METODE NUMERICE IN INGINERIA ELECTRICA

Notite de curs: 2019 – 2020, semestrul 1

Conf. Dr. Ing. Mihai Iulian REBICAN

mihai.rebican@upb.ro

Curs: joi, 08:00 - 10:00, EA004

Consultatii: luni 10:00 - 11:00; joi 10-11; EC205 (IE, hol EB, etaj 2)

Universitatea Politehnica Bucuresti

Facultatea de Inginerie Electrica

Bucuresti, 2019 – 2020

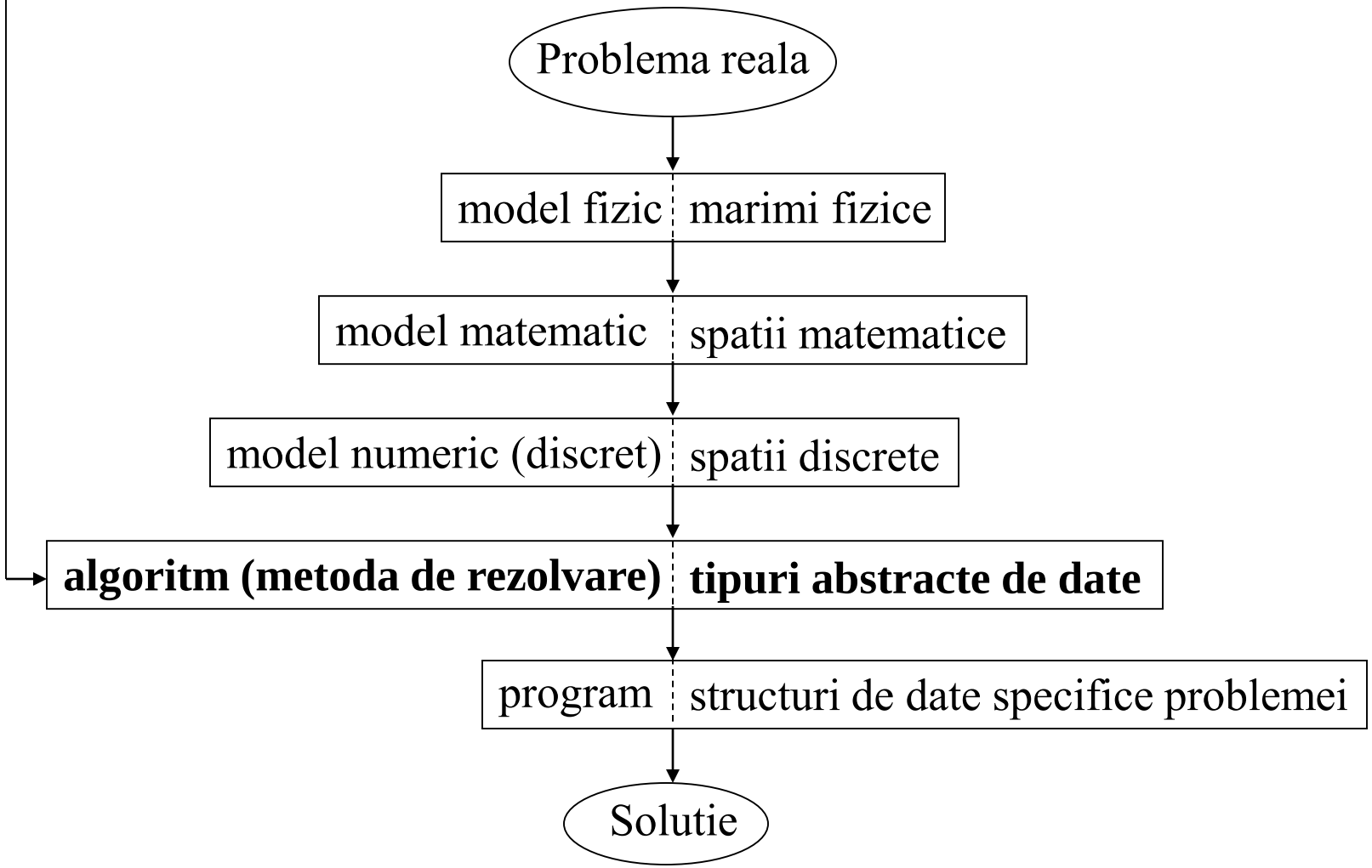
- **Curs** - 2 ore/sapt. - joi, 8-10, EA004
  - Conf. dr. ing. Mihai REBICAN
- **Laborator** - 2 ore/sapt. - miercuri, 8-18, EB214-215
  - Conf. dr. ing. Mihai REBICAN
  - As. ing. Mihai POPESCU
  - ȘL. dr. ing. Sorin LUP
- **Laborator** (activitate semestriala) - 50%
  - Activitate (15%)
  - Teste intrebari si aplicatii numerice (20%)
  - Teste implementare in C/Matlab (15%)
- **Examen scris** in sesiunea de iarna - 50%
  - subiecte aplicatii numerice si pseudocod (50%)

Bibliografie

- Pagina web curs MN – reguli de notare, altele http://mn.lmn.pub.ro
- Indrumar de laborator MN:

  M. Rebican, G. Ciuprina, D. Ioan http://mn.lmn.pub.ro/indrumar/IndrumarMN_Printech2013.pdf/
- D. Ioan - *Metode numerice in ingineria electrica*, Ed. Matrix Rom, Bucuresti, 1998
- W.H. Press - *Numerical recipes in C* http://www.nrbook.com/a/bookcpdf.php

Obiectul cursului - descrierea metodelor prin care se pot rezolva cu ajutorul calculatorului probleme de inginerie electrica formulate corect d.p.d.v. matematic



Inginerie electrica asistata de calculator

Computer aided engineering - CAE

Inginerie electrica

Metode numerice in ingineria electrica

Matematica

Stiinta calculatoarelor

Analiza numerica

Programare

- Formularea corecta a problemei d.p.d.v matematic: unicitatea solutiei
- Analiza erorilor

- Algoritmi numerici

Tipuri de probleme in ingineria electrica

- Analiza circuitelor electrice rezistive liniare
- Analiza circuitelor electrice rezistive neliniare
- Analiza circuitelor electrice in regim tranzitoriu
- Analiza campului electromagnetic in regim stationar
- Analiza campului electromagnetic in regim cuasistationar
- Analiza campului electromagnetic in regim general

Continutul cursului

1. Algoritmi si structuri de date
2. Erori in calcule numerice
3. Metoda directa Gauss de rezolvare a sistemelor de ecuatii liniare
4. Metode iterative de rezolvare a sistemelor de ecuatii liniare
5. Analiza numerica a circuitelor electrice in regim permanent
6. Interpolarea polinomiala a functiilor reale
7. Derivarea numerica a functiilor reale
8. Integrarea numerica a functiilor reale
9. Metode iterative de rezolvare a ecuatiilor neliniare

1. Algoritmi si structuri de date

Algoritm – metoda de rezolvare a unei probleme bazata pe descompunerea sa in etape simple, elementare, susceptibile de a fi implementate pe un calculator.

Pseudolimbajul (pseudocodul) – metoda de descriere si reprezentare a algoritmilor. *Sintaxa pseudocodului* nu este stricta si cuprinde cuvinte cheie din limba romana.

Liniile pseudolimbajului sunt de doua feluri:

- *declaratia*, linie care descrie datele;
- *instructiunea*, linie care descrie actiuni.

Variabila – zona de memorie identificata prin:

- *nume* – adresa zonei de memorie unde se afla variabila;
- *valoare* – continutul zonei de memorie;
- *tip* – in functie de acesta se interpreteaza valoarea.

Sintaxa declaratiilor

Tipuri de variabile:

1. *Simple* (*fundamentale*): **tip** *nume_var*

- logic: **logic** *a*, *b*
- intreg: **intreg** *N*
- real: **real** *m*
- caracter: **caracter** *c*

2. *Agregate*

- tablou: **tablou**
- inregistrare: **inregistrare**

- **Tablou** – multime de date de acelasi tip
- Declaratia tabloului:

  tablou tip_simplu *nume_tablou* [*dimensiune*]

  Ex.: **tablou real** *V* [10]; tablou de 10 elemente reale

  intreg *N*

  tablou real *W* [*N*]; alocare dinamica de memorie
- Referire la tablou – prin indice

  Ex.: *V*(1), *V*(2), …

  V_1, V_2, …

- **Inregistrare** – multime de date de tipuri diferite
- Declaratia inregistrarii:

  inregistrare nume_inregistrare

  tip_simplu_1 *nume_camp_1*

  tip_simplu_2 *nume_camp_2*

  Ex.: **inregistrare** punct

  logic *cartezian*

  real *x*

  real *y*
- Referire la inregistrare: *nume_inregistrare.nume_camp*

  Ex.: *punct.x*

  punct.cartezian

Sintaxa instructiunilor

Tipuri de instructiuni:

1. Simple	***2. Structurate***
• de intrare	• secventa
• de iesire	• decizia
• de atribuire	• ciclul
	• rutina

Instructiuni simple

Instructiune de intrare: **citeste** *nume_variabila*

Ex.: **citeste** *N*; numar de noduri

citeste *a*, *b*

Instructiune de iesire: **scrie** *nume_variabila*

Ex.: **scrie** *N*

Instructiune de atribuire: *nume_variabila = expresie*

Expresia – amestec de operanzi si operatori:

- logica (rezultatul evaluarii este de tip logic), cuprinde:
  - operatori logici: **si**, **sau**, **nu**, se aplica operanzilor logici
  - operatori de relatie: $<$, $>$, $\leq$, $\geq$, $==$, se aplica operanzilor aritmetici

Ex.: **logic** l, p, q

real a, b

$l = p$ **sau** q

$l = (a < b)$

$l = (a = b)$

- aritmetica (rezultatul evaluarii este de tip aritmetic), cuprinde:
  - operatori aritmetici: +, -, /, x, √, sin, cos, se aplica operanzilor aritmetici

Ex.: **real** a, b, m

$m = a + b$

$m = m + 1$

$m = \sqrt{m}$

$m = a / b$

Instructiuni structurate

Secventa: multime de instructiuni scrise indentat una sub alta

Ex.: $a = b - 3$

$m = a + b$

Decizia: - fara alternativa

daca conditie [**atunci**] ; conditie – de tip logic

secventa

- cu alternativa

daca conditie [**atunci**]

secventa 1

altfel

secventa 2

Ex.: modulul unui numar real, |x| = x, daca x ≥ 0

-x, daca x < 0

real *x*, *modul*

citeste *x*

daca x ≥ 0 **atunci**

modul = *x*

altfel

modul = -*x*

scrie *modul*

Ciclul: - se foloseste cand avem de repetat actiuni

- cu test initial:

cat timp conditie [**repeta**]

secventa

- cu test final: secventa se executa cel putin o data

repeta

secventa

pana cand conditie; in "C": **cat timp** !conditie

- cu contor: secventa se executa de *n* ori

pentru contor = val_initiala, val_finala[, pas] [**repeta**]

secventa

Ex.: $s = \sum_{i=1}^{n} x_i$

intreg *n*, *i*; dimensiune si contor

tablou real *x*[*n*]

real s

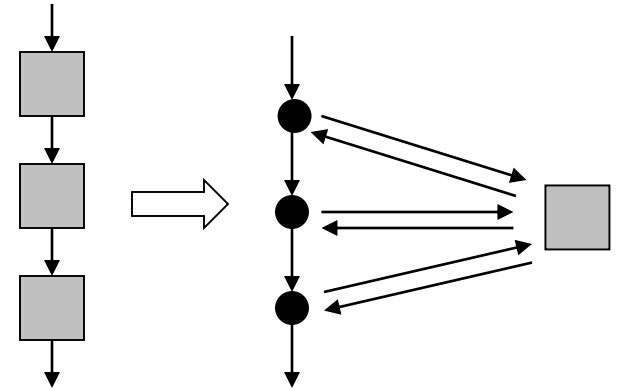
$s = 0$

pentru $i = 1, n$ [**repeta**]

$s = s + x_i$

Rutina:

- procedura;
- functia.



Definitia procedurii:

procedura nume_procedura (argumente formale de intrare/iesire)

; comentarii

declaratii

instructiuni

retur

Apelul procedurii:

nume_procedura (argumente actuale de intrare/iesire)

Obs.: numarul, tipul, ordinea argumentelor (parametrilor) actuale trebuie sa fie identice cu cele ale argumentelor formale!

Ex.: Determinarea minimului, maximului a 2 numere reale

```
; definitia procedurii
procedura MinMax (a, b, m, M)
; m=min(a,b), M=max(a,b)
real a, b; argumente de intrare
real m, M; argumente de iesire
daca a < b atunci
        m = a
        M = b
altfel
        m = b
        M = a
retur
```

```
; programul principal
real x, y
real min, max
citeste x, y
MinMax (x, y, min, max)
scrie min, max
stop
```

Obs.: "C": **void** function

Definitia functiei:

functie nume_functie (argumente formale de intrare)

; comentarii

declaratii

instructiuni

intoarce valoare

Apelul functiei:

nume_variabila = nume_functie (argumente actuale de intrare)

Obs.: - numarul, tipul, ordinea argumentelor actuale trebuie sa fie identice cu cele ale argumentelor formale!

- functia intoarce doar un singur parametru de iesire

- functia este asemenea functiei din matematica

Ex.: Determinarea modulului a unui numar real

; definitia functiei
functie modul (*a*)
real *a*; argument de intrare
real *rez*; argument de iesire
daca $a \geq 0$ **atunci**
    rez = *a*
altfel
    rez = **-***a*
intoarce *rez*

; programul principal
real *x*, *m*
citeste *x*
m = modul (*x*)
scrie *m*
stop

Obs.: "C": **float** function

Ex.: produsul scalar a 2 vectori

$$p = \sum_{i=1}^{n} x_i \cdot y_i$$

Pseudocod:

intreg *n, i;*

real *p*

tablou real *x*[*n*], *y*[n]

citeste *n*

pentru *i=1, n*

    citeste x_i, y_i

$p = 0$

pentru *i=1, n*

    $p=p+ x_i y_i$

scrie *p*

stop

"C":

```
#include<stdio.h>
int main()
{
int n,i;
float p,x[10],y[10];
printf("n=");
scanf("%d",&n);
for(i=1;i<=n;i++)
        {
        printf("x[%d]=",i);
        scanf("%f",&x[i]);
        }
for(i=1;i<=n;i++)
        {
        printf("y[%d]=",i);
        scanf("%f",&y[i]);
        }
p=0;
for(i=1;i<=n;i++)
        {
        p=p+x[i]*y[i];
        }
printf("p=%5.2f\n",p);
}
```

Evaluarea algoritmilor

Algoritmul: sa fie simplu; se analizeaza d.p.d.v. al timpului de calcul, necesarului de memorie, acuratetii solutiei.

Ordinul de complexitate d.p.d.v. al timpului de calcul = relatia dintre timpul de calcul exprimat in numarul de operatii elementare (+, -, *, /, …) si dimensiunea problemei (n).

timp = $c*n$ – algoritm liniar, de ordinul 1: $T = O(n)$, produsul scalar

timp = $c*n^2$ – algoritm patratic, de ordinul 2: $T = O(n^2)$, adunarea matricelor

c – constanta, depinde de sistemul de calcul

Algoritm de ordinul 4 – de evitat!!!

Necesarul de memorie = numarul de locatii elementare utilizate (numere reale).

real a; $M = O(1)$

tablou real $v[n]$; $M = O(n)$;

tablou real $c[n][n]$; $M = O(n^2)$;

$s = x \cdot y$ - $T = O(1)$, ordinul 0

pentru $i = 1, n$

  $s = s + x_i$ - $T = O(n)$, ordinul 1

pentru $i = 1, n$

  pentru $j = 1, n$

    $c_{ij} = x_{ij} + y_{ij}$ - $T = O(n^2)$, ordinul 2

pentru $i = 1, n$

  pentru $j = 1, n$

    pentru $k = 1, n$

      $c_{ik} = x_{ij} + y_{jk}$ - $T = O(n^3)$, ordinul 3

Ex.: produsul scalar a 2 vectori

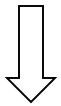
$T = O(2n)$

$M = O(2n+1)$

Ex.: **pentru** $i = 1, n$ sin, cos – operatii elementare

pentru $j = 1, n$ $T = O(2n^2)$, $M = O(n^2+2)$

$s_{ij} = \sin(a*i) + \cos(b*j)$



optimizare

pentru $i = 1, n$

$x_i = \sin(a*i)$

$y_i = \cos(b*i)$

pentru $i = 1, n$

pentru $j = 1, n$ $T = O(2n)$, $M = O(n^2+2n+2)$

$s_{ij} = x_i + y_j$

2. Erori in calcule numerice

Solutia unui algoritm implementat pe un sistem de calcul este afectata de **erori numerice**.

Tipuri de erori:

- erori de rotunjire, datorate reprezentarii finite a numerelor reale pe un sistem de calcul;
- erori inerente, datorate datelor de intrare, cum ar fi date obtinute pe cale experimentala;
- erori de trunchiere, datorate aproximarii finite a unor procese matematice teoretic infinite.

Evaluarea cantitativa a erorilor

x – solutia exacta, $\bar{x}$ – solutia numerica

Eroarea absoluta: $e_x = \bar{x} - x$

Marginea erorii absolute: $|e_x| \le a_x \Rightarrow x \in [\bar{x} - a_x, \bar{x} + a_x] \Rightarrow x = \bar{x} \pm a_x$

Eroarea relativa: $\varepsilon_x = \frac{e_x}{x} = \frac{x - \bar{x}}{x} \approx \frac{e_x}{\bar{x}}$

Marginea erorii relative: $|\varepsilon_x| \le r_x \Rightarrow x \in [\bar{x} - r_x, \bar{x} + r_x] \Rightarrow x = \bar{x} \pm r_x[\%]$

Ex.: Marginea erorii relative pentru numarul π

$x = 3.1415...;\ \bar{x} = 3.14$ (2 cifre semnificative)

$e_x = \bar{x} - x = 3.14 - 3.1415... = -0.0015... \Rightarrow$

$a_x = 0.0016 \Rightarrow \pi = 3.14 \pm 0.0016$

$\varepsilon_x = \frac{e_x}{x} = \frac{-0.0015...}{3.1415...} \Rightarrow r_x = 0.0006 = 0.06\% \Rightarrow \pi = 3.14 \pm 0.06\%$

Erori de rotunjire

Marginea erorii relative de rotunjire: $|\varepsilon_x| \le r_x = 10^{-n+1}$, unde n este numarul de cifre semnificative ale calculatorului

Ex.: $n = 6 \Rightarrow r_x = 10^{-5} = 0.001\%$

Pseudocod

; calculeaza eroarea relativa de rotunjire: *eps*

real *eps*

eps=1

repeta

eps=*eps*/2

pana cand (1 + *eps* = 1)

scrie *eps*

Obs.: *eps* - zeroul masinii (cel mai mare numar real care adunat la unitate nu schimba rezultatul).

Erori inerente

x (date de intrare) $\rightarrow$ ALGORITM $\rightarrow$ y (solutia)

Daca $\varepsilon_x \geq \varepsilon_y$, atunci algoritmul este stabil dpdv numeric

Daca $\varepsilon_x << \varepsilon_y$, atunci algoritmul prezinta instabilitati numerice.

Δx foarte mic $\rightarrow$ Δy mare

- adunarea: $y = x_1 + x_2$; $r_y = \left|\frac{x_1}{x_1 + x_2}\right| r_{x_1} + \left|\frac{x_2}{x_1 + x_2}\right| r_{x_2}$
- scaderea: $y = x_1 - x_2$; $r_y = \left|\frac{x_1}{x_1 - x_2}\right| r_{x_1} + \left|\frac{x_2}{x_1 - x_2}\right| r_{x_2}$
- inmultirea: $y = x_1 x_2$; $r_y = r_{x_1} + r_{x_2}$
- impartirea: $y = x_1 / x_2$; $r_y = r_{x_1} + r_{x_2}$

Operatia de scadere a doua numere foarte apropiate este instabila numeric datorita fenomenului de anulare prin scadere

Ex.: $x_1 = 2.23 \pm 0.2\%; x_2 = 2.22 \pm 0.2\% \Rightarrow$

$y = x_1 + x_2 = 4.45 \pm 0.2\%; y = x_1 - x_2 = 0.01 \pm 89\%$

Erori de trunchiere

Eroarea de trunchiere este de ordinul primului termen neglijat in procesul matematic.

Dezvoltarea in serie Taylor a functiei y = sin x este:

$$y = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{(2k-1)!} x^{2k-1}$$

Seria se trunchiaza dupa termenul de rang *n*:

$$r_y = \frac{|x|^{2n+1}}{(2n+1)!}$$

Pe un sistem de calcul, prin adunarea unui numar infinit de termeni ai unei serii, eroarea de trunchiere se satureaza la zeroul masinii!

3. Metoda directa Gauss de rezolvare a sistemelor de ecuatii liniare

Metoda directa: solutia sistemului de ecuatii se obtine dupa un numar finit de pasi.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \Leftrightarrow \mathrm{Ax} = \mathrm{b}$$

A – matricea coeficientilor ($n \times n$, $\det(A) \neq 0$)

b – vectorul termenilor liberi ($n \times 1$)

x – vectorul necunoscutelor ($n \times 1$)

Metoda Gauss:

- etapa de eliminare: A devine superior triunghiulara;
- etapa de retrosubstitutie (substitutie regresiva): determinarea propriu-zisa a necunoscutelor.

Etapa de eliminare (pentru $n = 3$)

$$\begin{matrix}(L_1)\\(L_2)\\(L_3)\end{matrix}\begin{cases}\boldsymbol{a_{11}}x_1 + a_{12}x_2 + a_{13}x_3 = b_1\\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2\\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3\end{cases} \quad (L'_2 = L_2 - \frac{a_{21}}{a_{11}} L_1) \Rightarrow$$

$$\begin{matrix}(L_1)\\(L'_2)\\(L_3)\end{matrix}\begin{cases}a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1\\ a'_{22}\,x_2 + a'_{23}\,x_3 = b'_2\\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3\end{cases} \quad (L'_3 = L_3 - \frac{a_{31}}{a_{11}} L_1) \Rightarrow$$

$$\begin{matrix}(L_1)\\(L'_2)\\(L'_3)\end{matrix}\begin{cases}a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1\\ \boldsymbol{a'_{22}}\,x_2 + a'_{23}\,x_3 = b'_2\\ a'_{32}\,x_2 + a'_{33}\,x_3 = b'_3\end{cases} \quad (L''_3 = L'_3 - \frac{a'_{32}}{a'_{22}} L'_2) \Rightarrow$$

$$a'_{22} = a_{22} - \frac{a_{21}}{a_{11}} a_{12} \quad a'_{23} = a_{23} - \frac{a_{21}}{a_{11}} a_{13} \quad b'_2 = b_2 - \frac{a_{21}}{a_{11}} b_1$$

$$a'_{32} = a_{32} - \frac{a_{31}}{a_{11}} a_{12} \quad a'_{33} = a_{33} - \frac{a_{31}}{a_{11}} a_{13} \quad b'_3 = b_3 - \frac{a_{31}}{a_{11}} b_1$$

$$\begin{matrix}(L_1)\\(L'_2)\\(L''_3)\end{cases}\begin{matrix}a_{11}x_1+a_{12}x_2+a_{13}x_3=b_1\\ a'_{22}\,x_2+a'_{23}\,x_3=b'_2\\ \boldsymbol{a''_{33}}\,x_3=b''_3\end{matrix}\qquad \mathrm{A}=\begin{pmatrix}a_{11}&a_{12}&a_{13}\\0&a'_{22}&a'_{23}\\0&0&a''_{33}\end{pmatrix}$$

$$a''_{33}=a'_{33}-\frac{a'_{32}}{a'_{22}}a'_{23}\quad b''_3=b'_3-\frac{a'_{32}}{a'_{22}}b'_2\qquad a_{11},a'_{22},a''_{33}-\text{pivoti}$$

Etapa de retrosubstitutie

$$x_3=\frac{b''_3}{a''_{33}}\Rightarrow$$

$$x_2=\frac{b'_2-a'_{23}\,x_3}{a'_{22}}\Rightarrow$$

$$x_1=\frac{b_1-a_{12}x_2-a_{13}x_3}{a_{11}}$$

Pseudocod

procedura Gauss_fp(n, a, b, x)
; declaratii
intreg n, i, j, k
real s, p
tablou real $a(n, n)$, $b(n)$, $x(n)$
;eliminare
pentru $k = 1, n-1$; etapele eliminarii
    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 pivotului
            $a_{ij} = a_{ij} - a_{kj} \cdot p$
        $b_i = b_i - b_k \cdot p$
; retrosubstitutie
$x_n = b_n / a_{nn}$
pentru $i = n-1, 1, -1$
    $s = b_i$
    pentru $j = n, i+1, -1$
        $s = s - a_{ij} \cdot x_j$
    $x_i = s / a_{ii}$
retur

program principal
; declaratii
intreg n, i, j
tablou real $a(n, n)$, $b(n)$, $x(n)$
; introducere date intrare
citeste n ; dimensiunea problemei
pentru $i = 1, n$
    pentru $j = 1, n$
        citeste a_{ij} ; elementele matricei
    citeste b_i ; elementele vectorului termenilor liberi
; apelare procedura metoda de rezolvare
Gauss_fp(n, a, b, x)
; afisare date iesire
pentru $i = 1, n$
    scrie x_i ; elementele vectorului necunoscutelor
stop

- $T = O(2n^3/3)$ – algoritm de ordinul 3 dpdv al timpului de calcul
- $M = O(n^2+2n+2)$ – algoritm de ordinul 2 dpdv al necesarului de memorie
- nu exista eroare de metoda
- pe un sistem de calcul apar erori de rotunjire
- datele de intrare (A, b) introduc erori inerente
- A - matrice slab conditionata (coeficienti cu valori foarte mici si valori foarte mari) → erori de rotunjire importante

Eroarea solutiei numerice ($\bar{x}$):

- reziduul: $r = A\bar{x} - b$
- eroarea absoluta: $e = \bar{x} - x$
- eroarea relativa: $\varepsilon = e/x$

Determinarea valorii erorii se face de fapt prin norma sa:

- norma euclidiana: $\|e\| = \sqrt{\sum_{k=1}^{n} e_k^2}$
- norma Cebisev: $\|e\| = \max_{k=1,n} |e_k|$

Strategii de pivotare

- pivot nul → metoda Gauss esueaza!
- pivotare partiala: se permuta liniile, algoritm simplu
- pivotare totala: se permuta liniile si coloanele, algoritm complicat, timp de calcul ridicat
- pivotare diagonala: pentru pastrarea simetriei matricei A
- pentru erori de rotunjire minime, se permuta linia cu pivot si linia cu coeficientul cel mai mare in modul de sub pivotul nul

Ex.:

$$\begin{matrix}(L_1)\\(L_2)\\(L_3)\end{cases}\begin{matrix}-2x+y-z=-2\\4x-3y+4z=5\\-6x+5y-8z=-9\end{matrix}\qquad (L'_2 = L_2 - \frac{4}{-2}L_1)\Rightarrow$$

$$\begin{matrix}(L_1)\\(L'_2)\\(L_3)\end{cases}\begin{matrix}-2x+y-z=-2\\-y+2z=1\\-6x+5y-8z=-9\end{matrix}\qquad (L'_3 = L_3 - \frac{-6}{-2}L_1)\Rightarrow$$

$$\begin{matrix}(L_1)\\(L'_2)\\(L'_3)\end{cases}\begin{matrix}-2x+y-z=-2\\-y+2z=1\\2y-5z=-3\end{matrix}\qquad (L''_3 = L'_3 - \frac{2}{-1}L'_2)\Rightarrow$$

$$\begin{matrix}(L_1)\\(L'_2)\\(L''_3)\end{cases}\begin{matrix}-2x+y-z=-2\\-y+2z=1\\-z=-1\end{matrix}\Rightarrow$$

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

4. Metode iterative de rezolvare a sistemelor de ecuatii liniare

Metoda iterativa: solutia sistemului de ecuatii se obtine prin generarea unui sir de solutii care tinde catre solutia exacta.

- proprietate de autocorectie a erorii: solutia numerica poate fi mai precisa decat solutia metodei directe Gauss;
- algoritm hibrid: se aplica metoda Gauss si se continua cu o metoda iterativa pentru rafinarea solutiei.

$$Ax = b$$

$$x^{(0)} \to x^{(1)} \to x^{(2)} \to \ldots \to x^{(k)} \to \ldots \qquad \lim_{k\to\infty} x^{(k)} = x$$

$x^{(k)} = F(x^{(k-1)})$ - formula de recurenta; F – aplicatie cu punct fix

$$Ax = b \Leftrightarrow x = F(x) \text{ atunci } A = B - C \Rightarrow (B - C)x = b \Rightarrow$$

$$x = B^{-1}Cx + B^{-1}b \Rightarrow x = Mx + u,\ M - \text{matrice de iteratie}$$

$$x^{(k)} = Mx^{(k-1)} + u \text{ - formula de recurenta}$$

- raza de convergenta: $\rho(M) = \max_{k=1,n} |\lambda_k|$, unde $\det(M - \lambda I) = 0$
- procesul iterativ este convergent daca $\rho(M) < 1$
- pentru orice matrice: $\rho(M) \leq \|M\|$

  $\|M\| < 1 \Rightarrow \rho(M) < 1 \Rightarrow$ metoda iterativa este convergenta!

Metode iterative:

- metoda deplasarilor simultane (Jacobi)
- metoda deplasarilor succesive (Gauss-Seidel);
- metoda suprarelaxarilor succesive (Frankel-Young);
- metoda directiilor alternante (Peaceman-Rachford);
- metoda iteratiilor bloc.

Metoda Jacobi

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}=\begin{pmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{pmatrix}+\begin{pmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{pmatrix}+\begin{pmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

$$A=L+D+U=B-C \Rightarrow B=D; \quad C=-(L+U)$$

$$M=B^{-1}C=-D^{-1}(L+U); \quad u=B^{-1}b=D^{-1}b$$

$$x^{(k)}=Mx^{(k-1)}+u=D^{-1}\left(b-(L+U)x^{(k-1)}\right)$$ - formula de recurenta

$$(L_i): a_{i1}x_1^{(k-1)}+a_{i2}x_2^{(k-1)}+\ldots+a_{ii-1}x_{i-1}^{(k-1)}+a_{ii}x_i^{(k)}+a_{ii+1}x_{i+1}^{(k-1)}+\ldots+a_{in}x_n^{(k-1)}=b_i$$

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

- solutia la pasul curent se determina din solutia de la pasul anterior
- sunt necesari doi vectori solutie

Metoda Gauss-Seidel

$$A = L + D + U = B - C \Rightarrow \quad B = L + D; \quad C = -U$$

$$M = B^{-1}C = -(L+D)^{-1}U; \quad u = B^{-1}b = (L+D)^{-1}b$$

$$x^{(k)} = Mx^{(k-1)} + u = (L+D)^{-1}\left(b - Ux^{(k-1)}\right)$$ - formula de recurenta

$$(L_i): a_{i1}x_1^{(k)} + a_{i2}x_2^{(k)} + \ldots + a_{ii-1}x_{i-1}^{(k)} + a_{ii}x_i^{(k)} + a_{ii+1}x_{i+1}^{(k-1)} + \ldots + a_{in}x_n^{(k-1)} = b_i$$

$$x_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}}{a_{ii}}, \; i = \overline{1,n}$$

- solutia la pasul curent se determina din solutia de la pasul anterior si componentele deja calculate ale solutiei curente;
- principiul Seidel: imediat ce o componenta a fost determinata, ea este folosita in calculele urmatoare, inlocuind valoarea veche care se pierde
- este necesar un singur vector solutie

- la metodele Jacobi si Gauss-Seidel, o conditie suficienta pentru convergenta lor este ca matricea A sa fie diagonal dominanta

$$|a_{ii}| > \sum_{j=1,\, j\neq i}^{n} |a_{ij}|, i = 1, n$$

- conditiile de oprire ale procesului iterativ:
  - daca norma erorii Cauchy este mai mica decat o valoare impusa a erorii, ε:

$$e = \left\| \mathrm{x}^{(k)} - \mathrm{x}^{(k-1)} \right\| < \varepsilon$$

  - daca numarul maxim de iteratii este atins (proces divergent)
- eroarea de metoda este eroarea de trunchiere
- metoda Gauss-Seidel este mai rapid convergenta decat metoda Jacobi
- pentru matrici simetrice si pozitiv definite, metoda Gauss-Seidel este de aproximativ doua ori mai rapida decat metoda Jacobi.
- metodele iterative nu genereaza umpleri ale matricei, utile pentru rezolvarea sistemelor cu matrice rara

Pseudocod

procedura Jacobi(*n*, *a*, *b*, *x*, *nrit*, *eps*)
; declaratii
intreg *n*, *nrit*, *i*, *j*, *k*
real *s*, *err*
tablou real *a*(*n*, *n*), *b*(*n*), *x*(*n*), *xn*(*n*)
;initializari
$k = 0$
pentru $i = 1, n$
    $x_i = 0$
; iteratii
repeta
    $err = 0$
    pentru $i = 1, n$
        $s = b_i$
        pentru $j = 1, n$
            $s = s - a_{ij} x_j$
        $s = s + a_{ii} x_i$
        $xn_i = s / a_{ii}$
        $s = |xn_i - x_i|$

daca $err < s$ **atunci**

$err = s$

pentru $i = 1, n$

$x_i = xn_i$

$k = k + 1$

pana cand $(err < eps)$ sau $(k > nrit)$

retur

procedura Gauss-Seidel (*n*, *a*, *b*, *x*, *nrit*, *eps*)
; declaratii
intreg *n*, *nrit*, *i*, *j*, *k*
real *s*, *err*
tablou real *a*(*n*, *n*), *b*(*n*), *x*(*n*)
;initializari
$k = 0$
pentru $i = 1, n$
    $x_i = 0$
; iteratii
repeta
    $err = 0$
    pentru $i = 1, n$
        $s = b_i$
        pentru $j = 1, n$
            $s = s - a_{ij} x_j$
        $s = (s + a_{ii} x_i)/a_{ii}$
        $err = err + (s - x_i)^2$
        $x_i = s$
    $k = k + 1$
    $err = \text{sqrt}(err)$

– $T = O(mn^2)$, unde m este numarul de iteratii; daca $m < n$, algoritm de ordinul 2 dpdv al timpului de calcul

– $M = O(n^2+2n)$ – algoritm de ordinul 2 dpdv al necesarului de memorie

pana cand ($err < eps$) sau ($k > nrit$)
retur

program principal
; declaratii
intreg n, i, j, $nrit$
real eps
tablou real $a(n, n)$, $b(n)$, $x(n)$
; introducere date intrare
citeste $nrit$, eps ; numar maxim de iteratii si eroarea impusa
citeste n ; dimensiunea problemei
pentru $i = 1, n$
    pentru $j = 1, n$
        citeste a_{ij} ; elementele matricei
    citeste b_i ; elementele vectorului termenilor liberi
; apelare procedura metoda de rezolvare
Gauss_Seidel(n, a, b, x, $nrit$, eps)
; afisare date iesire
pentru $i = 1, n$
    scrie x_i ; elementele vectorului necunoscutelor
stop

Ex.: $x^{(0)} = y^{(0)} = z^{(0)} = 0$

$$\begin{cases} 3x - y + z = 3 \\ 3x - 6y + z = -2 \\ -x + 2y + 4z = 5 \end{cases} \qquad \begin{cases} |3| > |-1| + |1| \\ |-6| > |3| + |1| \Rightarrow \\ |4| > |-1| + |2| \end{cases}$$

matricea sistemului este diagonal dominanta

metoda Jacobi

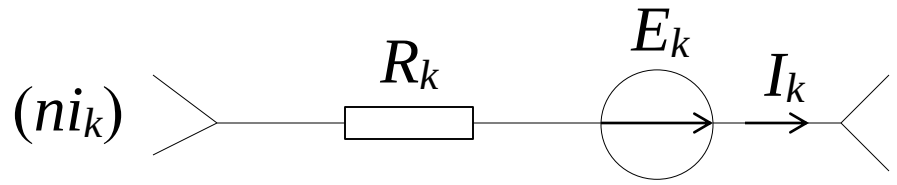
$$\begin{cases} x^{(1)} = \dfrac{3 + y^{(0)} - z^{(0)}}{3} = 1 \\ y^{(1)} = \dfrac{-2 - 3x^{(0)} - z^{(0)}}{-6} = \dfrac{1}{3} \\ z^{(1)} = \dfrac{5 + x^{(0)} - 2y^{(0)}}{4} = \dfrac{5}{4} \end{cases} \Rightarrow \begin{cases} x^{(2)} = \dfrac{3 + y^{(1)} - z^{(1)}}{3} = \dfrac{25}{36} \\ y^{(2)} = \dfrac{-2 - 3x^{(1)} - z^{(1)}}{-6} = \dfrac{25}{24} \\ z^{(2)} = \dfrac{5 + x^{(1)} - 2y^{(1)}}{4} = \dfrac{4}{3} \end{cases}$$

metoda Gauss-Seidel

$$\begin{cases} x^{(1)} = \dfrac{3 + y^{(0)} - z^{(0)}}{3} = 1 \\ y^{(1)} = \dfrac{-2 - 3x^{(1)} - z^{(0)}}{-6} = \dfrac{5}{6} \\ z^{(1)} = \dfrac{5 + x^{(1)} - 2y^{(1)}}{4} = \dfrac{13}{12} \end{cases} \Rightarrow \begin{cases} x^{(2)} = \dfrac{3 + y^{(1)} - z^{(1)}}{3} = \dfrac{11}{12} \\ y^{(2)} = \dfrac{-2 - 3x^{(2)} - z^{(1)}}{-6} = \dfrac{35}{36} \\ z^{(2)} = \dfrac{5 + x^{(2)} - 2y^{(2)}}{4} = \dfrac{143}{144} \end{cases}$$

5. Analiza numerica a circuitelor electrice in regim permanent

Circuit electric rezistiv liniar cu N noduri si L laturi:



$R_k \neq 0;\ k = 1, L$

$$U_k = R_k I_k - E_k$$

Metoda potentialelor la noduri: $\mathrm{G\,V = I_s}$

G – matricea conductantelor nodale $((N\text{-}1)\times(N\text{-}1))$

$\mathrm{I_s}$ – vectorul injectiilor de curent $((N\text{-}1)\times 1)$

V – vectorul potentialelor nodurilor $((N\text{-}1)\times 1)$

Contributiile laturii k a circuitului la matricea G si vectorul $\mathrm{I_s}$:

	Matricea G		Vectorul $\mathrm{I_s}$
	coloana ni_k	coloana nf_k	
linia ni_k	$1/R_k$	$-1/R_k$	$-E_k/R_k$
linia nf_k	$-1/R_k$	$1/R_k$	E_k/R_k

Pseudocod

program principal
; declaratii
intreg N, L, k, i, j
tablou intreg $ni(L)$, $nf(L)$
tablou real $R(L)$, $E(L)$, $U(L)$, $Icrt(L)$
tablou real $G(N, N)$, $Is(N)$, $V(N)$
real pc, pg
; PREPROCESARE
; Introducere date intrare
citeste N, L ; numar de noduri, numar de laturi
pentru $k = 1, L$
    citeste ni_k, nf_k ; nodul initial, nodul final ale laturii k
    citeste R_k, E_k ; rezistenta, tensiunea electromotoare ale laturii k
; Generarea automata a structurilor de date prin parcugerea laturilor circuitului
; initializare
pentru $i = 1, n$
    pentru $j = 1, n$
        $G_{ij} = 0$
    $Is_i = 0$

; parcurgere laturi
pentru $k = 1, L$
$n1 = ni_k$,
$n2 = nf_k$,
$G_{n1n1} = G_{n1n1} + 1/R_k$
$G_{n2n2} = G_{n2n2} + 1/R_k$
$G_{n1n2} = G_{n1n2} - 1/R_k$
$G_{n2n1} = G_{n2n1} - 1/R_k$
$Is_{n1} = Is_{n1} - E_k/R_k$
$Is_{n2} = Is_{n2} + E_k/R_k$
; PROCESARE
; rezolvarea sistem cu N-1 ecuatii
Gauss_fp(N-1, G, Is, V)
$V_N = 0$; potential nul al nodului de referinta
; POSTPROCESARE
; determinarea curentilor si tensiunilor laturilor, puterile consumata si generata
$pg = 0$; puterea generata
$pc = 0$; puterea consumata

; parcurgere laturi
pentru $k = 1, L$
    $n1 = ni_k$,
    $n2 = nf_k$,
    $U_k = V_{n1} - V_{n2}$
    $Icrt_k = (U_k + E_k)/R_k$
    $pg = pg + E_k\, Icrt_k$
    $pc = pc + R_k\, Icrt_k\, Icrt_k$
; afisare solutii
pentru $k = 1, L$
    scrie k, $Icrt_k$, U_k
scrie pc, pg
stop

- Matricea G este diagonal dominanta si rara, metodele iterative sunt mai eficiente.
- Erorile inerente si de rotunjire se propaga in procesul de calcul si pot genera instabilitati numerice importante.
- Daca matricea sistemului este slab conditionata, valorile puterilor consumata si generata pot sa nu fie egale pana la ultima zecimala.

6. Interpolarea polinomiala a functiilor reale

Fie functia $f:[a,b] \rightarrow$ IR, $f(x)=y$ reprezentata prin date (tabelar): se cunosc valorile functiei doar intr-o retea de puncte din domeniul de definitie, numite noduri.

x	x_0	x_1	...	x_n
y	y_0	y_1	…	y_n

← retea de discretizare cu n+1 noduri

Interpolarea reprezinta aproximarea functiei cu un polinom, astfel evaluarea functiei se reduce la operatii aritmetice elementare.

Problema fundamentala a interpolarii consta in determinarea unei functii $g:[a,b] \rightarrow$ IR de forma:

$$g(x) = \sum_{k=0}^{n} \left[c_k \varphi_k(x) \right],$$

care aproximeaza functia f, avand aceleasi valori in nodurile retelei de discretizare cu valorile functiei f:

$g(x_k) = f(x_k) = y_k, k = 0, n$ - conditii de interpolare

$\varphi_k(x)$, $k = 0, n$ – functii de baza, date de intrare ale problemei de interpolare.

Numarul de noduri ale retelei de discretizare = numarul de functii de baza = n+1.

Problema este bine formulata si are solutie unica daca functiile de baza sunt liniar independente si nodurile retelei de discretizare sunt distincte.

Practic, coeficientii c_k sunt necunoscutele problemei de interpolare.

Metode de interpolare globala

1. Metoda clasica de interpolare

Functiile de baza sunt alese astfel:

$$\varphi_0(x)=1,\ \varphi_1(x)=x,\varphi_2(x)=x^2,...,\varphi_k(x)=x^k,...,\varphi_n(x)=x^n$$

Functia de interpolare devine:

$$g(x)=\sum_{k=0}^{n}\left[c_k\varphi_k(x)\right]=\sum_{k=0}^{n}\left(c_k x^k\right)=c_0+c_1x+c_2x^2+...+c_nx^n$$

Gradul polinomului este cu 1 mai mic decat numarul de noduri:

n+1 noduri → polinom gradul n

Din conditiile de interpolare

$$g(x_0)=y_0,\ g(x_1)=y_1,\ g(x_2)=y_2,...,\ g(x_n)=y_n,$$

rezulta sistemul de ecuatii algebrice liniare (n+1 ecuatii, n+1 necunoscute):

$$\begin{cases} c_0 + c_1 x_0 + c_2 x_0^2 + \ldots + c_n x_0^n = y_0 \\ c_0 + c_1 x_1 + c_2 x_1^2 + \ldots + c_n x_1^n = y_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ c_0 + c_1 x_n + c_2 x_n^2 + \ldots + c_n x_n^n = y_n \\ c_0 + c_1 x_n + c_2 x_n^2 + \ldots + c_n x_n^n = y_n \end{cases} \Rightarrow \quad c_{0,}\, c_1, c_2, \ldots, c_n \Rightarrow g(x)$$

- matricea sistemului nu este diagonal dominanta, nu se aplica metode iterative.
- timpul de pregatire (determinare coeficienti c_k) este $2n^3/3$ (metoda directa Gauss), mult prea mare.
- timpul de evaluare este $2n$.
- erorile solutiei sunt mari, deoarece matricea sistemului este slab conditionata pentru valori mari ale gradului $n > 5$ (functiile de baza devin aproape liniar dependente).

2. Metoda Lagrange

Functiile de baza $\varphi_k(x) = l_k(x)$, sunt ortogonale cu proprietatile:

$$l_i(x_i) = 1, l_i(x_j) = 0, i \neq j,$$

reprezentand polinoamele Lagrange:

$$l_k(x) = a_k(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{k-1})(x - x_{k+1}) \cdot \ldots \cdot (x - x_n)$$

$$l_k(x) = a_k \prod_{\substack{i=0 \\ i \neq k}}^{n} (x - x_i)$$

Coeficientii a_k se determina astfel:

$$l_k(x_k) = 1 \Rightarrow a_k \prod_{\substack{i=0 \\ i \neq k}}^{n} (x_k - x_i) = 1 \Rightarrow a_k = \frac{1}{\prod_{\substack{i=0 \\ i \neq k}}^{n} (x_k - x_i)}$$

Astfel, polinoamele Lagrange au forma finala:

$$l_k(x) = \frac{\prod_{\substack{i=0 \\ i \neq k}}^{n} (x - x_i)}{\prod_{\substack{i=0 \\ i \neq k}}^{n} (x_k - x_i)}, k = 0, n$$

Functia de interpolare Lagrange este:

$$g(x) = \sum_{k=0}^{n} \left[c_k l_k(x) \right] = \sum_{k=0}^{n} \left[c_k \frac{\prod_{\substack{i=0 \\ i \neq k}}^{n} (x - x_i)}{\prod_{\substack{i=0 \\ i \neq k}}^{n} (x_k - x_i)} \right]$$

Din conditiile de interpolare $g(x_k) = y_k, k = 0, n$, rezulta sistemul:

$$\begin{cases} c_0 l_0(x_0) + c_1 l_1(x_0) + \ldots + c_n l_n(x_0) = y_0 \\ c_0 l_0(x_1) + c_1 l_1(x_1) + \ldots + c_n l_n(x_1) = y_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ c_0 l_0(x_n) + c_1 l_1(x_n) + \ldots + c_n l_n(x_n) = y_n \end{cases} \Rightarrow$$

$$\begin{cases} c_0 = y_0 \\ c_1 = y_1 \\ \ldots\ldots\ldots \\ c_n = y_n \end{cases} \Rightarrow c_k = y_k, k = 0, n$$

Functia de interpolare Lagrange are forma finala:

$$g(x) = \sum_{k=0}^{n} \left[y_k l_k(x) \right] = \sum_{k=0}^{n} \left[y_k \prod_{\substack{i=0 \\ i \neq k}}^{n} \left(\frac{x - x_i}{x_k - x_i} \right) \right]$$

Cazuri particulare:

- n=1 (2 noduri prin care trece o dreapta):

$$g(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0}$$

- n=2 (3 noduri prin care trece o parabola):

$$g(x) = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

- matricea sistemului este diagonala.
- timpul de evaluare este $4n^2$, si reprezinta timpul total (inclusiv pregatirea datelor).
- matricea sistemului este foarte bine conditionata (functiile de baza sunt ortogonale).

- pentru a reduce timpul de evaluare, care include si pregatirea datelor, functia de interpolare Lagrange se modifica astfel:

$$g(x)=\sum_{k=0}^{n}\left[y_k\prod_{\substack{i=0\\ i\neq k}}^{n}\left(\frac{x-x_i}{x_k-x_i}\right)\right]=\prod_{i=0}^{n}(x-x_i)\sum_{k=0}^{n}\left[\frac{1}{x-x_k}\frac{y_k}{\prod_{\substack{i=0\\ i\neq k}}^{n}(x_k-x_i)}\right]\Rightarrow$$

$$g(x)=\prod_{i=0}^{n}(x-x_i)\sum_{k=0}^{n}\left(\frac{p_k}{x-x_k}\right),$$

unde coeficientii p_k se pot determina inainte de evaluarea propriu-zisa (etapa de pregatire):

$$p_k=\prod_{\substack{i=0\\ i\neq k}}^{n}\left(\frac{y_k}{x_k-x_i}\right),k=0,n$$

- Pentru metoda Lagrange cu pregatire, timpul de pregatire este $2n^2$, iar timpul de evaluare este $5n$, ceea ce reprezinta un avantaj important daca se efectueaza foarte multe evaluari ale functiei de interpolare Lagrange.

3. Metoda Newton

Functiile de baza $\varphi_k(x)$ sunt alese astfel:

$$\begin{cases} \varphi_0(x) = 1 \\ \varphi_1(x) = x - x_0 \\ \varphi_2(x) = (x - x_0)(x - x_1) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \varphi_n(x) = (x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1}) \end{cases} \Rightarrow \varphi_k(x) = \prod_{i=0}^{k-1} (x - x_i), k = 0, n$$

Functia de interpolare Newton este:

$$g(x) = \sum_{k=0}^{n} [c_k \varphi_k(x)] = \sum_{k=0}^{n} \left[c_k \prod_{i=0}^{k-1} (x - x_i) \right] \Rightarrow$$

$$g(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots +$$
$$+ c_n(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1})$$

Din conditiile de interpolare $g(x_k) = y_k$, $k = 0, n$, rezulta sistemul cu o matrice de forma triunghiular inferioara, care se rezolva prin subtitutie progresiva:

$$\begin{cases} c_0 = y_0 \\ c_0 + c_1(x_1 - x_0) = y_1 \\ c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) = y_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ c_0 + c_1(x_n - x_0) + c_2(x_n - x_0)(x_n - x_1) + \ldots + c_2(x_n - x_0)(x_n - x_1)\cdot\ldots\cdot(x_n - x_{n-1}) = y_n \end{cases} \Rightarrow$$

$$c_0 = y_0 = f(x_0)$$

$$c_1 = \frac{y_1 - y_0}{x_1 - x_0} = f[x_0, x_1]$$

$$c_2 = \frac{y_2 - y_0 - \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = f[x_0, x_1, x_2]$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$c_n = \ldots = f[x_0, x_1, x_2, \ldots, x_n]$$

$$\Rightarrow c_k = f[x_0, x_1, \ldots, x_k], k = 0, n$$

Coeficientii c_k sunt diferentele divizate de ordinul k, $f[x_0, x_1, \ldots, x_k]$

Diferentele divizate de ordinul k, se determina recursiv, din diferentele divizate de ordinul k-1:

$$f[x_0, x_1, ..., x_k] = \frac{f[x_1, x_2, ..., x_k] - f[x_0, x_1, ..., x_{k-1}]}{x_k - x_0}$$

Astfel, diferentele divizate de ordinul 1 sunt:

$$f[x_0, x_1] = \frac{y_1 - y_0}{x_k - x_0}, \; f[x_1, x_2] = \frac{y_2 - y_1}{x_2 - x_1}, ...,$$

diferentele divizate de ordinul 2 sunt:

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}, \; f[x_1, x_2, x_3] = \frac{f[x_2, x_3] - f[x_1, x_2]}{x_3 - x_1}, ...,$$

iar in cele din urma, diferenta divizata de ordinul n este:

$$f[x_0, x_1, ..., x_n] = \frac{f[x_1, x_2, ..., x_n] - f[x_0, x_1, ..., x_{n-1}]}{x_n - x_0}.$$

Functia de interpolare Newton are forma finala:

$$g(x) = \sum_{k=0}^{n} \left[f[x_0, x_1, ..., x_k] \prod_{i=0}^{k-1} (x - x_i) \right] \Rightarrow$$

$$g(x) = y_0 + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + ... +$$
$$+ f[x_0, x_1, ..., x_n](x - x_0)(x - x_1) \cdot ... \cdot (x - x_{n-1})$$

- matricea sistemului este relativ bine conditionata.
- timpul de pregatire (determinare coeficienti c_k) este $3n^2/2$.
- timpul de evaluare este $2n$.
- permite marirea gradului polinomului de interpolare, prin adaugarea unui nod nou in reteaua de discretizare, cu reutilizarea coeficientilor de la gradul anterior, care nu se modifica, deci cu un efort minim de calcul.
- astfel, exista control automat asupra erorii de interpolare.

- Metode clasica, Lagrange, Newton sunt metode de interpolare globala, ce determina un singur polinom de grad n care sa treaca prin cele $n + 1$ puncte ale retelei de discretizare
- Deoarece polinomul de interpolare este unic ($n = 1$, exista o dreapta unica ce trece prin cele doua puncte; $n = 2$, exista o parabola unica ce trece prin cele trei puncte, …), cele trei metode rulate pe un calculator ideal, de precizie infinita (in care nu exista erori de rotunjire) ar furniza aceeasi solutie.
- Eroarea de interpolare este eroarea de trunchiere, care depinde invers proportional de gradul polinomului de interpolare, n, si direct proportional de pasul de discretizare, $h=x_k-x_{k-1}$=ct. (retea uniforma).
- Teoretic, eroare scade cu cresterea gradului polinomului de interpolare, dar pentru anumite functii, de exemplu functia Runge, s-a observat o comportare contrara pe o retea uniforma.

- Pentru functia Runge, cresterea numarului de noduri ale retelei de discretizare uniforme conduce la aparitia de oscilatii importante ale functiei de interpolare intre nodurile retelei, in special la capetele domeniului de definitie.
- Solutia este utilizarea unei retele neuniforme, ale carei noduri sunt chiar radacinile polinomului Cebisev, ceea ce reprezinta interpolarea neuniforma Cebisev.
- Interpolarea Cebisev nu se poate aplica functiilor definite tabelar, trebuie sa se cunoasca expresia analitica a functiei pentru evaluarea acesteia in radacinile polinomului Cebisev.

Pseudocod

functia interp_L (*n*, *x*, *y*, *xcrt*)
; evalueaza functia de interpolare Lagrange fara pregatire in punctul *xcrt*
intreg *n* ; gradul polinomului
tablou real *x*(*n*+1), y(*n*+1) ; reteaua de discretizare, indici de la 0 la n
real *xcrt* ; punctul in care se doreste evaluarea functiei de interpolare
real *ycrt* ; valoarea functiei de interpolare in *xcrt*
real *s*
ycrt = 0
pentru *k* = 0, *n*
    s = 1
    pentru *j* = 0, *n*
        daca $j \neq k$ **atunci**
            $s = s \cdot (xcrt - x_j)/(x_k - x_j)$
    $ycrt = ycrt + s \cdot y_k$
intoarce *ycrt*

procedura preg_L (n, x, y, p)
; pregateste datele pentru interpolarea Lagrange
intreg n ; gradul polinomului
tablou real $x(n+1)$, y(n+1) ; reteaua de discretizare, indici de la 0 la n
tablou real $p(n+1)$; coeficientii polinomului, date pregatite, indici de la 0 la n
pentru $k = 0, n$
    $p_k = y_k$
    pentru $j = 0, n$
        daca $j \neq k$ **atunci**
            $p_k = p_k /(x_k - x_j)$
retur

functia eval_L (*n*, *x*, *y*, *xcrt*, *p*)
; evalueaza functia de interpolare Lagrange cu pregatire in punctul *xcrt*
intreg *n* ; gradul polinomului
tablou real *x*(*n*+1), y(*n*+1) ; reteaua de interpolare, indici de la 0 la n
tablou real *p*(*n*+1) ; coeficientii polinomului, date pregatite, indici de la 0 la n
real *xcrt*, *ycrt*, *s*
s = 1
pentru *k* = 0, *n*
    s = *s*·(*xcrt* − x_k)
    daca *s* = 0 **atunci**
        intoarce y_k
ycrt = 0
pentru *k* = 0, *n*
    ycrt = *ycrt* + p_k /(*xcrt* − x_k)
ycrt = *s*·*ycrt*
intoarce *ycrt*

program principal
; declaratii
intreg n, k
real $xcrt$, $ycrt$
tablou real $x(n+1)$, $y(n+1)$
; introducere date intrare
citeste $xcrt$; punctul in care se doreste evaluarea functiei de interpolare
citeste n ; gradul polinomului
pentru $k = 0, n$
    citeste x_k, y_k ; nodurile retelei de discretizare
; apelare functie metoda de interpolare Lagrange fara pregatire
$ycrt$ = interp_L (n, x, y, $xcrt$)
; afisare date iesire
scrie $ycrt$; valoarea functiei de interpolare in $xcrt$
stop

Metode de interpolare locala

Metoda de interpolare liniara pe portiuni

Nu se determina un singur polinom pe intervalul [x_0, x_n] ca la metodele de interpolare globala.

Pe fiecare subinterval [x_k, x_{k+1}] se determina cate un polinom de gradul 1.

Astfel, intre doua noduri succesive, (x_k, y_k) si (x_{k+1}, y_{k+1}), graficul functiei este aproximat cu dreapta care uneste nodurile respective.

Graficul functiei $f(x)$ este aproximat cu linia poligonala ($g(x)$) care uneste toate nodurile retelei de discretizare.

Pentru $x \in [x_k, x_{k+1}]$:

$$g(x) = y = y_k + \frac{y_{k+1} - y_k}{x_{k+1} - x_k}(x - x_k)$$

7. Derivarea numerica a functiilor reale

Fie functia $f:[a,b] \rightarrow IR$, $f(x)=y$ reprezentata prin date (tabelar): se cunosc valorile functiei doar intr-o retea de puncte din domeniul de definitie, numite noduri.

x	x_0	x_1	...	x_n
y	y_0	y_1	…	y_n
y'	$y'_0=?$	$y'_1=?$	…	$y'_n=?$

← retea de discretizare cu $n+1$ noduri

Derivarea numerica se bazeaza pe interpolarea numerica!

Daca se determina functia de interpolare $g:[a,b] \rightarrow IR$ care trece prin nodurile retele de discretizare, atunci derivata numerica a functiei $f(x)$ se obtine prin derivarea functiei $g(x)$, care reprezinta un polinom de gradul n.

Pentru $n = 1$, functia de interpolare este un polinom de gradul 1:

$$g(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0},$$

iar derivata sa este o constanta pe intervalul $[x_0, x_1]$:

$$g'(x) = \frac{y_1 - y_0}{x_1 - x_0}$$

Astfel, aproximarea de ordinul unu a derivatei numerice este discontinua in nodurile retelei de discretizare:

$$g'(x_k) = y'_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k} \leftarrow \text{formula progresiva de ordinul 1}$$

$$g'(x_k) = y'_k = \frac{y_k - y_{k-1}}{x_k - x_{k-1}} \leftarrow \text{formula regresiva de ordinul 1}$$

Pentru $n = 2$, functia de interpolare este un polinom de gradul 2:

$$g(x) = y_0 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + y_1 \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + y_2 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)},$$

iar derivata sa este un polinom de gradul 1 pe diviziunea $[x_0,x_1,x_2]$:

$$g'(x) = y_0 \frac{x-x_1+x-x_2}{(x_0-x_1)(x_0-x_2)} + y_1 \frac{x-x_0+x-x_2}{(x_1-x_0)(x_1-x_2)} + y_2 \frac{x-x_0+x-x_1}{(x_2-x_0)(x_2-x_1)}$$

Pentru o retea de discretizare uniforma, $h=x_k-x_{k-1}$=ct., aproximarile de ordinul doi ale derivatei numerice sunt:

$$g'(x_{k-1}) = y'_{k-1} = \frac{-3y_{k-1}+4y_k-y_{k+1}}{2h} \quad \leftarrow \text{ formula progresiva de ordinul 2}$$

$$g'(x_k) = y'_k = \frac{y_{k+1}-y_{k-1}}{2h} \quad \leftarrow \text{ formula centrata de ordinul 2}$$

$$g'(x_{k+1}) = y'_{k+1} = \frac{y_{k-1}-4y_k+3y_{k+1}}{2h} \quad \leftarrow \text{ formula regresiva de ordinul 2}$$

- Pentru determinarea derivatei numerice in nodurile retelei de discretizare, se recomanda utilizarea:
  - formulei centrate de ordinul 2 in nodurile interioare
  - formulei progresiva de ordinul 1 sau 2 in primul nod
  - formulei regresiva de ordinul 1 sau 2 in ultimul nod.
- Pentru determinarea derivatei intr-un punct diferit de nodurile retelei de discretizare, se utilizeaza interpolarea numerica pe o retea de discretizare diferita (x_k, y'_k), $k=0,n$.
- Pentru calcul derivatelor de ordin superior (a doua derivata, a treia derivata, a patra derivata, ...), se utilizeaza aproximari de ordin mare.
- De exemplu, pentru determinarea numerica a celei de-a doua derivate, este necesar cel putin un polinom de gradul 3 pentru a obtine o functie continua.

- Derivarea numerica este afectata de eroarea de trunchiere.
- Teoretic, eroarea de trunchiere scade cu cresterea ordinul aproximarii.
- Datorita fenomenului Runge, formulele de derivare de ordin superior ($n > 5$) pot fi afectate de erori de trunchiere importante.
- Astfel, derivarea numerica poate prezenta instabilitati numerice, in special pentru formulele de ordin superior.
- Eroarea de trunchiere depinde direct proportional cu pasul de derivare h.
- Eroarea de rotunjire depinde invers proportional cu pasul de derivare h.
- Pentru valori foarte mici ale pasului h, eroarea de trunchiere scade considerabil, insa eroarea de rotunjire devine importanta.
- Practic, exista un pas optim pentru care eroarea totala este minima.

Pseudocod

procedura derivtab (n, h, y, dy)
; calculeaza tabelul de derivare a unei functii definite tabelar in n+1 noduri
intreg n ; n+1 este numarul de noduri ale retelei de discretizare
real h ; pasul constant al retelei de discretizare
tablou real $y(n+1)$; valorile cunoscute ale functiei, indici de la 0 la n
tablou real $dy(n+1)$; valorile derivatei functiei, indici de la 0 la n
$dy_0 = (-3y_0 + 4y_1 - y_2)/(2h)$; formula progresiva de ordinul 2
$dy_n = (y_{n-2} - 4y_{n-1} + 3y_n)/(2h)$; formula regresiva de ordinul 2
pentru $i = 1, n-1$
    $dy_i = (y_{i+1} - y_{i-1})/(2h)$; formula centrata de ordinul 2
retur

program principal
; declaratii
intreg n, k
tablou real $x(n+1)$, $y(n+1)$, $dy(n+1)$
; introducere date intrare
citeste n ; n+1 este numarul de noduri ale retelei de discretizare
pentru $k = 0, n$
    citeste x_k, y_k ; nodurile retelei de discretizare
$h = x_1 - x_0$; determinare pasul retelei de discretizare uniforme
; apelare procedura metodei de calcul
derivtab (n, h, y, dy)
; afisare date iesire
pentru $k = 0, n$
    scrie dy_k ; valorile derivatei numerice in nodurile retelei de discretizare
stop

8. Integrarea numerica a functiilor reale

Fie functia $f:[a,b] \rightarrow$ IR, $f(x)=y$ reprezentata prin date (tabelar): se cunosc valorile functiei doar intr-o retea de puncte din domeniul de definitie, numite noduri.

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

← retea de discretizare cu n+1 noduri

Valoarea exacta a integralei reprezinta aria suprafetei subintinse de graficul functiei intre capetele domeniului de definitie:

$$I_0 = \int_a^b f(x)\,\mathrm{d}\,x$$

Integrarea numerica se bazeaza pe interpolarea polinomiala pe portiuni!

Integrarea numerica este mult mai robusta decat derivarea numerica.

Metoda trapezelor

In cazul metodei de interpolare liniara pe portiuni, graficul functiei $f(x)$ este aproximat cu linia poligonala ($g(x)$) care uneste toate nodurile retelei de discretizare.

Astfel, intre doua noduri succesive, (x_k, y_k) si (x_{k+1}, y_{k+1}), graficul functiei este aproximat cu dreapta care uneste nodurile respective.

Integrala numerica pe intervalul (x_k, x_{k+1}) este aria trapezului sprijinit de axa Ox, determinat de abscisele x_k, x_{k+1} si ordonatele y_k, y_{k+1}:

$$I_k = \int_{x_k}^{x_{k+1}} g(x)\,\mathrm{d}\,x = \frac{(y_k + y_{k+1})(x_{k+1} - x_k)}{2}$$

Integrala numerica pe intervalul (a,b) este suma ariilor celor n trapeze:

$$I = \sum_{k=0}^{n-1} I_k = \sum_{k=0}^{n-1} \frac{(y_k + y_{k+1})(x_{k+1} - x_k)}{2}$$

Pentru o retea de discretizare uniforma, $h=x_k-x_{k-1}$=ct., formula integralei numerice se simplifica:

$$I=\frac{h}{2}\sum_{k=0}^{n-1}\left(y_k+y_{k+1}\right)=\frac{h}{2}\left(y_0+2y_1+2y_2+\ldots+2y_{n-1}+y_n\right)$$

Metoda Simpson 1/3

Graficul functiei $f(x)$ este aproximat intre trei noduri consecutive, (x_{k-1}, y_{k-1}), (x_k, y_k), (x_{k+1}, y_{k+1}), cu o parabola ($g(x)$) care uneste cele trei noduri.

Integrala numerica pe intervalul (x_{k-1}, x_{k+1}) este aria suprafetei subintinse de parabola:

$$I_k=\int_{x_{k-1}}^{x_{k+1}} g(x)\,\mathrm{d}\,x=\frac{h}{3}\left(y_{k-1}+4y_k+y_{k+1}\right),$$

unde, din motive de simplitate, s-a considerat o retea de discretizare uniforma, $h=x_k-x_{k-1}$=ct..

Integrala numerica pe intervalul (a,b) este suma ariilor suprafetelor subintinse de parabole:

$$I = \sum_{\substack{k=1 \\ k=k+2}}^{n-1} I_k = \frac{h}{3} \sum_{\substack{k=1 \\ k=k+2}}^{n-1} \left(y_{k-1} + 4y_k + y_{k+1} \right) \Rightarrow$$

$$I = \frac{h}{3} \left(y_0 + 4y_1 + 2y_2 + 4y_3 + \ldots + 2y_{n-2} + 4y_{n-1} + y_n \right)$$

- Numarul de noduri (n+1) al retelei de discretizare trebuie sa fie impar.
- Metoda Simpson este mai precisa decat metoda trapezelor
- Metoda trapezelor este mai robusta decat metoda Simpson, numarul de noduri nefiind restrictionat.

- Eroarea de metoda este eroarea de trunchiere.
- Eroarea scade cu cresterea numarului de noduri.
- Eroare: locala (pe fiecare subinterval) si globala (suma erorilor locale),
- Eroarea locala are ordinul $O(h^3)$ la metoda trapezelor si ordinul $O(h^5)$ la metoda Simpson.
- Eroarea globala are ordinul $O(h^2)$ la metoda trapezelor si ordinul $O(h^2)$ la metoda Simpson, fiind mai mare decat eroarea locala.
- In practica, metoda trapezelor ofera rezultate satisfacatoare daca numarul de noduri este rezonabil de mic.

Pseudocod

functia trapez (a, b, n)
; calculeaza integrala functiei f pe intervalul [a, b], cu metoda trapezelor,
; se cunoaste expresia analitica a functiei
intreg n ; numarul de subintervale ale retelei de discretizare uniforme
real h ; pasul constant al retelei de discretizare
real a, b ; capetele intervalului de definitie
intreg k
real r ; valoarea integralei
$h = (b - a)/n$
$r = 0$
pentru k= 1, n−1
    $r = r + f(a+k \cdot h)$
$r = (2r + f(a)+f(b)) \cdot h/2$
intoarce r

functia Simpson (a, b, n, y)
; calculeaza integrala functiei f pe intervalul [a, b] cu metoda Simpson,
; functia este definita tabelar
intreg n ; numarul de subintervale ale retelei de discretizare uniforme
real h ; pasul constant al retelei de discretizare
real a, b ; capetele intervalului de definitie
tablou real $y(n+1)$; valoarea functiei in nodurile retelei de discretizare
intreg k
real r ; valoarea integralei
$h = (b - a)/n$
$r = 0$
pentru $k = 1, n-1, 2$
    $r = r + y_{k-1} + 4y_k + y_{k+1}$
$r = r \cdot h/3$
intoarce r

program principal
; declaratii
intreg n, k
tablou real $x(n+1)$, $y(n+1)$
real a, b, val
; introducere date intrare
citeste n ; n+1 este numarul de noduri ale retelei de discretizare
pentru $k = 0, n$
    citeste x_k, y_k ; nodurile retelei de discretizare
$a = x_0$; limita inferioara a domeniului de definitie
$b = x_n$; limita superioara a domeniului de definitie
; apelare functia metodei de calcul
val = Simpson (a, b, n, y)
; afisare date iesire
scrie val ; valorea integralei numerice determinata cu metoda Simpson
stop

9. Metode iterative de rezolvare a ecuatiilor neliniare

Fie functia continua $f:[a,b] \rightarrow IR$, $f(x)=y$.

Solutia numerica a ecuatiei neliniare (transcendente) $f(x)=0$ se obtine prin generarea unui sir de solutii care tinde catre solutia exacta.

Metoda bisectiei (metoda injumatatirii intervalului)

In intervalul de definitie $[a,b]$ trebuie sa existe o singura solutie a ecuatiei neliniare, $f(a)\cdot f(b)<0$.

La fiecare iteratie se calculeaza jumatatea intervalului de definitie $xm=(a+b)/2$, si se determina jumatatea in care se afla solutia, aceasta fiind noul interval de definitie:

daca $f(xm) \cdot f(a) < 0$ **atunci**

$b=xm$; solutia se afla in prima jumatate

altfel

$a=xm$; solutia se afla in a doua jumatate

Procesul iterativ se opreste cand $(b - a) < eps$ (valoare impusa).

Metoda iteratiei simple

$$x^{(0)} \to x^{(1)} \to x^{(2)} \to \ldots \to x^{(k)} \to \ldots \qquad \lim_{k\to\infty} x^{(k)} = \mathrm{x}$$

$$f(x) = 0 \Leftrightarrow x = g(x)$$

$x^{(k+1)} = g\left(x^{(k)}\right)$ – formula de recurenta

$g(x) = x + c \cdot f(x)$ – functie de iteratie, reprezinta o aplicatie cu punct fix (solutia exacta)

$$\Rightarrow x^{(k+1)} = x^{(k)} + c \cdot f\left(x^{(k)}\right)$$ – formula de recurenta

Procesul iterativ se opreste cand ($x^{k+1} - x^{k}$) < *eps* (valoare impusa de utilizator) sau numarul maxim de iteratii este atins.

Conditia suficienta pentru ca procesul iterativ sa fie convergent este:

- g sa fie o contractie: $|g(u) - g(v)| \leq L \cdot |u - v|$, cu $L < 1$, pentru orice $u, v \in [a, b]$
- $|g'| < 1 \rightarrow |1 + c \cdot f'(x)| < 1$.

Valoarea constantei c infuenteaza puternic convergenta procesului iterativ, aceasta trebuie sa aiba semn opus derivatei functiei f si trebuie aleasa astfel incat ultima inegalitate sa fie adevarata.

Cu cat modulul derivatei functiei de iteratie g este mai mic ca unitatea, cu atat procesul iterativ este mai rapid convergent.

Metoda Newton (metoda tangentelor)

Metoda cea mai rapid convergenta, la fiecare iteratie valoarea coeficientului c se modifica astfel incat $|g'| = 0 \rightarrow |1 + c \cdot f'(x)| = 0$:

$$c = c_k = -\frac{1}{f'\left(x^{(k)}\right)}$$

$$\Rightarrow x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(k)}\right)}$$ – formula de recurenta

La fiecare iteratie graficul functiei este aproximat cu tangenta dusa in punctul de coordonate $(x^k, f(x^k))$. Noua solutie x^{k+1} se afla la intersectia tangentei cu abscisa OX (y=0).

Derivata functiei trebuie evaluata la fiecare iteratie.

Metoda esueaza daca atinge un punct de extrem, $f'(x^k)$=0.

Metoda Newton-Kantorovici (metoda tangentelor paralele)

Reprezinta o varianta simplificata a metodei Newton, in care derivata functiei se evalueaza o singura data, in punctul corespunzator solutiei initiale:

$$c = -\frac{1}{f'\left(x^{(0)}\right)}$$

$$\Rightarrow x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right)}{f'\left(x^{(0)}\right)}$$ – formula de recurenta

Metoda este mult mai slab convergenta decat metoda Newton, valoarea lui c este optima doar in punctul corespunzator solutiei initiale.

Metoda Newton discreta (metoda secantelor)

Derivata functiei $f(x)$ se calculeaza prin formula regresiva de ordinul 1:

$$f'\left(x^{(k)}\right) \approx \frac{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}{x^{(k)} - x^{(k-1)}}$$

$$\Rightarrow x^{(k+1)} = x^{(k)} - \frac{f\left(x^{(k)}\right) \cdot \left(x^{(k)} - x^{(k-1)}\right)}{f\left(x^{(k)}\right) - f\left(x^{(k-1)}\right)}$$ – formula de recurenta

Necesita o dubla initializare a solutiei: $x^{(0)}$, $x^{(1)}$.

Metoda este mai slab convergenta decat metoda Newton, dar mai rapid convergenta decat metoda Newton-Kantorovici.

functia bisectie (*a*, *b*, *eps*, *itmax*)
; calculeaza solutia ecuatiei cu metoda bisectiei
real *a*, *b* ; capetele intervalului de definitie
real *eps* ; eroarea impusa
intreg *itmax* ; numarul maxim de iteratii
intreg *k*
real *xm* ; solutia
k = 0
repeta
    k = *k* + 1
    xm = (*a* + *b*)/2
    daca *f*(*xm*) · *f*(*a*) < 0 **atunci**
        b = *xm* ; solutia se afla in prima jumatate
    altfel
        a = *xm* ; solutia se afla in a doua jumatate
pana cand (*b* − *a*) < *eps* **sau** *k* > *itmax*
intoarce *xm*

functia Newton (*a*, *b*, *eps*, *itmax*)
; calculeaza solutia ecuatiei cu metoda Newton
real *a*, *b* ; capetele intervalului de definitie
real *eps* ; eroarea impusa
intreg *itmax* ; numarul maxim de iteratii
intreg *k*
real *d*
real *xnou*, *xvechi* ; solutii
k = 0
xvechi = (*a* + *b*)/2
repeta
    k = *k* + 1
    xnou = *xvechi* – *f*(*xvechi*)/*f* ’(*xvechi*)
    d = |*xnou* – *xvechi*| ; eroarea Cauchy
     xvechi = *xnou*
pana cand *d* < *eps* **sau** *k* > *itmax*
intoarce *xnou*

functia secante (*a*, *b*, *eps*, *itmax*)
; calculeaza solutia ecuatiei cu metoda Newton discreta
real *a*, *b* ; capetele intervalului de definitie
real *eps* ; eroarea impusa
intreg *itmax* ; numarul maxim de iteratii
intreg *k*
real *d*
real *xnou*, *xvechi*, *xvvechi*; solutii
k = 0
xvechi = *a*
xvvechi = *b*
repeta
    k = *k* + 1
    xnou = *xvechi* – (*xvechi* – *xvvechi*)·*f*(*xvechi*)/(*f*(*xvechi*) – *f*(*xvvechi*))
    d = |*xnou* – *xvechi*| ; eroarea Cauchy
    xvvechi = *xvechi*
     xvechi = *xnou*
pana cand *d* < *eps* **sau** *k* > *itmax*
intoarce *xnou*

program principal
; declaratii
intreg *itmax*
real *a*, *b*
real *eps*, *s*
; introducere date intrare
citeste *a*, *b* ; capetele intervalului de definitie
citeste *eps* ; eroarea impusa
citeste *itmax* ; numarul maxim de iteratii
; apelare functia metodei de calcul
s = bisectie (*a*, *b*, *eps*, *itmax*)
; afisare date iesire
scrie *s* ; solutia ecuatiei neliniare
stop

Ex.: $x^2 + x = 6 \Leftrightarrow f(x) = x^2 + x - 6 = 0$

Metoda bisectiei: $a = -8, b = 0$

$$f(a) = f(-8) = 50,\ f(b) = f(0) = -6 \Rightarrow f(a) \cdot f(b) < 0$$

$$k = 1: xm^{(1)} = \frac{a+b}{2} = -4 \Rightarrow f(xm^{(1)}) = f(-4) = 6$$

$$f(a) \cdot f(xm^{(1)}) = f(-8) \cdot f(-4) > 0 \Rightarrow a = xm^{(1)} = -4 \Rightarrow$$

$$x \in [-4, 0]$$

$$k = 2: xm^{(2)} = \frac{a+b}{2} = -2 \Rightarrow f(xm^{(2)}) = f(-2) = -4$$

$$f(a) \cdot f(xm^{(2)}) = f(-4) \cdot f(-2) < 0 \Rightarrow b = xm^{(2)} = -2 \Rightarrow$$

$$x \in [-4, -2]$$

Metoda Newton: $x^{(0)} = -1 \Rightarrow$

$$x^{(1)} = x^{(0)} - \frac{f\left(x^{(0)}\right)}{f'\left(x^{(0)}\right)} = -1 - \frac{f(-1)}{f'(-1)} = -1 - \frac{-6}{-1} = -7$$

$$x^{(2)} = x^{(1)} - \frac{f\left(x^{(1)}\right)}{f'\left(x^{(1)}\right)} = -7 - \frac{f(-7)}{f'(-7)} = -7 - \frac{36}{-13} = -\frac{55}{13}$$

Metoda Newton discreta: $x^{(0)} = -2, x^{(1)} = -1 \Rightarrow$

$$x^{(2)} = x^{(1)} - \frac{f\left(x^{(1)}\right) \cdot \left(x^{(1)} - x^{(0)}\right)}{f\left(x^{(1)}\right) - f\left(x^{(0)}\right)} = -1 - \frac{f(-1) \cdot (-1 - (-2))}{f(-1) - f(-2)} = -4$$

$$x^{(3)} = x^{(2)} - \frac{f\left(x^{(2)}\right) \cdot \left(x^{(2)} - x^{(1)}\right)}{f\left(x^{(2)}\right) - f\left(x^{(1)}\right)} = -4 - \frac{f(-4) \cdot (-4 - (-1))}{f(-4) - f(-1)} = -\frac{5}{2}$$