numerică. Deoarece funcţia $I(h)$ este pară (trapezele obţinute pornind din a cu pasul h sunt aceleaşi cu cele obţinute pornind din b cu pasul $-h$), prin dezvoltarea ei in serie Taylor în jurul lui 0 se obţine, în cazul metodei trapezelor:

$$I = I_0 + a_0h^2 + a_1h^4 + a_2h^6 + \ldots \tag{8.11}$$

Reţinând din această serie numai doi termeni, se obţin pentru două valori diferite h_1 si h_2 ale pasului de integrare:

$$\begin{aligned} I_1 &= I_0 + a_0h_1^2; \\ I_2 &= I_0 + a_0h_2^2. \end{aligned}$$

Din cele două ecuaţii rezultă următoarea aproximare pentru valoarea exactă a integralei:

$$I_0 = \frac{h_2^2I_1 - h_1^2I_2}{h_2^2 - h_1^2}, \tag{8.12}$$

cunoscută sub numele de *formula de integrare Romberg*.

Reducând succesiv valoarea lui h: $h_1 > h_2 > h_3 > \ldots > h_k \ldots$, se obţin valorile integralei numerice:

$$I_{0k} = \frac{h_{k+1}^2I_k - h_k^2I_{k+1}}{h_{k+1}^2 - h_k^2}, \quad k = 1, 2, \ldots, \tag{8.13}$$

care au precizie din ce în ce mai bună.

Formula lui Romberg se simplifică, dacă la fiecare iteraţie se înjumătăţeşte pasul de integrare ($h_{k+1} = h_k/2$):

$$I_{0k} = \frac{4I_{k+1} - I_k}{3}. \tag{8.14}$$

Valorile numerice ale integralelor $I_{01}, I_{02}, I_{03}, \ldots$, calculate pe reţele tot mai fine, pot fi folosite pentru calculul unei valori numerice de precizie ridicată, prin aplicarea recursivă a formulei Romberg:

- Nr. intervale: n 2n 4n ...
- Met. trapezelor: I_{01} I_{02} I_{03} ...

- Romberg ord. 1: $I_{11} = \frac{4I_{02}-I_{01}}{3}$ $\quad I_{12} = \frac{4I_{03}-I_{02}}{3}$ ...
- Romberg ord. 2: $I_{21} = \frac{16I_{12}-I_{11}}{15}$ ...

În general, formula lui Romberg de ordinul k pentru înjumătăţirea pasului este

$$I_{mk} = \frac{4^m I_{m-1,k+1} - I_{m-1,k}}{4^m - 1}, \tag{8.15}$$

cu $k = 1, 2, \ldots$ şi $m = 1, 2, \ldots$. Iteraţiile se pot opri pentru acel ordin m, la care eroarea relativă

$$\left| \frac{I_{mk} - I_{m-1,k}}{I_{mk}} \right|$$

este mai mică decât o valoare impusă.

8.3 Pseudocodul algoritmilor

În cazul funcţiilor definite prin cod, **metoda trapezelor** are următorul pseudocod:

```
funcţia trapez (a, b, n)                ; calculează integrala definită a
                                        ; funcţiei f in intervalul [a, b],
                                        ; discretizat in n subintervale
                                        ; egale
    real a, b
    întreg n
    h = (b − a)/n
    rezultat = 0
    pentru k = 1, n − 1
            rezultat = rezultat+f(a + kh)
    rezultat=(2· rezultat +f(a) + f(b)) ∗ h/2
întoarce rezultat
```

Pseudocodul **metodei Simpson**, pentru funcţiile definite tabelar, are forma:

```
citeşte a, b                            ; limitele de integrare
citeşte n                               ; trebuie să fie par
pentru k = 0, n                         ; valorile funcţiei în nodurile reţelei
        citeşte y_k
h = (b − a)/n
rezultat = 0
```

pentru $k = 1, n-1, 2$
    rezultat = rezultat $+y_{k-1} + 4y_k + y_{k+1}$
rezultat = rezultat $\cdot h/3$
scrie rezultat

Pseudocodul **metodei Romberg** este:

citeşte a, b
citeşte n ; numărul iniţial al subintervalelor de integrare
citeşte eps ; diferenţa maximă dintre rezultatele
; iteraţiilor succesive
rezultat_nou = trapez (a, b, n)
repetă
    rezultat_vechi = rezultat_nou
    $n = 2n$
    rezultat_nou = trapez (a, b, n)
    rezultat = $(4\cdot$ rezultat_nou - rezultat_vechi$)/3$
până când (|rezultat - rezultat_vechi|) <eps
scrie rezultat

8.4 Analiza algoritmilor

Analiza erorilor

În cazul integrării numerice, se pot defini două tipuri de erori:

- *eroarea locală*, care apare prin aproximarea ariei subîntinse pe un interval elementar (de lungime h, respectiv $2h$) cu aria elementară specifică metodei: aria trapezului (la metoda trapezelor), respectiv aria subîntinsă de parabolă (la metoda Simpson).
- *eroarea globală*, care apare prin aproximarea integralei cu suma ariilor elementare.

La metoda trapezelor, eroarea locală are ordinul $O(h^3)$, iar în cazul metodei Simpson ea are ordinul $O(h^5)$. Eroarea globală este mai mare decât eroarea locală, având ordinul $O(h^2)$ respectiv $O(h^4)$.

Efort de calcul

Timpul de calcul depinde de numărul de noduri din reţeaua de discretizare, având ordinul liniar $O(n)$, atât în cazul metodei trapezelor cât şi în cazul metodei Simpson.

Metoda trapezelor e mai flexibilă, deoarece foloseşte o reţea cu număr oarecare de noduri, spre deosebire de metoda Simpson, care necesită un număr *impar* de noduri. Aceasta din urmă este însă mai precisă.

Metoda Romberg este cea mai exactă, dar şi cea mai lentă, necesitând un număr relativ mic de iteraţii, însă la fiecare iteraţie reţeaua de noduri se îndeseşte de două ori.

8.5 Chestiuni de studiat

1. Calculul numeric al integralei definite, pentru câteva funcţii elementare;
2. Analiza experimentală a erorii la integrarea numerică;
3. Implementarea algoritmului;
4. Căutarea de informaţii pe Internet.

8.6 Modul de lucru

Pentru desfăşurarea lucrării se selectează lucrarea *Integrarea numerică a funcţiilor reale* din meniul general de lucrări.

Aceasta are ca efect afişarea meniului:

1. Program demonstrativ
2. Analiza erorilor

8.6.1 Calculul integralei unor funcţii elementare

În urma selectării opţiunii *Program demonstrativ*, se poate alege una din funcţiile:

- x^2
- $\sqrt{x}$
- e^x
- $\ln(x)$
- $\sin(x)$
- $\mathrm{tg}\,(x)$

După alegerea unei funcţii, utilizatorul trebuie să introducă:

- limitele inferioară şi superioară ale intervalului de integrare;
- numărul subintervalelor în care se împarte intervalul de integrare.

După efectuarea calculelor, se trasează graficul funcţiei pe domeniul de integrare, punându-se în evidenţă trapezele corespunzătoare acestui domeniu.

În consola Scilab se afişează rezultatele:

- valoarea exactă a integralei definite;
- valoarea aproximativă şi modulul erorii absolute.

Se vor comenta rezultatele obţinute.

8.6.2 Analiza erorii la integrarea numerică

Se selectează *Analiza erorilor* din meniul principal al lucrării. Programul ilustrează grafic dependenţa modulului erorilor absolute de pasul de integrare h. Se folosesc, pe rând, metodele:

- trapezelor
- Simpson
- Romberg cu 2 iteraţii
- Romberg cu 3 iteraţii

pentru calculul integralei funcţiei sinus între punctele 0 şi π.

Se vor analiza şi comenta rezultatele obţinute. Se va estima ordinul erorii de trunchiere pentru toate cele patru cazuri.

8.6.3 Implementarea algoritmului

Se va scrie în limbajul de programare C, un program care să permita calculul integralei definite a funcţiei $f(x) = x^2 \sin(x)$, folosind metoda Simpson 1/3. Datele de intrare ale programului vor fi cele două limite de integrare.

8.6.4 Căutare de informaţii pe Internet

Se vor căuta pe Internet informaţii (coduri) legate de integrarea numerică a funcţiilor. Cuvinte cheie recomandate: *Numerical Integration*, *Numerical Quadrature*, *Gauss Quadrature*.

8.7 Exemple

8.7.1 Exemple rezolvate

1. Fie funcţia $f : [1, 5] \to \mathbb{R}$, $y = f(x)$, definită tabelar:

x	1	2	4	5
y	2	1	3	5

Se cer:

(a) Reprezentaţi grafic interpolarea liniară pe porţiuni a acestor date.

(b) Câte trapeze se consideră în urma aplicării metodei trapezelor pentru determinarea integralei funcţiei $f(x)$ pe intervalul $[1, 5]$? Indicaţi grafic aceste trapeze pe figura de la punctul precedent.

(c) Reţeaua de discretizare a domeniului de definiţie a funcţiei $f(x)$ este neuniformă?

(d) Să se determine integrala $I = \int_1^5 f(x)\,\mathrm{d}x$ prin metoda trapezelor.

(e) Se poate aplica metoda Simpson 1/3? Dacă răspunsul este afirmativ, să se determine $I = \int_1^5 f(x)\,\mathrm{d}x$ prin această metodă.

Rezolvare:

(a) În figura 9 este reprezentată grafic interpolarea liniară pe porţiuni a funcţiei $y = f(x)$.

(b) Funcţia $f(x)$ este cunoscută în 4 puncte (3 segmente). Astfel, numărul de trapeze considerat este 3. Cele trei trapeze sunt haşurate diferit în figura 9.

(c) O reţea de discretizare x_k, $k = 0, n$ este uniformă dacă pasul de discretizare corespunzător unui segment, $h_k = x_k - x_{k-1}$, $k = 1, n$, este constant, indiferent de segmentul k considerat. În cazul acestui exemplu, avem:

$$h_1 = x_1 - x_0 = 2 - 1 = 1,$$

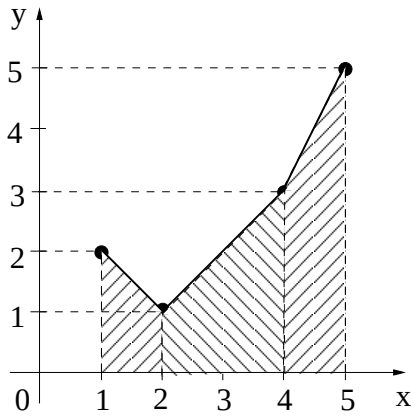

Fig. 9. Interpolarea liniară pe porţiuni a funcţiei f

$$h_2 = x_2 - x_1 = 4 - 2 = 2,$$
$$h_3 = x_3 - x_2 = 5 - 4 = 1.$$

Deoarece pasul nu este constant pe întreg domeniul de definiţie, reţeaua de discretizare este neuniformă.

(d) Aria primului trapez (8.3) este:

$$I_0 = \frac{y_0 + y_1}{2}(x_1 - x_0) = \frac{2+1}{2}(2-1) = \frac{3}{2}.$$

În mod similar se determină ariile celorlalte două trapeze:

$$I_1 = \frac{y_1 + y_2}{2}(x_2 - x_1) = \frac{1+3}{2}(4-2) = 4,$$
$$I_2 = \frac{y_2 + y_3}{2}(x_3 - x_2) = \frac{3+5}{2}(5-4) = 4.$$

Integrala prin metoda trapezelor se obţine prin însumarea ariilor trapezelor:

$$I = \int_1^5 f(x)\,\mathrm{d}x = I_0 + I_1 + I_2 = \frac{3}{2} + 4 + 4 = \frac{19}{2}.$$

(e) Metoda Simpson 1/3 se poate aplica numai când numărul de noduri este impar. Pentru cazul dat, numărul de noduri este 4, şi în consecinţă, metoda Simpson 1/3 nu se poate aplica.

2. Fie funcţia $f : [1, 5] \to \mathbb{R}$, $y = f(x)$, definită tabelar:

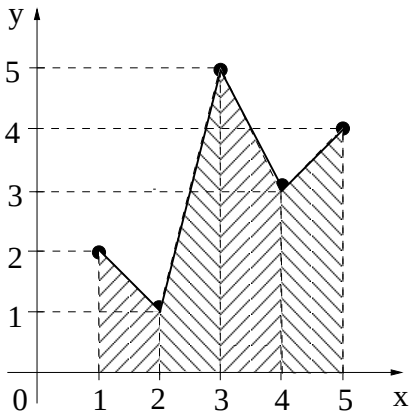

Fig. 10. Interpolarea liniară pe porţiuni a funcţiei f

x	1	2	3	4	5
y	2	1	5	3	4

Se cer:

(a) Reprezentaţi grafic interpolarea liniară pe porţiuni a acestor date.

(b) Câte trapeze se consideră în urma aplicării metodei trapezelor pentru determinarea integralei funcţiei $f(x)$ pe intervalul $[1, 5]$? Indicaţi grafic aceste trapeze pe figura de la punctul precedent.

(c) Reţeaua de discretizare a domeniului de definiţie a funcţiei $f(x)$ este neuniformă?

(d) Să se determine integrala $I = \int_1^5 f(x)\,\mathrm{d}x$ prin metoda trapezelor.

(e) Se poate aplica metoda Simpson 1/3? Dacă răspunsul este afirmativ, să se determine $I = \int_1^5 f(x)\,\mathrm{d}x$ prin această metodă.

Rezolvare:

(a) În figura 10 este reprezentată grafic interpolarea liniară pe porţiuni a funcţiei $y = f(x)$.

(b) Deoarece gridul de discretizare al domeniului de definiţie conţine 5 puncte, determinând o împărţire în 4 segmente, numărul de trapeze considerat este 4. Trapezele sunt haşurate diferit în figura 10.

(c) Se observă că pasul de discretizare corespunzător fiecărui segment este constant pe întreg domeniul de definiţie, $h = 1$. Din acest motiv, reţeaua este uniformă.

(d) Deoarece pasul este constant, se poate aplica direct formula simplificată pentru reţea uniformă a metodei trapezelor (8.4):

$$I = \int_1^5 f(x)\,dx = \frac{h}{2}(y_0+2y_1+2y_2+2y_3+y_4) = \frac{1}{2}(2+2\cdot 1+2\cdot 5+2\cdot 3+4) = 12.$$

Alternativ, se pot însuma cele 4 arii ale trapezelor pentru determinarea integralei numerice.

(e) Metoda Simpson 1/3 se poate aplica deoarece numărul de noduri este impar (5).

Deoarece reţeaua de discretizare este uniformă, se aplică formula (8.10):

$$I = \int_1^5 f(x)\,dx = \frac{h}{3}(y_0+4y_1+2y_2+4y_3+y_4) = \frac{1}{3}(2+4\cdot 1+2\cdot 5+4\cdot 3+4) = \frac{32}{3}.$$

3. Fie funcţia $f : [1, 11] \to \mathbb{R}$, $y = f(x)$, definită tabelar:

x	1	3	5	6	7	9	11
y	2	3	4	1	5	2	3

Se cer:

(a) Câte trapeze se consideră în urma aplicării metodei trapezelor pentru determinarea integralei funcţiei $f(x)$ pe intervalul $[1, 11]$?

(b) Reţeaua de discretizare a domeniului de definiţie a funcţiei $f(x)$ este neuniformă?

(c) Să se determine integrala $I = \int_1^1 1f(x)\,\mathrm{d}x$ prin metoda trapezelor.

(d) Se poate aplica metoda Simpson 1/3? Dacă răspunsul este afirmativ, să se determine $I = \int_1^1 1f(x)\,\mathrm{d}x$ prin această metodă

Rezolvare:

(a) Funcţia $f(x)$ este cunoscută în 7 puncte. Acestea determină 6 segmente, care discretizează domeniul de definiţie $[1, 11]$. Numărul de trapeze considerat este 6, egal cu numărul de segmente.

(b) Se observă că pasul de discretizare nu este constant pe fiecare segment:

$$h_1 = x_1 - x_0 = 3 - 1 = 2,\ h_3 = x_3 - x_2 = 6 - 5 = 1.$$

Astfel, reţeaua de discretizare este neuniformă.

(c) Integrala prin metoda trapezelor, pentru o reţea neuniformă, se determină prin însumarea ariilor trapezelor:

$$I = \int_1^{11} f(x)\,dx = \frac{y_0+y_1}{2}(x_1-x_0) + \frac{y_1+y_2}{2}(x_2-x_1) + \frac{y_2+y_3}{2}(x_3-x_2) +$$

$$+\frac{y_3+y_4}{2}(x_4-x_3) + \frac{y_4+y_5}{2}(x_5-x_4) + \frac{y_5+y_6}{2}(x_6-x_5) =$$

$$= \frac{2+3}{2}(3-1) + \frac{3+4}{2}(5-3) + \frac{4+1}{2}(6-5) + \frac{1+5}{2}(7-6) +$$

$$+\frac{5+2}{2}(9-7) + \frac{2+3}{2}(11-9) = 5 + 7 + \frac{5}{2} + 3 + 7 + 5 = \frac{59}{2}.$$

(d) Metoda Simpson 1/3 se poate aplica deoarece numărul de noduri este impar (7).

Reţeaua este neuniformă, astfel integrala prin metoda Simpson 1/3 se determină prin însumarea ariilor suprafeţelor subîntinse de parabolele care trec prin trei noduri consecutive (8.9). În total, sunt trei parabole: $[x_0, x_1, x_2]$ (pasul este 2), $[x_2, x_3, x_4]$ (pasul este 1), $[x_4, x_5, x_6]$ (pasul este 2).

$$I_1 = \frac{h_1}{3}(y_0 + 4y_1 + y_2) = \frac{2}{3}(2 + 4\cdot 3 + 4) = 12,$$

$$I_3 = \frac{h_3}{3}(y_2 + 4y_3 + y_4) = \frac{1}{3}(4 + 4\cdot 1 + 5) = \frac{13}{3},$$

$$I_5 = \frac{h_5}{3}(y_4 + 4y_5 + y_6) = \frac{2}{3}(5 + 4\cdot 2 + 3) = \frac{32}{3}.$$

$$I = \int_1^{11} f(x)\,dx = I_1 + I_3 + I_5 = 12 + \frac{13}{3} + \frac{32}{3} = 27.$$

8.7.2 Exemple propuse

1. Fie funcţia $f : [1, 6] \to \mathbb{R}$, $y = f(x)$, definită tabelar:

x	1	2	4	6
y	1	3	2	4

Se cer:

(a) Reprezentaţi grafic interpolarea liniară pe porţiuni a acestor date.

(b) Câte trapeze se consideră în urma aplicării metodei trapezelor pentru determinarea integralei funcţiei $f(x)$ pe intervalul $[1, 6]$? Indicaţi grafic aceste trapeze pe figura de la punctul precedent.

(c) Reţeaua de discretizare a domeniului de definiţie a funcţiei $f(x)$ este neuniformă?

(d) Să se determine integrala $I = \int_1^6 f(x)\,\mathrm{d}x$ prin metoda trapezelor.

(e) Se poate aplica metoda Simpson 1/3? Dacă răspunsul este afirmativ, să se determine $I = \int_1^6 f(x)\,\mathrm{d}x$ prin această metodă.

2. Fie funcţia $f : [1, 13] \to \mathbb{R}$, $y = f(x)$, definită tabelar:

x	1	3	5	7	9	11	13
y	1	4	2	3	1	2	5

Se cer:

(a) Reprezentaţi grafic interpolarea liniară pe porţiuni a acestor date.

(b) Câte trapeze se consideră în urma aplicării metodei trapezelor pentru determinarea integralei funcţiei $f(x)$ pe intervalul $[1, 13]$? Indicaţi grafic aceste trapeze pe figura de la punctul precedent.

(c) Reţeaua de discretizare a domeniului de definiţie a funcţiei $f(x)$ este neuniformă?

(d) Să se determine integrala $I = \int_1^{13} f(x)\,\mathrm{d}x$ prin metoda trapezelor.

(e) Se poate aplica metoda Simpson 1/3? Dacă răspunsul este afirmativ, să se determine $I = \int_1^{13} f(x)\,\mathrm{d}x$ prin această metodă.

3. Fie funcţia $f : [1, 7] \to \mathbb{R}$, $y = f(x)$, definită tabelar:

x	1	2	3	5	7
y	3	1	2	3	4

Se cer:

(a) Reprezentaţi grafic interpolarea liniară pe porţiuni a acestor date.

(b) Câte trapeze se consideră în urma aplicării metodei trapezelor pentru determinarea integralei funcţiei $f(x)$ pe intervalul $[1, 7]$? Indicaţi grafic aceste trapeze pe figura de la punctul precedent.

(c) Reţeaua de discretizare a domeniului de definiţie a funcţiei $f(x)$ este neuniformă?

(d) Să se determine integrala $I = \int_1^7 f(x)\,\mathrm{d}x$ prin metoda trapezelor.

(e) Se poate aplica metoda Simpson 1/3? Dacă răspunsul este afirmativ, să se determine $I = \int_1^7 f(x)\,\mathrm{d}x$ prin această metodă.

8.8 Probleme şi întrebări

1. Generaţi algoritmul *metodei dreptunghiurilor*, bazat pe aproximarea funcţiei de integrat printr-o funcţie "scară". Comparaţi precizia acestei metode cu metoda trapezelor. Analizaţi cazul unei funcţii la care derivata a doua îşi păstrează semnul pe intervalul de integrare.

2. Generaţi algoritmul metodei trapezelor pentru o funcţie definită tabelar, pe o reţea cu pas variabil. Aplicati algoritmul pentru datele de mai jos:

x	1	7	9
y	5	13	15

3. Demonstraţi expresia ordinului erorii locale la metoda trapezelor.

4. Calculaţi integrala următoare, folosind metoda trapezelor şi $h_x = h_y = 1$:

$$I = \int_0^3 \int_0^2 xy\,\mathrm{d}x\mathrm{d}y.$$

Comparaţi cu rezultatul exact.

5. Obţineţi formula Simpson 3/8, folosind un polinom de interpolare de ordinul 3. Ce restricţie trebuie să îndeplinească numărul de subintervale al reţelei de discretizare?

6. Generaţi un algoritm de integrare numerică, bazat pe interpolarea "spline" a funcţiei de integrat.

7. Generaţi un algoritm de integrare numerică, bazat pe pseudocodul metodei Simpson, dedicat funcţiilor definite prin cod. Modificaţi acest algoritm astfel încât să fie efectuat un număr minim de evaluări.

8. Generaţi un algoritm pentru calculul integralei duble: $\int\int f(x,y)\,\mathrm{d}x\mathrm{d}y$ pe un dreptunghi discretizat cu o reţea cu doi paşi constanţi h_x şi respectiv h_y.

9. Generaţi un algoritm (metoda de integrare Gauss), bazat pe aproximarea

$$\int_{-1}^{1} f(x)\,\mathrm{d}x = \omega_0 f(x_0) + \omega_1 f(x_1) + \omega_2 f(x_2).$$

   Alegeţi ponderile ω_0, ω_1, ω_2 şi nodurile x_0, x_1, x_2 în mod optim, astfel încât eroarea să fie minimă. Demonstraţi că pentru: $x_1 = 0$, $x_2 = -x_0 = 3/5$ şi $\omega_1 = 8/9$, $\omega_0 = \omega_2 = 5/9$ formula anterioară este exactă dacă $f(x)$ este un polinom de gradul 5.

10. Generaţi un algoritm de integrare numerică prin metoda trapezelor, la care reţeaua de noduri este adaptată funcţiei. Pornind de la o reţea cu pas constant se vor îmbunătăţi succesiv acele subintervale pentru care eroarea locală de trunchiere

$$e = \frac{h^2}{12}\frac{\partial^2 f(z)}{\partial z^2}, \tag{8.16}$$

    cu z în subinterval, este maximă. Procesul de rafinare a reţelei va continua până când eroarea totală de trunchiere scade sub o limită impusă.

11. Calculaţi numeric integrala definită a unor funcţii date prin expresiile lor analitice în mediul MATLAB/SCILAB.

12. Identificaţi în biblioteca matematică descrisă în [2] funcţii avansate pentru integrarea numerică. Ce aduc nou aceste funcţii faţă de cele analizate în lucrare?

Thomas Simpson (1710, Anglia - 1761, Anglia)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Simpson.html

Lewis Fry Richardson (1881, Anglia - 1953, Scoţia)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Richardson.html

Lucrarea 9

Rezolvarea numerică prin metode iterative a ecuaţiilor neliniare

9.1 Caracterizarea lucrării

Rezolvarea numerică a unei ecuaţii algebrice neliniare sau transcendente $f(x) = 0$ constă în construirea unui şir $x_0, x_1, x_2, \ldots, x_k, \ldots$ convergent către solutia x a ecuaţiei considerate. Se adoptă drept soluţie numerică un termen x_k al şirului suficient de apropiat de limita x.

Modul în care este generat şirul determină calităţile metodei (viteza de convergenţă, eroare numerică, efort de calcul, etc.). De obicei şirul este definit recurent, iar procesul iterativ se opreşte atunci când distanţa dintre doi termeni succesivi devine suficient de mică sau $|f(x)|$ este neglijabil.

9.2 Principiul lucrării

Se consideră o ecuaţie neliniară:

$$f(x) = 0, \tag{9.1}$$

în care f este o funcţie continuă definită pe intervalul $[a, b]$ cu valori în $\mathbb{R}$, iar x este soluţia exactă a ecuaţiei (presupunând că există un x astfel încât $a < x < b$ şi acesta este unic).

Pentru determinarea numerică a soluţiei x se construieşte un şir $x_0, x_1, \ldots, x_k, \ldots$ convergent către soluţia exactă x a ecuaţiei. Termenul x_k al şirului se determină în mod recurent:

$$x_k = g(x_{k-1}), \tag{9.2}$$

în funcţie de termenul precedent. Iteraţiile se opresc atunci când distanţa dintre doi termeni succesivi $|x_k - x_{k-1}|$ este mai mică decât o valoare impusă ε. Se presupune că valoarea x_k astfel obţinută este o aproximare suficient de bună a soluţiei exacte şi se adoptă ca soluţie numerică. În consecinţă, metoda iterativă este caracterizată prin *funcţia de iteraţie* g, care trebuie aleasă astfel încât punctul ei fix $x = g(x)$ să coincidă cu soluţia ecuaţiei $f(x) = 0$.

Metoda bisecţiei

Una din cele mai simple metode iterative pentru rezolvarea unei ecuaţii neliniare, dar care nu se bazează pe construcţia unei funcţii de iteraţie, este metoda bisecţiei, care se bazează pe observaţia că $f(x)$ are semne diferite la capetele intervalului $[a, b]$, în interiorul căruia se determină soluţia, deci $f(a)f(b) < 0$. Considerând $x_m = (a + b)/2$, se va înjumătăţi intervalul $[a, b]$ şi se va selecta din cele două subintervale acela care conţine soluţia. Pentru aceasta se determină semnul valorii $f(x_m)$ şi se modifică $a = x_m$ sau $b = x_m$ în funcţie de acest semn, astfel încât $f(a)f(b) < 0$. Şirul valorilor succesive ale lui x_m este convergent către soluţia exactă x. Procedura iterativă de înjumătăţire se opreşte atunci când lungimea intervalului $|b - a|$ devine mai mică decât eroarea impusă solutiei.

Metoda iteraţiei simple

O altă variantă a metodei iterative este cunoscută sub numele de metoda iteraţiei simple. Ea constă în transformarea ecuaţiei $f(x) = 0$ într-o ecuaţie de forma $x = g(x)$, de exemplu, prin artificiul $x = x + cf(x)$, în care $g(x) = x + cf(x)$. Aparent, valoarea constantei c cu care se multiplică ecuaţia $f(x) = 0$ nu este importantă, totuşi se va constata că ea infuenţează puternic convergenţa şirului de iteratii.

În consecinţă, în metoda iteraţiei simple, şirul soluţiilor numerice este generat de relaţia:

$$x_k = x_{k-1} + cf(x_{k-1}) \tag{9.3}$$

în care $k = 1,2,\ldots,n$.

Se porneşte de la o iniţializare arbitrară $x_0 \in [a, b]$ şi se calculează succesiv $x_1, x_2, \ldots, x_n$ până când $|x_n - x_{n-1}| < eps$. O condiţie suficientă pentru convergenţa şirului (16.2) este ca g să fie o contracţie, respectiv:

$$|g(x_1) - g(x_2)| \leq L|x_1 - x_2| \tag{9.4}$$

cu L strict subunitar, pentru orice $x_1, x_2 \in [a, b]$. Dacă f este derivabilă, această condiţie este satisfăcută atunci când $|g'| < 1$, deci:

$$|1 + cf'(x)| < 1, \quad x \in [a, b]. \tag{9.5}$$

Condiţia (9.5) evidenţiază importanţa constantei c, care trebuie să aibă semn opus derivatei funcţiei f şi trebuie aleasă astfel încât $|1 + cf'(x)|$ să fie subunitar. Cu cât $|g'| = |1 + cf'(x)|$ este mai mic, cu atât şirul iterativ este mai rapid convergent.

Metoda Newton (a tangentelor)

Una din metodele iterative cele mai eficiente este metoda Newton, în care $c = c_k$ se modifică la fiecare iteraţie, astfel încât $1 + c_k f'(x_k) = 0$, respectiv:

$$c_k = -\frac{1}{f'(x_k)}. \tag{9.6}$$

Rezultă următoarea relaţie pentru definirea şirului soluţiilor numerice în metoda Newton:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 1, 2, \ldots \tag{9.7}$$

care evidenţiază faptul că la fiecare iteraţie graficul funcţiei este aproximat cu tangenta dusă în punctul de coordonate $x_k, f(x_k)$. Ecuaţia dreptei tangente este:

$$y = f'(x)(x - x_k) + f(x_k), \tag{9.8}$$

iar prin intersecţia ei cu axa, rezultă, pentru $y = 0$:

$$x = x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{9.9}$$

Din acest motiv, metoda Newton este cunoscută şi sub numele de *metoda tangentelor*. Dezavantajul acestei metode constă în faptul că la fiecare iteraţie trebuie evaluată derivata $f'(x_k)$, ceea ce poate necesita un efort mare de calcul.

Metoda Newton-Kantorovici (a tangentelor paralele)

O variantă simplificată, cunoscută sub numele de **Newton-Kantorovici (a tangentelor paralele)** foloseşte o singură evaluare a derivatei în punctul iniţial $c = -1/f'(x_0)$, iar iteraţiile sunt date de:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)} \tag{9.10}$$

Această metodă este, în fond, o variantă a metodei iteraţiei simple, cu o alegere convenabilă a factorului c.

Metoda Newton discretă (a secantelor)

Dacă expresia derivatei $f'(x)$ nu este cunoscută, atunci metoda Newton şi varianta sa simplificată nu pot fi aplicate. În acest caz se poate face apel la aproximarea numerică a derivatei:

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}, \tag{9.11}$$

aceasta reprezentând panta secantei ce uneşte ultimele două puncte din şirul iterativ, având coordonatele $x_{k-1}, f(x_{k-1})$ şi respectiv $x_k, f(x_k)$. Metoda astfel obţinută bazată pe relaţia iterativă:

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}, \quad k = 1, 2, \ldots \tag{9.12}$$

este cunoscută sub numele de *metoda Newton (sau metoda secantelor)*. Această metodă foloseşte o funcţie de iteraţie de două variabile $x_{k+1} = g(x_k, x_{k-1})$, deci necesită o iniţializare dublă x_0, x_1 (alese, de exemplu: $x_0 = a, x_1 = b$).

9.3 Pseudocodul algoritmilor

Metoda bisecţiei

Rezolvarea ecuaţiei $f(x) = 0$ prin metoda bisecţiei poate fi reprezentată în pseudocod prin următoarea procedură:

```
procedura bisecţie (a, b, eps, nit)
        real a, b                          ; domeniul de definiţie al funcţiei f
        real eps                           ; eroarea impusă de soluţie
        întreg nit                         ; număr maxim de iteraţii
        întreg k = 0                       ; contor iteraţii
        repetă
                k = k + 1
                xm = (a + b)/2
                dacă f(xm)f(a) > 0 atunci
                        a = xm
                altfel
                        b = xm
        până când (b − a) < eps sau k > nit
        dacă k ≤ nit
                scrie xm                   ; soluţie
retur
```

Metoda iteraţiei simple

Rezolvarea ecuaţiei $f(x) = 0$ prin metoda iteraţiei simple poate fi reprezentată în pseudocod prin următoarea procedură:

```
procedura iteraţie simplă (x0, eps, nit)
        real x0                            ; iniţializare soluţie
        real eps                           ; eroarea impusă de soluţie
        întreg nit                         ; număr maxim de iteraţii
        întreg k = 0                       ; contor iteraţii
        real xvechi = x0                   ; iniţializarea soluţiei
        repetă
                k = k + 1
                xnou = g(xvechi)           ; unde g(x)=x+cf(x)
                d = |xnou − xvechi|
                xvechi = xnou
        până când d < eps sau k > nit
        dacă k ≤ nit
```

scrie $xnou$

retur

Metoda Newton-Kantorovici (tangente paralele)

Rezolvarea ecuaţiei $f(x) = 0$ prin metoda Newton-Kantorovici(tangente paralele) poate fi reprezentată în pseudocod prin următoarea procedură:

procedura tangente paralele (x_0, eps, nit)
- **real** x_0 ; iniţializare soluţie
- **real** eps ; eroarea impusă de soluţie
- **întreg** nit ; număr maxim de iteraţii
- **real** $xvechi = x_0$; iniţializarea soluţiei
- **real** $fd = f'(x_0)$; valoarea derivatei în x_0
- **repetă**
  - $k = k + 1$
  - $xnou = xvechi - f(xvechi)/fd$
  - $d = |xnou - xvechi|$
  - $xvechi = xnou$
- **până când** $d < eps$ **sau** $k > nit$
- **dacă** $k \leq nit$
  - **scrie** $xnou$

retur

Metoda Newton (tangente)

Rezolvarea ecuaţiei $f(x) = 0$ prin metoda Newton (a tangentelor) poate fi reprezentată în pseudocod prin următoarea procedură:

procedura Newton (x_0, eps, nit)
- **real** x_0 ; iniţializare soluţie
- **real** eps ; eroarea impusă de soluţie
- **întreg** nit ; număr maxim de iteraţii
- **întreg** $k = 0$; contor iteraţii
- **real** $xvechi = x_0$; iniţializarea soluţiei
- **repetă**
  - $k = k + 1$
  - $xnou = xvechi - f(xvechi)/f'(xvechi)$
  - $d = |xnou - xvechi|$
  - $xvechi = xnou$
- **până când** $d < eps$ **sau** $k > nit$
- **dacă** $k \leq nit$

scrie $xnou$

retur

Metoda Newton discretă (a secantelor)

Rezolvarea ecuaţiei $f(x) = 0$ prin metoda Newton discretă (a secantelor) poate fi reprezentată în pseudocod prin următoarea procedură:

procedura secante (a, b, eps, nit)
- **real** a, b ; domeniul de definiție al funcției
- **real** eps ; eroarea impusă de soluție
- **întreg** nit ; număr maxim de iterații
- **întreg** $k = 0$; contor iterații
- **real** $xv = a$; inițializări ale soluției
- **real** $xvv = b$
- **repetă**
  - $k = k + 1$
  - $xnou = xv - (xv - xvv)f(xv)/(f(xv) - f(xvv))$
  - $d = |xnou - xv|$
  - $xvv = xv$
  - $xv = xnou$
- **până când** $d < eps$ **sau** $k > nit$
- **dacă** $k \leq nit$
  - **scrie** $xnou$

retur

9.4 Analiza algoritmilor

Efort de calcul

Efortul de calcul pentru determinarea soluţiei numerice este dependent de eroarea impusă soluţiei. În general, cu cât precizia impusă soluţiei este mai mare, cu atât timpul de calcul creşte, deoarece sunt necesare mai multe iteraţii.

Efortul de calcul pentru o iteraţie depinde de metoda adoptată şi el constă în principal în evaluarea funcţiei f sau a derivatei acesteia:

Metoda	Număr de evaluări pe iteraţie
Bisecţiei	2 evaluări pentru f (poate fi redusă la o evaluare)
Iteraţia simplă	1 evaluare pentru f
Tangente paralele	1 evaluare pentru f
Newton	1 evaluare pentru f şi 1 evaluare pentru derivata f'
Secante	2 evaluări pentru f (poate fi redusă la o evaluare)

Deoarece viteza cu care şirul de iteraţii converge către soluţia exactă este diferită de la o metodă la alta, este posibil ca numărul de iteraţii necesare atingerii unei precizii dorite să fie mult mai mic în cazul metodei Newton decât în cazul metodei iteraţiei simple. În acest caz se obţine o eficienţă globală superioară prin folosirea metodei Newton, chiar dacă la fiecare iteraţie timpul de calcul este mai mare.

Analiza erorilor

O altă problemă importantă în procedurile iterative constă în încrederea care se poate acorda soluţiei numerice. Faptul că doi termeni succesivi sunt suficient de apropiaţi $|x_{k+1} - x_k| < eps$ nu înseamnă întotdeauna că aceştia sunt foarte aproape de soluţia exactă x. Este posibil ca distanţa $err = |x_{k+1} - x|$ să fie mult mai mare decât eps. Folosind principiul contracţiei se obţin următoarele relaţii privind eroarea impusă eps şi eroarea reală err în cazul metodei iteraţiei simple:

$$err = |x_{n+1} - x| \leq \frac{L^{n+1}}{1-L}|x_1 - x_0| \tag{9.13}$$

$$eps = |x_{n+1} - x_n| \leq L^{n+1}|x_0 - x| \tag{9.14}$$

în care s-a notat cu $L < 1$ constanta Liepschitz a funcţiei de iteraţie g, ce satisface:

$$|g(u) - g(v)| \leq L|u - v| \tag{9.15}$$

pentru orice u, v din domeniul (a, b). Cu cât această constantă are valori mai mici, cu atât şirul iterativ este mai rapid convergent, numărul de iteraţii necesar atingerii preciziei impuse fiind:

$$n = \left\lceil \frac{\log(1-L)\dfrac{eps}{|x_1 - x_0|}}{\log L} \right\rceil \tag{9.16}$$

Raportul dintre eroarea impusă şi cea reală depinde de constanta L.

Deoarece în metoda bisecţiei marginea erorii reale la iteraţia n ($merr_n$) satisface inegalitatea:

$$merr_n \leq \frac{merr_{n-1}}{2}, \tag{9.17}$$

rezultă că metoda iteraţiei simple este mai avantajoasă decât metoda bisecţiei, doar atunci când constanta Liepschitz $L < 1/2$.

Metoda Newton este mai rapid convergentă decât metoda iteraţiei simple deoarece în acest caz, eroarea reală la fiecare pas este:

$$err_{n+1} = |x_{n+1} - x| = |g(x_n) - g(x)| \leq L(|x_n - x|)^2 \tag{9.18}$$

spre deosebire de eroarea reală la fiecare pas în cazul metodei iteraţiei simple:

$$err_{n+1} = |x_{n+1} - x| = |g(x_n) - g(x)| \leq L|x_n - x|. \tag{9.19}$$

În concluzie, efortul de calcul pentru determinarea soluţiei numerice a unei ecuaţii neliniare printr-o metodă iterativă depinde de:

- tipul ecuaţiei;
- iniţializarea adoptată (distanţa faţă de soluţia exactă);
- metoda iterativă adoptată;
- eroarea impusă soluţiei numerice.

Metodele iterative nu sunt convergente în orice caz. Aceasta situatie poate avea loc de exemplu, în cazul metodei iteraţiei simple dacă constanta L a funcţiei g are valori supraunitare.

9.5 Chestiuni de studiat

1. Rezolvarea unor ecuaţii neliniare prin diferite metode iterative;
2. Analiza experimentală a erorilor şi a timpului de calcul la rezolvarea prin metode iterative a ecuaţiilor neliniare;
3. Implementarea într-un limbaj de programare a unor algoritmi de rezolvare iterativă a ecuaţiilor neliniare şi testarea acestora;
4. Căutarea de informaţii pe Internet.

9.6 Modul de lucru

Pentru desfăşurarea lucrării selectaţi opţiunea *Rezolvarea numerică prin metode iterative a ecuaţiilor neliniare* din meniul principal de lucrări. Aceasta are ca urmare lansarea următorului meniu:

- Rezolvare ecuaţii
- Analiza rezolvării

Utilizatorul va selecta opţiunile din acest meniu.

9.6.1 Rezolvarea unor ecuaţii neliniare prin diferite metode iterative

Prin selectarea opţiunii *Rezolvare ecuaţii* din meniul lucrării se apelează un program care poate rezolva următoarele ecuaţii neliniare:

$$e^x = a_0$$
$$x + a_0 \ln x = a_1$$
$$a_0 \sin x + a_1 \cos x = a_2$$
$$a_3 x^3 + a_2 x^2 + a_1 x + a_0 = 0$$

în care a_0, a_1, a_2, a_3, sunt parametri reali ce vor fi introduşi de utilizator. După introducerea acestor parametri, programul solicită domeniul de definiţie $[x_{min}, x_{max}]$ al funcţiei, numărul maxim de iteraţii şi apoi alegerea metodei de rezolvare dintre variantele:

1. metoda bisecţiei;
2. metoda iteraţiei simple (cu $c = -1$);
3. metoda tangentelor paralele;
4. metoda tangentelor;
5. metoda secantelor.

În continuare programul afişează graficul funcţiei $f(x)$ şi marchează succesiv punctele de coordonate $x_k, f(x_k), k = 0, 1, 2, \ldots$

Pentru trecerea la iteraţia următoare trebuie apăsată tasta <ENTER> în consola Scilab.

Se recomandă rezolvarea ecuaţiilor: $e^x = 2, x + \ln x = 2, \sin(x) + \cos(x) = 1$ şi $x^2 = 2$, prin diferite metode. Se vor alege limitele x_{min} şi x_{max} ale domeniului de definiţie în mod convenabil pentru fiecare ecuaţie şi se va comenta convergenţa procesului iterativ, aşa cum este ea evidenţiată din grafic.

Pentru fiecare metodă disponibilă, se va căuta un exemplu de ecuaţie, la care iteraţiile nu sunt convergente.

9.6.2 Analiza experimentală a erorilor şi a timpului de calcul la rezolvarea prin metode iterative a ecuaţiilor neliniare

Prin selectarea optiunii *Analiza rezolvării* din meniul principal se apelează un program care rezolvă ecuaţia algebrică:

$$a_2 x^2 + a_1 x + a_0 = 0$$

prin diferite metode iterative şi anume:

1. metoda bisecţiei;
2. metoda iteraţiei simple;
3. metoda tangentelor paralele;
4. metoda tangentelor;
5. metoda secantelor.

La fiecare metodă este afişată şi reprezentată grafic variaţia erorii absolute reale $|x_k - x|$ (abaterea iteraţiei curente faţă de soluţia exactă), a erorii aparente Cauchy $|x_k - x_{k-1}|$ (distanţa dintre două iteraţii succesive egală cu modulul corecţiei efectuate asupra ultimei soluţii numerice) şi a reziduului $|f(x_k)|$ pe parcursul iteraţiilor. Programul afişează şi timpul necesar pentru efectuarea unei iteraţii.

Se va rezolva ecuaţia $x^2 = 2$ şi se va calcula timpul necesar atingerii unei precizii dorite (de exemplu, primele trei cifre semnificative exacte) pentru soluţia numerică. Se vor compara rezultatele obţinute prin diferite metode (eficienţa lor relativă în rezolvarea problemei date) şi se va determina metoda optimă.

9.6.3 Implementarea algoritmilor de rezolvare iterativă a ecuaţiilor neliniare

Se va implementa în limbajul C una din procedurile de rezolvare iterativă prezentate. Se va scrie un program principal, care va apela procedura implementată şi care va rezolva ecuaţia algebrică neliniară:

$$a_3x^3 + a_2x^2 + a_1x + a_0 = 0$$

Programul va permite:

- introducerea datelor a_0, a_1, a_2, a_3, a, b;
- rezolvarea ecuaţiei prin apelul procedurii;
- afişarea rezultatelor: x - soluţia numerică; k - numărul de iteraţii; *eps* - eroarea aparentă Cauchy.

9.6.4 Căutare de informaţii pe Internet

Căutaţi pe Internet informaţii şi coduri legate de rezolvarea numerică a ecuaţiilor neliniare. Cuvinte cheie recomandate: *solving nonlinear equations*, *solving nonlinear systems*, *secant method*, etc.

9.7 Exemple

9.7.1 Exemple rezolvate

1. Fie funcţia $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2 - 9$. Se cer:

   (a) Să se calculeze primele două iteraţii ale metodei bisecţiei pentru rezolvarea ecuaţiei $f(x) = 0$ în intervalul $[a, b]$, unde $a = 0$, $b = 8$.

   (b) Să se calculeze primele trei iteraţii ale metodei Newton pentru rezolvarea ecuaţiei $f(x) = 0$, pornind de la iniţializarea $x_0 = 1$, respectiv $x_0 = -1$.

   (c) Comentaţi comportarea metodei Newton pentru rezolvarea ecuaţiei $f(x) = 0$, dacă iniţializarea este $x_0 = 0$.

   (d) Să se calculeze primele două iteraţii ale metodei Newton-Kantorovici pentru rezolvarea ecuaţiei $f(x) = 0$, pornind de la iniţializarea $x_0 = 2$.

   (e) Să se calculeze primele două iteraţii ale metodei Newton discretă pentru rezolvarea ecuaţiei $f(x) = 0$, pornind de la iniţializarea dublă $x_0 = 0$, $x_1 = 1$.

   Rezolvare:

   (a) Metoda bisecţiei (înjumătăţirii intervalului) este garantat convergentă dacă funcţia este continuă, soluţia este unică (problema este bine formulată matematic) şi funcţia îşi schimbă semnul în intervalul de căutare, $[a, b]$ ($f(a)f(b) < 0$).

   Pentru funcţia dată avem: $f(a) = f(0) = -9$ şi $f(b) = f(8) = 55$, adică $f(a)f(b) = -9 \cdot 55 < 0$.

   La prima iteraţie, $k = 1$, mijlocul intervalului este: $x_{m,1} = (a + b)/2 = 4$, $f(x_{m,1}) = f(4) = 7$. Deoarece $f(a)f(x_{m,1}) = -9 \cdot 7 < 0$, soluţia se află în prima jumătate a intervalului $[a, b]$. Valoarea lui b se modifică, $b = x_{m,1} = 4$. Noul interval de căutare este $[0, 4]$.

   La a doua iteraţie, $k = 2$, mijlocul intervalului este: $x_{m,2} = (a + b)/2 = 2$, $f(x_{m,2}) = f(2) = -5$. Deoarece $f(a)f(x_{m,2}) = -9 \cdot (-5) > 0$, soluţia se află în a doua jumătate a intervalului $[a, b]$. Acum, valoarea lui a se modifică, $a = x_{m,2} = 2$. Noul interval de căutare este $[2, 4]$.

   Procedeul iterativ este ilustrat în figura 11.

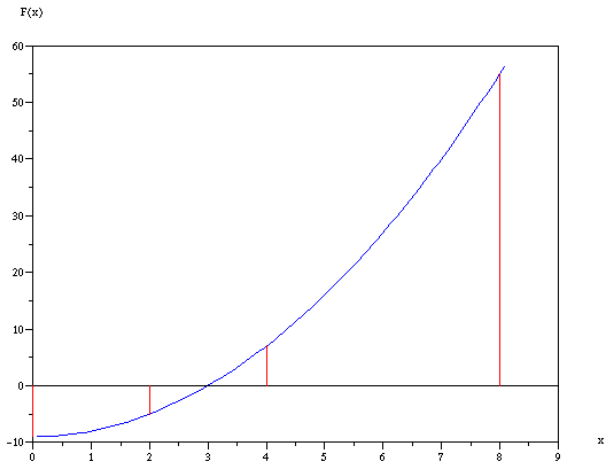

Fig. 11. Primele două iteraţii ale metodei bisecţiei

(b) Formula de recurenţă a metodei Newton (a tangentelor) pentru generarea şirului de soluţii este:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 1, 2, \ldots.$$

Se observă că este necesară evaluarea derivatei la fiecare iteraţie. În cazul problemei considerate, $f'(x) = 2x$,

Dacă iniţializarea este $x_0 = 1$, primele 3 iteraţii ale metodei Newton sunt:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 1 - \frac{f(1)}{f'(1)} = 1 - \frac{-8}{2} = 5,$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 5 - \frac{f(5)}{f'(5)} = 5 - \frac{16}{10} = \frac{17}{5} = 3.4,$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = \frac{17}{5} - \frac{f(17/5)}{f'(17/5)} = \frac{17}{5} - \frac{32}{85} = \frac{257}{85} = 3.02.$$

Procedeul iterativ este ilustrat în figura 1b-stânga.

Dacă iniţializarea este $x_0 = -1$, primele 3 iteraţii sunt:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = -1 - \frac{f(-1)}{f'(-1)} = -1 - \frac{-8}{-2} = -5,$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = -5 - \frac{f(-5)}{f'(-5)} = -5 - \frac{16}{-10} = -\frac{17}{5} = -3.4,$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = -\frac{17}{5} - \frac{f(-17/5)}{f'(-17/5)} = -\frac{17}{5} + \frac{32}{85} = -\frac{257}{85} = -3.02.$$

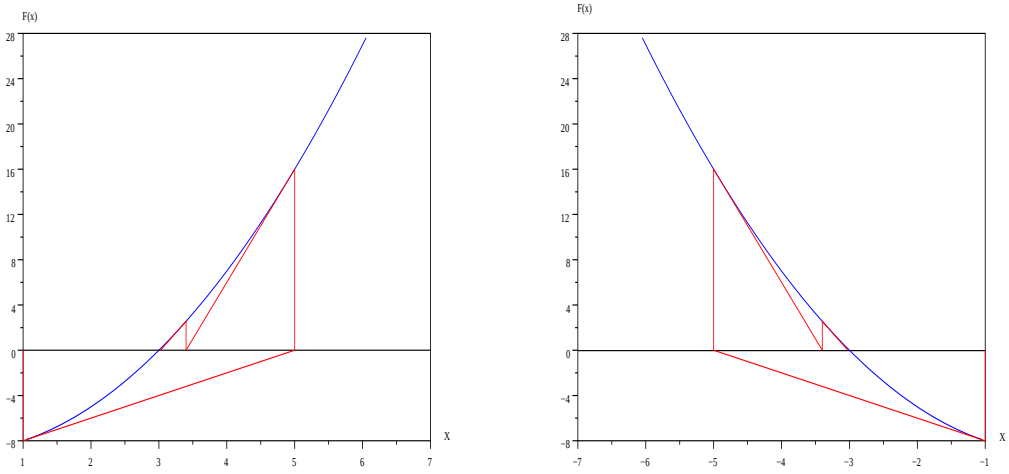

Figura 9.1: Primele trei iteraţii ale metodei Newton: stânga $x_0 = 1$, dreapta $x_0 = -1$.

Procedeul iterativ este ilustrat în figura 1b-dreapta.

Se observă că, în funcţie de iniţializarea soluţiei, metoda Newton converge către una dintre cele două soluţii exacte ale ecuaţiei $x^2 - 9 = 0$, $x = \pm 3$.

(c) Dacă iniţializarea este $x_0 = 0$, atunci prima iteraţie a metodei Newton ar fi:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0 - \frac{f(0)}{f'(0)} = \frac{9}{0},$$

şi metoda eşuează încă de la prima iteraţie.

Metoda Newton va eşua dacă o iteraţie ajunge într-un punct critic al funcţiei (punct pentru care derivata este nulă). Din punct de vedere geometric, într-un punct critic, tangenta la graficul funcţiei $f(x)$ este paralelă cu abscisa OX, deci nu există intersecţie cu aceasta, implicit soluţie la iteraţia următoare.

(d) Formula de recurenţă a metodei Newton-Kantorovici (a tangentelor paralele) pentru generarea şirului de soluţii este:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}, \quad k = 1, 2, \ldots.$$

Se observă că este necesară o singură evaluare a derivatei, $f'(x_0) = 2x_0$.

Dacă iniţializarea este $x_0 = 2$, primele 2 iteraţii ale metodei Newton-Kantorovici sunt:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 2 - \frac{f(2)}{f'(2)} = 2 - \frac{-5}{4} = \frac{13}{4} = 3.25,$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_0)} = \frac{13}{4} - \frac{f(13/4)}{f'(2)} = \frac{13}{4} - \frac{25}{64} = \frac{183}{64} = 2.859.$$

Procedeul iterativ este ilustrat în figura 1e-stânga.

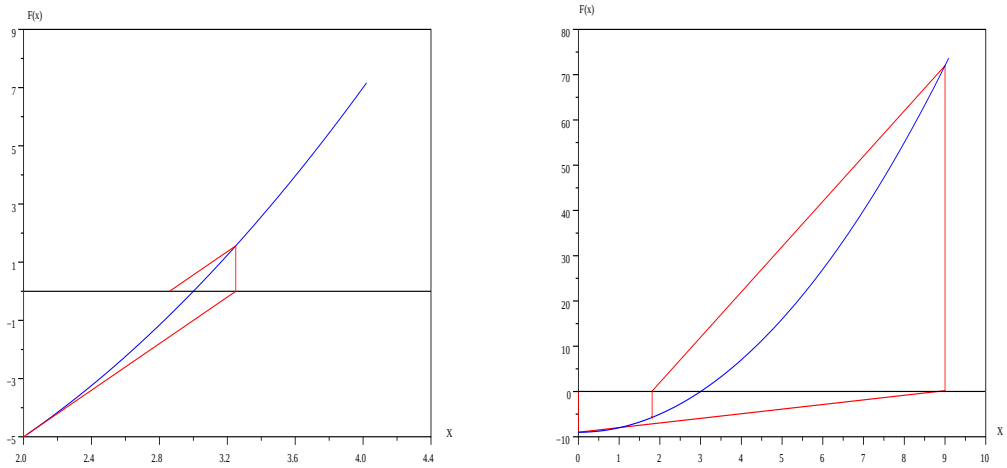

Figura 9.2: Primele trei iteraţii ale metodei: Newton Kantorovici (stânga), Newton discretă (dreapta).

(e) Formula de recurenţă a metodei Newton discretă (a secantelor) pentru generarea şirului de soluţii este:

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}, \quad k = 1, 2, \ldots.$$

Pornind de la iniţializarea dublă $x_0 = 0$, $x_1 = 1$, primele 2 iteraţii ale metodei Newton discretă sunt:

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} = 1 - \frac{f(1)(1-0)}{f(1) - f(0)} = 1 - \frac{-8}{1} = 9,$$

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} = 9 - \frac{f(9)(9-1)}{f(9) - f(1)} = 9 - \frac{36}{5} = \frac{9}{5} = 1.8.$$

Procedeul iterativ este ilustrat în figura 1e-dreapta.

2. Fie ecuaţia neliniară $x^2 + x = 2$. Se cer:

(a) Să se calculeze primele două iteraţii ale metodei bisecţiei pentru rezolvarea ecuaţiei pentru $x \in [a, b]$, unde $a = -6$, $b = 0$.

(b) Să se calculeze primele două iteraţii ale metodei Newton pentru rezolvarea ecuaţiei, pornind de la iniţializarea $x_0 = 0$.

(c) Să se calculeze primele două iteraţii ale metodei Newton-Kantorovici pentru rezolvarea ecuaţiei, pornind de la iniţializarea $x_0 = 0$.

(d) Să se calculeze primele două iteraţii ale metodei Newton discretă pentru rezolvarea ecuaţiei, pornind de la iniţializarea dublă $x_0 = -1/2$, $x_1 = 0$.

Rezolvare:

La rezolvarea oricărei ecuaţii neliniare este indicat să avem forma $f(x) = 0$. În cazul considerat, $f(x) = x^2 + x - 2$.

(a) Pentru funcţia dată avem: $f(a) = f(-6) = 28$ şi $f(b) = f(0) = -2$, adică $f(a)f(b) = 28 \cdot (-2) < 0$, o singură soluţie în intervalul $[a, b]$.

La prima iteraţie, $k = 1$: $x_{m,1} = (a + b)/2 = -3$, $f(x_{m,1}) = f(-3) = 4$. Deoarece $f(a)f(x_{m,1}) = 28 \cdot 4 > 0$, soluţia se află în a doua jumătate a intervalului $[a, b]$. Valoarea lui a se modifică, $a = x_{m,1} = -3$. Noul interval de căutare este $[-3, 0]$.

La a doua iteraţie, $k = 2$: $x_{m,2} = (a + b)/2 = -3/2$, $f(x_{m,2}) = f(-3/2) = -5/4$. Deoarece $f(a)f(x_m) = 4 \cdot (-5/4) < 0$, soluţia se află în prima jumătate a intervalului $[a, b]$. Acum, valoarea lui b se modifică, $b = x_{m,2} = -3/2$. Noul interval de căutare este $[-3, -3/2]$.

(b) Pentru metoda Newton este necesară cunoaşterea expresiei derivatei funcţiei. În cazul problemei considerate, $f'(x) = 2x + 1$.

Pornind de la iniţializarea $x_0 = 0$, primele două iteraţii ale metodei Newton sunt:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0 - \frac{f(0)}{f'(0)} = -\frac{-2}{1} = 2,$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 2 - \frac{f(2)}{f'(2)} = 2 - \frac{4}{5} = \frac{6}{5} = 1.2.$$

(c) Pornind de la iniţializarea $x_0 = 0$, primele două iteraţii ale metodei Newton-Kantorovici sunt:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0 - \frac{f(0)}{f'(0)} = -\frac{-2}{1} = 2,$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_0)} = 2 - \frac{f(2)}{f'(0)} = 2 - \frac{4}{1} = -2.$$

(d) Pornind de la iniţializarea dublă $x_0 = -1/2$, $x_1 = 0$, primele două iteraţii ale metodei Newton discretă sunt:

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} = 0 - \frac{f(0)[0 - (-1/2)]}{f(0) - f(-1/2)} = -\frac{-1}{-2 - (-9/4)} = 4,$$

$$x_3 = x_2 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} = 4 - \frac{f(4)(4 - 0)}{f(4) - f(0)} = 4 - \frac{18 \cdot 4}{20} = \frac{2}{5} = 0.4.$$

9.7.2 Exemple propuse

1. Fie funcţia $f : \mathbb{R} \to \mathbb{R}$, $f(x) = x^2 - 16$. Se cer:
   (a) Să se calculeze primele două iteraţii ale metodei bisecţiei pentru rezolvarea ecuaţiei $f(x) = 0$ în intervalul $[a, b]$, unde $a = 0$, $b = 12$.
   (b) Să se calculeze primele două iteraţii ale metodei Newton pentru rezolvarea ecuaţiei $f(x) = 0$, pornind de la iniţializarea $x_0 = 1$.
   (c) Să se calculeze primele două iteraţii ale metodei Newton-Kantorovici pentru rezolvarea ecuaţiei $f(x) = 0$, pornind de la iniţializarea $x_0 = 1$.
   (d) Să se calculeze primele două iteraţii ale metodei Newton discretă pentru rezolvarea ecuaţiei $f(x) = 0$, pornind de la iniţializarea dublă $x_0 = 0$, $x_1 = 1$.
2. Fie ecuaţia neliniară $x^2 + x = 6$. Se cer:
   (a) Să se calculeze primele două iteraţii ale metodei bisecţiei pentru rezolvarea ecuaţiei pentru $x \in [a, b]$, unde $a = -8$, $b = 0$.
   (b) Să se calculeze primele două iteraţii ale metodei Newton pentru rezolvarea ecuaţiei, pornind de la iniţializarea $x_0 = -1$.
   (c) Să se calculeze primele două iteraţii ale metodei Newton-Kantorovici pentru rezolvarea ecuaţiei, pornind de la iniţializarea $x_0 = -1$.
   (d) Să se calculeze primele două iteraţii ale metodei Newton discretă pentru rezolvarea ecuaţiei, pornind de la iniţializarea dublă $x_0 = -2$, $x_1 = -1$.

9.8 Întrebări şi probleme

1. Comparaţi eficienţa diferitelor metode iterative pentru rezolvarea aceleiaşi ecuaţii neliniare;
2. Analizaţi cum depinde eficienţa rezolvării de iniţializarea adoptată;
3. Analizaţi cum depinde timpul de calcul pentru determinarea soluţiei numerice a unei ecuaţii neliniare de eroarea impusă soluţiei numerice;
4. Modificaţi algoritmul metodei bisecţiei astfel încât să se efectueze o singură evaluare a funcţiei f la fiecare înjumătăţire;
5. Analizaţi efectul pe care îl are introducerea unui factor de relaxare (subrelaxare, respectiv suprarelaxare) în iteraţiile neliniare;
6. Care din metodele prezentate poate fi generalizată la rezolvarea unui sistem de ecuaţii neliniare şi cum se realizează această generalizare?

7. Propuneţi diferite criterii sigure pentru oprirea iteraţiilor;

8. Generaţi un algoritm pentru determinarea iterativă a zerourilor funcţiei complexe analitice.

9. Descrieţi algoritmul Newton în cazul unui sistem de ecuaţii algebrice neliniare. Cum poate fi aplicat acest algoritm în analiza numerică a circuitelor electrice neliniare în regim staţionar (de c.c.)? Arătaţi că la fiecare iteraţie trebuie rezolvat un circuit electric liniar, pentru care se poate aplica algoritmul din Lucrarea nr. 5.

10. Identificaţi în biblioteca matematică descrisă în [2] funcţii avansate pentru rezolvarea sistemelor de ecuaţii neliniare. Ce aduc nou aceste funcţii, faţă de cele analizate în lucrare?

Sir Isaac Newton (1643, Anglia - 1727, Anglia)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Newton.html

Leonid Vitalyevich Kantorovich (1912, Rusia - 1986, URSS)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Kantorovich.html
http://www.nobel.se/economics/laureates/1975/kantorovich-autobio.html

Lucrarea 10

Rezolvarea ecuaţiilor diferenţiale ordinare cu condiţii iniţiale prin metoda Euler

10.1 Caracterizarea metodei

Metoda Euler este o metodă elementară de integrare numerică a unei ecuaţii diferenţiale cu condiţii iniţiale de forma:

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= f(t,y) \\ y(t_0) &= y_0 \end{aligned}$$

pentru intervalul de definiţie $t_0 \leq t \leq t_{max}$.

În această metodă, pentru evaluarea numerică a derivatei se utilizează aproximarea obţinută prin reţinerea primilor doi termeni din dezvoltarea în serie Taylor a funcţiei $y(t)$. Deoarece soluţia numerică are pe fiecare pas de timp variaţie liniară, se spune că metoda este de ordinul I.

10.2 Principiul metodei

Se consideră următoarea ecuaţie diferenţială de ordinul I:

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= f(t,y) \\ y(0) &= y_0 \end{aligned} \tag{10.1}$$

a cărei soluţie $y(t)$ este o funcţie de timp definită pe intervalul $[0, t_{max}]$.

Rezolvarea numerică a ecuaţiei (10.1) constă în evaluarea funcţiei $y(t)$ în nodurile $0 < t_1 < t_2 < \ldots < t_{max}$ ale intervalului de definiţie. Se alege un pas de integrare h, astfel încât intervalul $[0, t_{max}]$ să fie împărţit în n paşi egali:

$$h = \frac{t_{max}}{n}. \tag{10.2}$$

Nodurile intervalului de definiţie sunt în acest caz următoarele:

$$t_0 = 0;\ t_1 = h;\ t_2 = 2h;\ t_3 = 3h; \ldots; t_n = nh = t_{max}.$$

Dezvoltarea în serie Taylor a funcţiei $y(t)$ la dreapta punctului $t_i = ih$ este:

$$y(t_i + h) = y_i + y_i' h + y_i'' \frac{h^2}{2} + y_i''' \frac{h^3}{6} + \ldots \tag{10.3}$$

unde $y_i = y(t_i)$ şi $y_i' = f(t_i, y_i)$. Dacă se neglijează termenii de grad mai mare sau egal cu 2, şi se notează cu y_{i+1} soluţia la momentul $t_{i+1} = t_i + h$, atunci se face următoarea aproximare:

$$y_{i+1} = y_i + hf(t_i, y_i). \tag{10.4}$$

Această expresie reprezintă *formula explicită* a metodei lui Euler. Pornind de la condiţia iniţială y_0, relaţia (10.4) permite calculul succesiv al soluţiei numerice $y_1, y_2, \ldots, y_n$, în toate nodurile reţelei de discretizare.

Dezvoltarea în serie Taylor a funcţiei $y(t)$ la stânga punctului $t_{i+1} = (i+1)h$ este:

$$y(t_{i+1} - h) = y_{i+1} - y_{i+1}' h + y_{i+1}'' \frac{h^2}{2} - y_{i+1}''' \frac{h^3}{6} + \ldots \tag{10.5}$$

unde $y_{i+1}' = f(t_{i+1}, y_{i+1})$. Dacă se neglijează în (10.5) termenii de grad mai mare sau egal cu 2 rezultă următoarea aproximare:

$$y_{i+1} = y_i + hf(t_{i+1}, y_{i+1}). \tag{10.6}$$

Această expresie reprezintă *formula implicită* a metodei lui Euler. Determinarea soluţiei numerice presupune în acest caz rezolvarea la fiecare pas de timp a ecuaţiei (10.6).

10.3 Pseudocodul metodei

Rezolvarea prin *metoda explicită* este descrisă de următorul pseudocod:

```
procedura euler_exp (x0, xmax, y0, h, y)
    real x0,                    ; nod inițial
    real xmax,                  ; nod final
    real y0,                    ; condiția inițială
    real h,                     ; pasul de integrare
    tablou real y(n)            ; vectorul soluției numerice (indici de la 0)
    t = x0
    y(0) = y0
    n = (xmax - x0)/h           ; număr de pași
    pentru i = 1, n             ; evaluează funcția în cele n noduri
                y(i) = y(i-1) + h · func(t, y(i -1))
                t = t + h
retur
```

Procedura are următorii parametri:

- de intrare:
  - $x0$ = limita inferioară a intervalului de integrare;
  - $xmax$ = limita superioară a intervalului de integrare;
  - $y0$ = condiţia iniţială;
  - h = pasul de integrare.
- de ieşire: $y(n)$ = vectorul soluţie.

Procedura apelează funcţia `func` ce evaluează funcţia $f(t, y)$ din membrul drept al ecuaţiei (10.1) pentru nodul i.

Rezolvarea prin *metoda implicită* este descrisă de următorul pseudocod:

```
procedura euler_imp (x0, xmax, y0, h, err, itmax, y)
    real x0,                                ; nodul inițial
    real xmax,                              ; nod final
    real y0,                                ; condiția inițială
    real h,                                 ; pasul de integrare
    real err                                ; eroarea maxim admisă
    întreg itmax                            ; nr. max. de iterații
    tablou real y(N)                        ; vectorul soluției
    t = x0
    y(0) = y0
    n = (xmax - x0)/h                       ; nr. de pași
```

```
pentru i = 1, n                                  ; evaluează funcţia în cele n noduri
    t = t + h                                    ; pas nou
    ynou = y(i-1) + h · func(t, y(i-1))          ; prima evaluare se face cu metoda explicită
    j = 0                                        ; contor iteraţii
    repetă
            yvechi = ynou
            ynou = y(i-1) + h · func(t, yvechi)  ; evaluare nouă
            j = j + 1
            eps = abs(yvechi - ynou)             ; evaluare eroare
    până când (abs(eps) ≤ err sau j > itmax)
    y(i) = ynou
retur
```

Procedura are următorii parametri:

- de intrare:
  - $x0$ - limita inferioară a intervalului de integrare;
  - $xmax$ - limita superioară a intervalului de integrare;
  - $y0$ - condiţia iniţială;
  - h - pasul de integrare;
  - err - eroarea maximă admisă;
  - itmax - nr. maxim de iteraţii admis.
- de ieşire: y(n) - vectorul soluţiei numerice.

Procedura apelează funcţia `func` ce calculează valoarea funcţiei $f(t, y)$ din membrul drept al ecuaţiei (10.1) pentru nodul i.

Metoda implementată pentru rezolvarea ecuaţiei neliniare (10.6) este metoda iteraţiei simple, la care iniţializarea este făcută prin valori date de relaţia explicită (10.4).

10.4 Analiza algoritmului

Efortul de calcul

Deoarece durata evaluării funcţiei $f(t, y)$ este în general mult mai mare decât durata efectuării câtorva înmulţirii şi adunări, timpul de calcul depinde de numărul de evaluări ale funcţiei $f(t, y)$.

În *metoda explicită* se efectuează n evaluări ale funcţiei f.

În cazul *metodei implicite*, aplicată pe aceeaşi reţea de discretizare, timpul de calcul creşte odată cu creşterea preciziei de calcul dorite în rezolvarea iterativă a ecuaţiei neliniare (10.6). Ciclul de iteraţii corectează succesiv valoarea soluţiei de la momentul curent. Această tehnică bazată pe relaţiile (10.4) şi (10.6) se numeşte **metoda predictiv-corectivă**.

Numărul de evaluări ale funcţiei f este majorat de $n \cdot$itmax.

Deoarece metoda implicită are o stabilitate numerică superioară faţă de metoda explicită, pentru calculul soluţiei numerice la un moment dat (de exemplu t_{max}) cu o eroare impusă, trebuie folosită o reţea de discretizare mai fină în cazul utilizării metodei explicite, ceea ce poate conduce la un efort de calcul global mai mic prin utilizarea metodei implicite.

Necesar de memorie

Pentru memorarea vectorului y sunt necesare n locaţii de memorie, fiecare din ele fiind rezervată unui număr real. În scopul eliminării acestui consum de memorie, soluţia poate fi afişată (sau salvată pe disc), imediat după ce a fost calculată.

Cele două variante ale metodei Euler (explicită şi implicită), necesită practic acelaşi necesar de memorie.

Analiza erorilor

Pentru o ecuaţie diferenţială simplă de forma:

$$\begin{aligned} \frac{dy}{dt} &= ay \\ y(0) &= y_0 \end{aligned} \tag{10.7}$$

se demonstrează că *eroarea de trunchiere* cumulată după k paşi de integrare cu metoda Euler explicită este:

$$eps_k \leq h y_0 a^2 t_k \exp(a t_k) = 0(h),$$

unde $t_k = hk$, $k \neq 1$, iar $eps_k = |y(t_k) - y_k|$ este eroarea la pasul k.

Pentru $k = 1$:

$$eps_1 \leq y_0 h^2 a^2 \exp(ah) = 0(h^2),$$

unde eps_1 este eroare locală, efectuată după un singur pas de integrare şi care este cu un ordin mai mare decât eroarea globală.

Eroarea de trunchiere este cu atât mai mică cu cât pasul de timp este mai mic. Pentru a obţine aceeiaşi eroare de trunchiere, în cazul metodei Euler explicite se impune alegerea unui pas de timp mai mic decât în cazul metodei Euler implicite.

În cazul în care se implementează metoda Euler pe un sistem de calcul care operează cu q cifre semnificative, *eroarea de rotunjire*, care se adaugă erorii de trunchiere, este:

$$eps_r = \left| \frac{10^{-q+1}}{ah} \right|.$$

Datorită efectului de anulare prin scădere, eroarea de rotunjire este cu atât mai mare cu cât pasul h este mai mic. Prin urmare pasul de integrare poate fi scăzut (în vederea micşorării erorii de trunchiere) numai atât timp cât eroarea de rotunjire rămâne suficient de mică. Din acest punct de vedere se poate găsi un pas de integrare optim.

10.5 Chestiuni de studiat

1. Rezolvarea unor ecuaţii diferenţiale de ordinul 1;
2. Analiza experimentală a erorilor şi a timpului de calcul în funcţie de pasul de integrare şi de eroarea impusă;
3. Rezolvarea unei ecuaţii diferenţiale asociate unui circuit electric, de ordin I, excitat cu un impuls;
4. Implementarea algoritmului într-un limbaj de programare şi testarea rutinei;
5. Căutarea de informaţii pe Internet.

10.6 Mod de lucru

Pentru desfăşurarea lucrării selectaţi opţiunea *Rezolvarea ecuaţiilor diferenţiale ordinare cu condiţii iniţiale prim metoda Euler* din meniul principal de lucrări. Aceasta are ca urmare lansarea următorului meniu:

- Rezolvare ecuaţie test
- Analiză erori
- Analiză ecuaţie

din care trebuie selectate succesiv opţiunile.

10.6.1 Rezolvarea unor ecuaţii diferenţiale de ordin 1

Se selectează opţiunea *Rezolvare ecuaţie test.* Programul rezolvă prin metoda Euler o ecuaţie diferenţială de forma:

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= ay + b \\ y(0) &= y_0 \end{aligned} \tag{10.8}$$

unde $y : [0, t_{max}] \to \mathbb{R}$. Această ecuaţie se obţine din studiul regimului tranzitoriu pentru următoarele circuite:

- Condensator de capacitate C ce se încarcă de la o sursă de tensiune continuă E, printr-un rezistor de rezistenţă R.
- Descărcarea unui condensator încărcat iniţial la tensiunea E pe un rezistor de rezistenţă R.

Conform ecuaţiilor lui Kirchhoff, tensiunea la bornele condensatorului într-un circuit serie RC excitat cu o sursă cu t.e.m. E satisface relaţia:

$$\begin{aligned} RC\frac{\mathrm{d}u}{\mathrm{d}t} + u &= E \\ u(0) &= u_0 \end{aligned}$$

unde u_0 este tensiunea iniţială la bornele condensatorului.

Se vor determina, în cele două cazuri, expresiile analitice ale tensiunii la bornele condensatorului.

Pentru R, C, E, cunoscute (de exemplu $R = 100\Omega$, $C = 10\mu$F, $E = 20$ V) se vor introduce coeficienţii corespunzători în ecuaţia (10.8) şi se vor compara graficele de variaţie a tensiunii la bornele condensatarului în timp cu graficele obţinute pe baza expresiei analitice găsite.

Comparaţiile se vor face pentru diferiţi paşi de integrare: $h = \tau/10$, $h = \tau$, $h = 2\tau$, unde $\tau = RC$ este constantă de timp a circuitului.

Se va testa ecuaţia şi pentru valori negative ale lui R (de exemplu $R = -100K\Omega$).

Se vor comenta rezultatele.

10.6.2 Analiza experimentală a erorilor şi a timpului de calcul în funcţie de pasul de integrare

Se selectează opţiunea *Analiză erori*. Prin această opţiune, se apelează un program ce rezolvă o ecuaţie diferenţială simplă de forma:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = -y$$
$$y(0) = 1$$

pe domeniul $[0,\ t_{max}]$. Programul necesită şi introducerea erorii admisibile şi numărului maxim de iteraţii necesar rezolvării ecuaţiei neliniare la fiecare pas al metodei implicite.

Programul afişează:

- eroarea locală pentru primul pas de calcul pentru metodele explicită şi implicită;
- eroarea globală pentru metodele explicită şi implicită;
- pentru metoda implicită se afişează numărul maxim de iteraţii necesare atingerii preciziei dorite.

Se va alege $t_{max} = 4\tau$ şi se va rezolva ecuaţia pentru diferiţi paşi de timp $h = 2\tau$, $h = \tau$, $h = \tau/10$, $h = \tau/100$, $h = \tau/1000$.

Se vor reprezenta grafic pe hârtie milimetrică:

- variaţia erorii locale şi globale în funcţie de h pentru metoda explicită;
- variaţia erorii locale şi globale în funcţie de h pentru metoda implicită, pentru o eroare impusă constantă de rezolvare a ecuaţiei neliniare a unui pas.
- variaţia numărului mediu de iteraţii necesar metodei implicite în funcţie de eroarea impusă rezolvării ecuaţiei neliniare, pentru $h = \tau/100$.;

Se vor compara cele două metode din punct de vedere al efortului de calcul necesar obţinerii unei erori globale impuse.

10.6.3 Rezolvarea unei ecuaţii diferenţiale caracteristice unui circuit electric de ordinul I

Se selectează opţiunea *Analiză ecuaţie*. În acest fel se apelează un program ce rezolvă o ecuaţie de forma:

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= -y + f(t) \\ y(0) &= 0 \end{aligned}$$

unde:

$$f(t) = (h(t) - h(t - t_0))A\exp(bt)\cos(\omega t + \varphi),$$

iar $h(t)$ reprezintă funcţia treaptă unitate. Domeniul de integrare este $[0, t_{max}]$, iar pasul de integrare este $h = t_{max}/100$.

Se introduc de la consolă următorii parametri:

- t_{max} - intervalul de integrare;
- t_0 - durata excitaţiei;
- A - amplitudinea excitaţiei;
- b - constanta de relaxare a excitaţiei;
- ω - pulsaţia excitaţiei;
- φ - faza iniţială a excitaţiei.

Se observă că:

- pentru $b = 0, \omega = 0$ şi $\varphi = 0$, excitaţia este un impuls dreptunghiular de durată t_0, iar dacă $t_0 \geq t_{max}$ excitaţia este constantă;
- pentru A, b nenuli, $\omega = 0$ şi $\varphi = 0$, excitaţia este un impuls cu variaţie exponenţială;
- pentru $b = 0$, excitaţia este armonică;
- pentru $b < 0$, excitaţia este oscilatorie amortizată;
- pentru $b > 0$, excitaţia este oscilatorie amplificată.

Se vor analiza toate aceste cazuri şi se vor comenta rezultatele obţinute.

10.6.4 Implementarea algoritmului într-un limbaj de programare şi testarea rutinei.

Se va implementa o procedură proprie de rezvoltare a unei ecuaţii diferenţiale, prin metoda Euler explicită. Se va scrie pseudocodul şi se va implementa un program ce apelează procedura de la punctul anterior şi rezolvă ecuaţia:

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= -y + \sin(\omega t) \\ y(0) &= y_0. \end{aligned}$$

Integrarea se face pe intervalul $0 < t < t_{max}$. Acest program va permite introducerea de la consolă a pulsaţiei ω, a condiţiei iniţiale $y(0)$, a limitei superioare a intervalului de integrare şi a pasului h. Soluţia va fi afişată alfanumeric pe ecran.

10.6.5 Căutare de informaţii pe Internet

Se vor căuta pe Internet informaţii (coduri) legate de rezolvarea ecuaţiilor diferenţiale ordinare. Cuvinte cheie recomandate: *ordinary differential equations*, *Euler's method*.

10.7 Probleme şi întrebări

1. Poate fi scăzut oricât de mult pasul de integrare în vederea micşorării erorilor? Justificaţi răspunsul.

2. Cum este influenţată precizia metodei de precizia sistemului de calcul?

3. Scrieţi o ecuaţie de forma:

$$\begin{aligned} \frac{\mathrm{d}y}{\mathrm{d}t} &= f(t,y) \\ y(0) &= y_0 \end{aligned}$$

   pentru a descrie variaţia curentului printr-o bobină cu inductivitate L şi rezistenţă R ce este conectată la o sursă de t.e.m. continuă de valoare E. Utilizaţi opţiunea *Rezolvare ecuaţie test* pentru a o rezolva.

4. Daţi exemple de categorii de circuite electrice ce se pot analiza cu metoda Euler.

5. Generaţi un algoritm care foloseşte metoda Euler pentru rezolvarea unor sisteme de ecuaţii diferenţiale de ordinul I.

6. Generaţi un algoritm care modifică automat pasul de integrare, pentru a menţine eroarea globală la o valoare impusă cu un efort minim de calcul.

7. Scrieţi un algoritm pentru calculul integralelor definite ale funcţiilor aplicând metoda Euler.

8. Analizaţi deosebirea dintre stabilitatea unei ecuaţii diferenţiale şi stabilitatea ei numerică.

9. Îmbunătăţiţi algoritmul Euler considerând mai mulţi termeni din seria Taylor a funcţiei $y(t)$.

10. Scrieţi o variantă a algoritmului `euler_imp` în care rezolvarea ecuaţiei neliniare să se facă cu metoda Newton.

11. Descrieţi rezolvarea numerică a unui sistem de ecuaţii diferenţiale ordinare. Cum poate fi aplicat acest algoritm în analiza numerică a circuitelor electrice neliniare în regim tranzitoriu? Arătaţi ca la fiecare pas de integrare în timp trebuie rezolvat

un circuit electric liniar, pentru care se poate aplica algoritmul din Lucrarea nr. 5. Cum se obtin circuitele companion, care sunt echivalente elementelor reactive: L, C?

12. Rezolvaţi numeric diferite ecuaţii de stare, liniare sau neliniare în mediul MATLAB/SCILAB.

13. Identificaţi în biblioteca matematică descrisă în [2] funcţii avansate pentru integrarea numerică a ecuaţiilor diferenţiale ordinare (Runge-Kutta, extrapolare Richardson, predictor-corector, pentru ecuaţii stiff). Ce aduc nou aceste funcţii, faţă de cele analizate în lucrare?

Leonhard Euler (1707, Elveţia - 1783, Rusia)

http://www-gap.dcs.st-and.ac.uk/ history/Mathematicians/Euler.html

Bibliografie şi webografie

[1] D. Ioan, I. Munteanu, B. Ionescu, M. Popescu, R. Popa, M. Lăzărescu şi G. Ciuprina. *Metode numerice în ingineria electrică*. MATRIX ROM, Bucureşti, România, a doua ed., 1998.

[2] W.H. Press, S.A. Teukolsky, W.T. Vetterling şi B.P. Flannery. *Numerical Recipes in C, The Art of Scientific Computing, Second Edition*. Cambridge University Press, 1992, `http://www.nrbook.com/a/bookcpdf.php`.

ISBN 978-606-23-0077-7


ISBN 978-606-23-00777-7