Referat 2

Tehnici şi metode numerice în securitatea informaţiei

Profesor universitar dr. habil. Mircea Merca

Student masterand: Ștefan-Daniel Wagner

Cuprins

1. Metoda Jacobi

Teorie

Ideea metodei

Pentru sistemul liniar

$$Ax = b,$$

scriem descompunerea

$$A = D - L - U,$$

unde D este partea diagonală, iar L și U sunt părțile strict triunghiulare (definite cu semn schimbat), astfel încât într-adevăr $A = D - L - U$.

Atunci ecuația

$$Ax = b$$

se rescrie ca

$$(D - L - U)x = b \;\Rightarrow\; Dx = (L + U)x + b.$$

Dacă D^{-1} există (adică $a_{ii} \neq 0$ pentru orice i), rezultă forma matricială a iterației Jacobi:

$$x^{(k)} = D^{-1}(L + U)x^{(k-1)} + D^{-1}b, \qquad k = 1,2, \dots$$

Metoda **Jacobi** definește o succesiune de aproximații $\{x^{(k)}\}$ prin:

$$x^{(k)} = D^{-1}(L + U)x^{(k-1)} + D^{-1}b.$$

Pe componente (pentru $i = 1, \dots, n$):

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j \neq i} a_{ij}\, x_j^{(k)}\right), \quad a_{ii} \neq 0.$$

Observația importantă: toate componentele lui $x^{(k+1)}$ se calculează folosind doar valorile din $x^{(k)}$.

Matricea de iterație

Formula se poate scrie în forma standard

$$x^{(k+1)} = T_J x^{(k)} + c_J,$$

unde

$$T_J = D^{-1}(L + U), \qquad c_J = D^{-1}b.$$

Convergență (condiții suficiente)

Metoda Jacobi converge dacă raza spectrală $\rho(T_J) < 1$.

O condiție suficientă practică (ușor de verificat) este ca A să fie **strict diagonal dominantă** pe linii:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \text{pentru orice } i.$$

Criterii de oprire

În practică se folosește un criteriu pe diferența dintre iterații (normă infinit):

$$\frac{\| x^{(k)} - x^{(k-1)} \|_\infty}{\| x^{(k)} \|_\infty} < \varepsilon,$$

și/sau un criteriu pe reziduu:

$$\| Ax^{(k)} - b \|_\infty < \varepsilon.$$

Legătura cu implementarea din proiect

- Implementare: `JacobiSolver::iterate` în `nm-lib/src/linear/Jacobi.cpp`.
- Driver/test: `nm-lib/tests/tema3_iterative.cpp`.

Exemplu (sistemul 4)

Calculăm primele două iterații pentru metoda **Jacobi**, pornind de la

$$x^{(0)} = 0,$$

pentru sistemul (4):

$$\begin{cases} 4x_1 + x_2 - x_3 + x_4 = -2 \\ x_1 + 4x_2 - x_3 - x_4 = -1 \\ -x_1 - x_2 + 5x_3 + x_4 = 0 \\ x_1 - x_2 + x_3 + 3x_4 = 1 \end{cases}$$

Formulele de actualizare (ecuație cu ecuație, rezolvând după variabila de pe diagonală):

$$\begin{aligned} x_1 &= \frac{-2 - x_2 + x_3 - x_4}{4} \\ x_2 &= \frac{-1 - x_1 + x_3 + x_4}{4} \\ x_3 &= \frac{x_1 + x_2 - x_4}{5} \\ x_4 &= \frac{1 - x_1 + x_2 - x_3}{3} \end{aligned}$$

Valorile numerice de mai jos sunt generate cu implementarea din proiect: `nm-lib/tests/tema3_iterative.cpp` (JacobiSolver), rulat cu `--json --trace 4`.

Iterația 1

$$\begin{aligned} x_1^{(1)} &= \frac{-2}{4} = -1/2 \\ x_2^{(1)} &= \frac{-1}{4} = -1/4 \\ x_3^{(1)} &= 0 \\ x_4^{(1)} &= \frac{1}{3} \end{aligned}$$

Iterația 2

$$x_1^{(2)} = \frac{-2 - x_2^{(1)} + x_3^{(1)} - x_4^{(1)}}{4} = -25/48$$
$$x_2^{(2)} = \frac{-1 - x_1^{(1)} + x_3^{(1)} + x_4^{(1)}}{4} = -1/24$$
$$x_3^{(2)} = \frac{x_1^{(1)} + x_2^{(1)} - x_4^{(1)}}{5} = -13/60$$
$$x_4^{(2)} = \frac{1 - x_1^{(1)} + x_2^{(1)} - x_3^{(1)}}{3} = 5/12$$

Tabel (iterațiile 0–2)

iter	x_1	x_2	x_3	x_4
0	0	0	0	0
1	$-1/2$	$-1/4$	0	$1/3$
2	$-25/48$	$-1/24$	$-13/60$	$5/12$

2. Metoda Gauss–Seidel

Teorie

Ideea metodei

Pentru sistemul liniar

$$Ax = b,$$

scriem

$$A = D - L - U,$$

unde D este diagonală, iar L și U sunt părțile strict triunghiulare (definite cu semn schimbat), astfel încât $A = D - L - U$.

Din

$$(D - L - U)x = b$$

obținem

$$(D - L)x = Ux + b.$$

Metoda **Gauss–Seidel** definește iterația:

$$(D - L)x^{(k)} = Ux^{(k-1)} + b, \qquad x^{(k)} = (D - L)^{-1}\left(Ux^{(k-1)} + b\right), \qquad k = 1,2, \dots$$

Pe componente (pentru $i = 1, \dots, n$):

$$x_i^{(k)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}\, x_j^{(k)} - \sum_{j>i} a_{ij}\, x_j^{(k-1)}\right), \quad a_{ii} \neq 0.$$

Observația importantă: când calculăm $x_i^{(k+1)}$, folosim imediat valorile deja actualizate $x_1^{(k+1)}, \dots, x_{i-1}^{(k+1)}$.

Matricea de iterație

În forma standard:

$$x^{(k)} = B_{GS}x^{(k-1)} + c_{GS},$$

unde

$$B_{GS} = (D - L)^{-1}U, \qquad c_{GS} = (D - L)^{-1}b.$$

Convergență (condiții suficiente)

Gauss–Seidel converge dacă $\rho(B_{GS}) < 1$.

Condiţii suficiente utilizate frecvent:

- A este **strict diagonal dominantă** pe linii;
- sau A este **simetrică pozitiv definită**.

Criterii de oprire

Tipic, folosim aceleaşi criterii ca la Jacobi:

$$\frac{\| x^{(k)} - x^{(k-1)} \|_\infty}{\| x^{(k)} \|_\infty} < \varepsilon, \qquad \| Ax^{(k)} - b \|_\infty < \varepsilon.$$

Legătura cu implementarea din proiect

- Implementare: `GaussSeidelSolver::iterate` în `nm-lib/src/linear/GaussSeidel.cpp`.
- Driver/test: `nm-lib/tests/tema3_iterative.cpp`.

Exemplu (sistemul 4)

Calculăm primele două iteraţii pentru metoda **Gauss–Seidel**, pornind de la

$$x^{(0)} = 0,$$

pentru sistemul (4):

$$\begin{cases} 4x_1 + x_2 - x_3 + x_4 = -2 \\ x_1 + 4x_2 - x_3 - x_4 = -1 \\ -x_1 - x_2 + 5x_3 + x_4 = 0 \\ x_1 - x_2 + x_3 + 3x_4 = 1 \end{cases}$$

Formulele de actualizare (ecuaţie cu ecuaţie, rezolvând după variabila de pe diagonală):

$$\begin{aligned} x_1 &= \frac{-2 - x_2 + x_3 - x_4}{4} \\ x_2 &= \frac{-1 - x_1 + x_3 + x_4}{4} \\ x_3 &= \frac{x_1 + x_2 - x_4}{5} \\ x_4 &= \frac{1 - x_1 + x_2 - x_3}{3} \end{aligned}$$

În Gauss–Seidel, în cadrul aceleiaşi iteraţii, folosim imediat valorile deja actualizate.

Valorile numerice de mai jos sunt generate cu implementarea din proiect: `nm-lib/tests/tema3_iterative.cpp` (GaussSeidelSolver), rulat cu `--json --trace 4`.

Iterația 1

$$x_1^{(1)} = -1/2$$
$$x_2^{(1)} = \frac{-1 - x_1^{(1)} + x_3^{(0)} + x_4^{(0)}}{4} = -1/8$$
$$x_3^{(1)} = \frac{x_1^{(1)} + x_2^{(1)} - x_4^{(0)}}{5} = -1/8$$
$$x_4^{(1)} = \frac{1 - x_1^{(1)} + x_2^{(1)} - x_3^{(1)}}{3} = 1/2$$

Iterația 2

$$x_1^{(2)} = \frac{-2 - x_2^{(1)} + x_3^{(1)} - x_4^{(1)}}{4} = -5/8$$
$$x_2^{(2)} = \frac{-1 - x_1^{(2)} + x_3^{(1)} + x_4^{(1)}}{4} = 0$$
$$x_3^{(2)} = \frac{x_1^{(2)} + x_2^{(2)} - x_4^{(1)}}{5} = -9/40$$
$$x_4^{(2)} = \frac{1 - x_1^{(2)} + x_2^{(2)} - x_3^{(2)}}{3} = 37/60$$

Tabel (iterațiile 0–2)

iter	x_1	x_2	x_3	x_4
0	0	0	0	0
1	$-1/2$	$-1/8$	$-1/8$	$1/2$
2	$-5/8$	0	$-9/40$	$37/60$

3. Newton pentru sisteme (neliniare)

Teorie

Un **sistem de ecuaţii neliniare** poate fi scris în forma

$$F(x) = 0,$$

unde $F: \mathbb{R}^n \to \mathbb{R}^n$ este o funcţie vectorială, iar $x \in \mathbb{R}^n$ este necunoscuta.

Metoda lui Newton pentru sisteme

Ideea este aceeaşi ca la cazul scalar, dar liniarizăm sistemul în jurul lui $x^{(k)}$ folosind Jacobianul:

$$F\left(x^{(k)} + \Delta x\right) \approx F\left(x^{(k)}\right) + J\left(x^{(k)}\right)\Delta x.$$

Impunem $F\left(x^{(k)} + \Delta x\right) = 0$ în aproximare şi obţinem problema liniară:

$$J\left(x^{(k)}\right)\Delta x^{(k)} = -F\left(x^{(k)}\right),$$

apoi actualizarea:

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}.$$

Jacobianul

Jacobianul este matricea derivatelor parţiale:

$$J(x) = \left[\frac{\partial f_i}{\partial x_j}(x)\right]_{i,j=1..n}.$$

În implementare, fie îl calculăm analitic (preferabil), fie aproximăm numeric (diferenţe finite) când nu avem o formulă convenabilă.

Criterii de oprire

În practică sunt folosite una sau mai multe condiţii:

- **Schimbare mică a iteraţiilor** (norma infinit):

$$\frac{\| x^{(k+1)} - x^{(k)} \|_\infty}{\| x^{(k+1)} \|_\infty} < \varepsilon.$$

- **Rezidual mic**:

$$\| F\left(x^{(k+1)}\right) \|_\infty < \varepsilon.$$

Observații despre convergență

- Convergența este, în general, **locală**: trebuie un $x^{(0)}$ suficient de aproape de soluție.
- Dacă $J(x^{(k)})$ este (aproape) singular, metoda poate eșua sau poate deveni instabilă numeric.
- În fiecare pas trebuie rezolvat un sistem liniar cu matricea $J(x^{(k)})$.

Exemplu (sistemul 4)

Luăm **sistemul (4)** (2 ecuații, 2 necunoscute):

$$\begin{cases} 3x_1^2 - x_2^2 = 0 \\ 3x_1x_2^2 - x_1^3 - 1 = 0 \end{cases}$$

Soluții (verificare rapidă)

Din prima ecuație rezultă $x_2^2 = 3x_1^2$. Înlocuim în a doua:

$$3x_1\,(3x_1^2) - x_1^3 - 1 = 0 \;\Rightarrow\; 8x_1^3 - 1 = 0 \;\Rightarrow\; x_1 = \frac{1}{2}.$$

Apoi $x_2 = \pm\frac{\sqrt{3}}{2}$.

Funcția vectorială și Jacobianul

Definim $F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix}$ cu

$$f_1(x_1, x_2) = 3x_1^2 - x_2^2, \qquad f_2(x_1, x_2) = 3x_1x_2^2 - x_1^3 - 1.$$

Jacobianul este:

$$J(x_1, x_2) = \begin{bmatrix} 6x_1 & -2x_2 \\ 3x_2^2 - 3x_1^2 & 6x_1x_2 \end{bmatrix}.$$

La fiecare pas Newton rezolvă:

$$J(x^{(k)})\,\Delta x^{(k)} = -F(x^{(k)}), \qquad x^{(k+1)} = x^{(k)} + \Delta x^{(k)}.$$

Legătura cu implementarea

Valorile numerice sunt cele obținute de implementarea din proiect.

- Program de test (definirea sistemelor + apel Newton): `nm-lib/tests/tema4_newton_systems.cpp`
- Solver Newton pentru sisteme: `nm-lib/include/nonlinear/Newton.h` și `nm-lib/src/nonlinear/Newton.cpp`

- Tipuri pentru sistem neliniar (F și Jacobian): `nm-lib/include/nonlinear/NonlinearSystem.h` și `nm-lib/src/nonlinear/NonlinearSystem.cpp`

Pentru sistemul (4), în cod este folosită de exemplu aproximația inițială:

$$x^{(0)} = (1,\ 2).$$

Soluția numerică este apropiată de:

$$x \approx \left(1/2\,,\ \sqrt{3}/2\right) \approx (0.5,\ 0.8660254),$$

cu rezidual $\| F(x) \|_\infty$ foarte mic (sub ε).