Sisteme de ecuaţii algebrice liniare - metode iterative

Prof.dr.ing. Gabriela Ciuprina

Universitatea "Politehnica" Bucureşti, Facultatea de Inginerie Electrică, Departamentul de Electrotehnică

Suport didactic pentru disciplina *Algoritmi Numerici*, 2017-2018

Notes

Cuprins

Notes

Formularea problemei

Sistem de n ecuaţii algebrice liniare cu n necunoscute:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases} \tag{1}$$

Notes

Formularea problemei

Se dă matricea coeficienţilor

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{2}$$

şi vectorul termenilor liberi

$$\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}^T \in \mathbb{R}^n, \tag{3}$$

se cere să se rezolve sistemul

$$\mathbf{Ax} = \mathbf{b}, \tag{4}$$

unde $\mathbf{x}$ este soluţia

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T \in \mathbb{R}^n. \tag{5}$$

Notes

Buna formulare matematică

Problema este bine formulată din punct de vedere matematic (soluţia există şi este unică)
$\Leftrightarrow$
matricea **A** este nesingulară (are determinantul nenul).
Se scrie formal:

$$"\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}"$$

trebuie citită ca:
"**x** *este soluţia sistemului algebric liniar* $\mathbf{Ax} = \mathbf{b}$"
şi **NU** "*se calculează inversa matricei* **A** *care se înmulţeşte cu vectorul* **b**".

Notes

Condiţionarea problemei

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\| \tag{6}$$

număr de condiţionare la inversare al matricei **A**.

$$\varepsilon_x \leq \kappa(\mathbf{A})\varepsilon_b, \tag{7}$$

- $\kappa(\mathbf{A}) \geq 1$:
  Cazul cel mai favorabil: $n_A = 1$ şi $\varepsilon_x = \varepsilon_b$. (matrice ortogonală)
- Numărul de condiţionare este o proprietate a matricei şi nu are legătură nici cu metoda de rezolvare propriu-zisă, nici cu erorile de rotunjire care apar în mediul de calcul.
  În practică:
  Dacă $\kappa(\mathbf{A}) > 1/\mathtt{eps}$ problema se consideră slab condiţionată.

Notes

Clasificarea metodelor

1. **Metode directe** - găsesc soluţia teoretică a problemei într-un număr finit de paşi. (Gauss, factorizare LU)
2. **Metode iterative** - generează un şir de aproximaţii ale soluţiei care se doreşte a fi convergent către soluţia exactă.
   - **staţionare:** Jacobi, Gauss-Seidel, SOR, SSOR
   - **nestaţionare (semiiterative):** gradienţi conjugaţi (GC), reziduu minim (MINRES), reziduu minim generalizat (GMRES), gradienţi biconjugaţi (BiGC), etc.

Notes

Ideea metodelor staţionare

$$\mathbf{Ax} = \mathbf{b} \tag{8}$$

se construieşte un şir de aproximaţii $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(k)}, \ldots$

$$\lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}^*, \quad \text{unde} \quad \mathbf{Ax}^* = \mathbf{b}. \tag{9}$$

$$\mathbf{x}^{(k)} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{bmatrix} \in \mathbb{R}^{n\times 1}.$$

Notes

Ideea metodelor staţionare

Algoritmul are nevoie de

1. o iniţializare $\mathbf{x}^{(0)}$;
2. un mod de generare a şirului de iteraţii;
3. un criteriu de oprire.

1. Iniţializarea

- în principiu, arbitrară;
- dacă este posibil, cât mai aproape de soluţie.

Notes

Ideea metodelor staţionare

2. Şirul de iteraţii se generează *recursiv*:

$$\mathbf{x}^{(k)} = F(\mathbf{x}^{(k-1)}), \tag{10}$$

$$\mathbf{x}^{*} = F(\mathbf{x}^{*}), \tag{11}$$

$\mathbf{x}^*$ este punct fix pentru aplicaţia F.
În concluzie, soluţia exactă a sistemului de ecuaţii este şi punct fix pentru F. Rezolvarea sistemului de ecuaţii algebrice liniare se face prin căutarea unui punct fix pentru F.

Notes

Ideea metodelor staţionare

$$\mathbf{A} = \mathbf{B} - \mathbf{C}. \tag{12}$$

$$\mathbf{Bx} = \mathbf{Cx} + \mathbf{b} \tag{13}$$

$$\mathbf{x} = \mathbf{Mx} + \mathbf{u}, \tag{14}$$

$$\begin{aligned} \mathbf{M} &= \mathbf{B}^{-1}\mathbf{C}, \\ \mathbf{u} &= \mathbf{B}^{-1}\mathbf{b}. \end{aligned} \tag{15}$$

$\mathbf{M} \in \mathbb{R}^{n\times n}$ se numeşte *matrice de iteraţie*.

$$F(\mathbf{x}) = \mathbf{Mx} + \mathbf{u}, \tag{16}$$

Notes

Ideea metodelor staţionare

$$\mathbf{x}^{(k)} = F(\mathbf{x}^{(k-1)}), \tag{17}$$

$$\mathbf{x}^{(k)} = \mathbf{Mx}^{(k-1)} + \mathbf{u}, \tag{18}$$

$$\mathbf{x}^{(k)} = \mathbf{B}^{-1}\mathbf{Cx}^{(k-1)} + \mathbf{B}^{-1}\mathbf{b}. \tag{19}$$

$$\mathbf{Bx}^{(k)} = \mathbf{Cx}^{(k-1)} + \mathbf{b}, \tag{20}$$

$\mathbf{B}$ are o structură particulară.

Notes

Ideea metodelor staţionare

3. Criteriul de oprire
Condiţie de oprire bazată de criteriul Cauchy de convergenţă:

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| \leq \varepsilon, \tag{21}$$

Se poate întâmpla însă ca şirul iteraţiilor să nu fie convergent.
Procedurile iterative vor avea ca parametri de intrare, pe lângă mărimile ce definesc sistemul:

- o eroare ce reprezintă criteriul dorit de oprire a iteraţiilor;
- un număr maxim de iteraţii, util pentru a asigura oprirea naturală a procedurii în caz de neconvergenţă.

Nu are sens ca $\varepsilon <$ eps $\|\mathbf{x}^{(k)}\|$.

Notes

Util: vectori si valori proprii

Definiţie: vectorii proprii **v** ai unei matrice pătrate reale **M**, de dimensiune *n* sunt acei vectori nenuli, pentru care există un scalar λ astfel încât

$$\mathbf{M}\mathbf{v} = \lambda \mathbf{v}. \tag{22}$$

Obs:

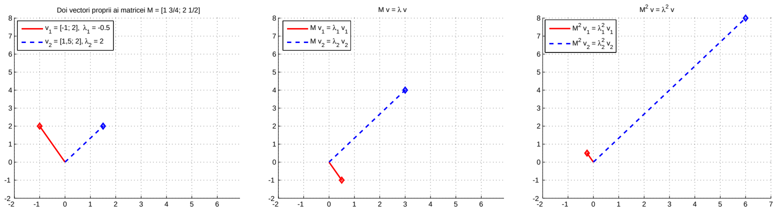
- Reprezentarea geometrică: prin aplicarea **M** asupra lui, vectorul nu se roteşte;
- Vectorii proprii ai unei matrice nu sunt unici. Dacă **v** este un vector propriu, atunci şi vectorul scalat α**v** este de asemenea vector propriu;
- λ se numeşte valoare proprie a matricei **M** asociată vectorului propriu **v**.

Notes

Util: vectori si valori proprii

$$\mathbf{M}^k \mathbf{v} = \lambda^k \mathbf{v}. \tag{23}$$

Dacă $|\lambda| < 1$ atunci $\lim_{k\to\infty} \|\mathbf{M}^k \mathbf{v}\| = 0$.
Dacă $|\lambda| > 1$ atunci $\lim_{k\to\infty} \|\mathbf{M}^k \mathbf{v}\| = \infty$.



Înmulţirea repetată dintre o matrice şi vectorii ei proprii.

Notes

Util: vectori si valori proprii

Dacă **M** este simetrică, atunci ea are n vectori proprii liniar independenţi:
$\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(n)}$.
Această mulţime nu este unică, dar fiecare vector propriu are asociat o valoare proprie unică.
Dacă cei n vectori proprii ai lui **M** formează o bază, atunci orice vector $\mathbf{u} = \sum_{i=1}^{n} \alpha_i \mathbf{v}^{(i)}$.

$$\mathbf{M}^k \mathbf{u} = \alpha_1 \lambda_1^k \mathbf{v}^{(1)} + \cdots + \alpha_n \lambda_n^k \mathbf{v}^{(n)}. \tag{24}$$

Dacă toate valorile proprii sunt subunitare în modul, atunci norma vectorului rezultant va tinde către zero. E suficient ca o singură valoare proprie să fie în modul mai mare ca 1, ca norma vectorului rezultant să tindă către infinit.

Notes

Util: vectori si valori proprii

Raza spectrală a unei matrice: proprii

$$\rho(M) = \max_i |\lambda_i|. \tag{25}$$

Valorile proprii sunt rădăcinile polinomului caracteristic. Deoarece

$$(\lambda \mathbf{I} - \mathbf{M})\mathbf{v} = \mathbf{0} \tag{26}$$

rezultă in mod necesar anularea *polinomului caracteristic al matricei*:

$$\det(\lambda \mathbf{I} - \mathbf{M}) = 0. \tag{27}$$

Ecuaţie de gradul n în λ care, cf. teoremei fundamentale a algebrei, are exact n soluţii (reale sau în perechi complex conjugate), care sunt valorile proprii ale matricei.

Notes

Convergenţă

Teorema 1: Condiţia necesară şi suficientă

ca procesul iterativ să fie convergent este ca raza spectrală a matricei de iteraţie să fie strict subunitară:

$$\rho(M) < 1.$$

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^* = F(\mathbf{x}^{(k-1)}) - F(\mathbf{x}^*) = \mathbf{M}\mathbf{x}^{(k-1)} + \mathbf{u} - \mathbf{M}\mathbf{x}^* - \mathbf{u} = \mathbf{M}\mathbf{e}^{(k-1)}, \tag{28}$$

$$\mathbf{e}^{(k)} = \mathbf{M}\mathbf{e}^{(k-1)} = \mathbf{M}^2\mathbf{e}^{(k-2)} = \cdots = \mathbf{M}^k\mathbf{e}^{(0)}. \tag{29}$$

Notes

Convergenţă

Teorema 2: O condiţie suficientă

ca procesul iterativ să fie convergent este ca norma matricei de iteraţie să fie strict subunitară:

$$\|\mathbf{M}\| < 1.$$

$$\rho(\mathbf{M}) \leq \|\mathbf{M}\|. \tag{30}$$

$$\|\mathbf{e}^{(k)}\| \leq \|\mathbf{M}\|^k \|\mathbf{e}^{(0)}\|. \tag{31}$$

Notes

Convergenţă

Mai mult

$$\begin{aligned} \|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\| &= \|\mathbf{M}\mathbf{x}^{(k-1)} + \mathbf{u} - \mathbf{M}\mathbf{x}^{(k-2)} - \mathbf{u}\| = \\ &= \|\mathbf{M}(\mathbf{x}^{(k-1)} - \mathbf{x}^{(k-2)})\| \leq \\ &\leq \|\mathbf{M}\|\|\mathbf{x}^{(k-1)} - \mathbf{x}^{(k-2)}\|, \end{aligned} \tag{32}$$

$\Rightarrow$ Utilizarea unui criteriu de oprire Cauchy este pe deplin justificată.

Notes

Convergenţă

Fie o margine a erorii absolute notată cu a_k, unde $\|\mathbf{e}^{(k)}\| \leq a_k$.

$$a_k = \|\mathbf{M}\|^k a_0, \tag{33}$$

$$\log(a_k) = k \log \|\mathbf{M}\| + \log a_0. \tag{34}$$

$R(\mathbf{M}) = -\log \|\mathbf{M}\|$ se numeşte rata de convergenţă.

$$\log(a_k) = -kR(\mathbf{M}) + \log a_0. \tag{35}$$

$$R(\mathbf{M}) = \log(a_{k-1}) - \log(a_k), \tag{36}$$

Notes

Convergenţă

$$R(\mathbf{M}) = \log(a_{k-1}) - \log(a_k), \tag{37}$$

Rata de convergenţă = numărul de cifre semnificative corecte ce se câştigă la fiecare iteraţie.

Exemplu:

- $\|\mathbf{M}\| = 10^{-3}$, rata de convergenţă este 3, deci la fiecare iteraţie numărul de cifre semnificative corecte creşte cu 3.
- $\|\mathbf{M}\| = 10^{-1}$, la fiecare iteraţie se câştigă o cifră semnificativă.

OBS:

- Alegerea valorii iniţiale nu are nici o influenţă asupra convergenţei sau neconvergenţei procesului iterativ;
- În cazul unui proces iterativ convergent, valoarea iniţială afectează doar numărul de iteraţii necesar pentru atingerea unei erori impuse.

Notes

Algoritm general

<u>procedură</u> metodă_iterativă($n, B, C, b, x0, er, maxit, x$)
...
xv = **x0** ; iniţializează şirul iteraţiilor
$k = 0$; iniţializare contor iteraţii
<u>repetă</u>
  $\mathbf{t} = \mathbf{C} * \mathbf{xv} + \mathbf{b}$
  metodă_directă (n, B, t, x)
  $d = \|\mathbf{xv} - \mathbf{x}\|$
  xv = **x** ; actualizează soluţia veche
  $k = k + 1$
<u>cât timp</u> $d > er$ <u>şi</u> $k \leq maxit$
<u>retur</u>

Notes

Algoritm general

Efortul de calcul

- poate fi făcut doar pentru o singură iteraţie;
- depinde de structura matricelor în care a fost descompusă matricea coeficienţilor;
- e consumat mai ales în calculul lui **t** şi în procedura de rezolvare directă, care în general are o complexitate liniară deoarece **B** are o structură rară, particulară.
- este de aşteptat ca procedeul iterativ să fie cu atât mai rapid convergent cu cât **B** conţine mai multă informaţie din **A**.

Notes

Metoda Jacobi: un exemplu simplu

A se descompune astfel încât **B** este diagonală.

$$\left\{\begin{array}{rcrcrcr} x & + & 2y & - & z & = & -1 \\ -2x & + & 3y & + & z & = & 0 \\ 4x & - & y & - & 3z & = & -2 \end{array}\right. \Leftrightarrow \left\{\begin{array}{rcl} x & = & -2y+z-1 \\ 3y & = & 2x-z \\ -3z & = & -4x+y-2 \end{array}\right. \tag{38}$$

$$\begin{array}{l} x^{(n)} = -2y^{(v)} + z^{(v)} - 1 \\ y^{(n)} = 2x^{(v)} - z^{(v)} \\ z^{(n)} = -4x^{(v)} + y^{(v)} - 2 \end{array} \tag{39}$$

$[0,0,0]^T$, $[-1,0,-2]^T$, $[-3,0,2]^T$, etc.

Notes

Algoritmul metodei Jacobi

Partiţionarea matricei în metodele iterative:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 \\ a_{31} & a_{32} & 0 & 0 \\ a_{41} & a_{42} & a_{43} & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ 0 & 0 & a_{23} & a_{24} \\ 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \mathbf{L} + \mathbf{D} + \mathbf{U}$$

$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$
$\mathbf{A} = \mathbf{B} - \mathbf{C}$ unde, în metoda Jacobi

$$\mathbf{B} = \mathbf{D}, \quad \mathbf{C} = -(\mathbf{L} + \mathbf{U}), \tag{40}$$

Notes

Algoritmul metodei Jacobi

Calculul recursiv al noii iteraţii

$$\mathbf{D}\mathbf{x}^{(k)} = -(\mathbf{L} + \mathbf{U})\mathbf{x}^{(k-1)} + \mathbf{b}. \tag{41}$$

Ecuaţia i:

$$a_{ii}\mathbf{x}_i^{(k)} = -\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k-1)} + b_i. \tag{42}$$

$$\mathbf{x}_i^{(n)} = (b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} x_j^{(v)})/a_{ii} \quad i = 1, \ldots, n. \tag{43}$$

Obs: Fiecare componentă nouă poate fi calculată independent de celelalte componente noi, motiv pentru care metoda Jacobi se mai numeşte şi metoda deplasărilor simultane.

Notes

Algoritmul metodei Jacobi

```
procedură Jacobi(n, a, b, x0, er, maxit, x)
; rezolvă sistemul algebric liniar ax = b, de dimensiune n prin metoda Jacobi
întreg n                  ; dimensiunea sistemului
tablou real a[n][n]       ; matricea coeficienţilor, indici de la 1
tablou real b[n]          ; vectorul termenilor liberi
; mărimi specifice procedurilor iterative
tablou real x0[n]         ; iniţializarea soluţiei
real er                   ; eroarea folosită de criteriul de oprire
întreg maxit              ; număr maxim de iteraţii admis
tablou real xv[n]         ; aproximaţia anterioară ("veche")
```


Notes

Algoritmul metodei Jacobi

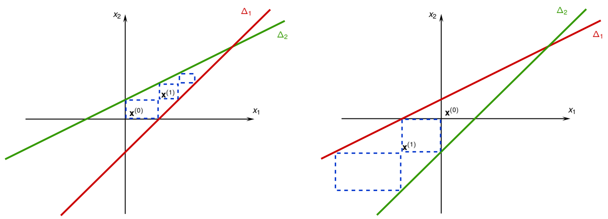
```
procedură Jacobi(n, a, b, x0, er, maxit, x)
.....
pentru i = 1, n
      xv_i = x0_i
•
k = 0                        ; iniţializare contor iteraţii
repetă
      d = 0
      pentru i = 1, n
            s = 0
            pentru j = 1, n
                  dacă j ≠ i
                        s = s + a_ij * xv_j
                  •
            •
            x_i = (b_i − s)/a_ii
            d = d + (x_i − xv_i)^2
      •
      d = √d
      pentru i = 1, n
            xv_i = x_i   ; actualizează soluţia veche
      •
      k = k + 1
cât timp d > er şi k ≤ maxit
retur
```


Notes

Convergenţa metodei Jacobi

Matricea de iteraţie

$$\mathbf{M} = -\mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}). \tag{44}$$



Schimbarea ordinii ecuaţiilor din sistem înseamnă schimbarea matricei de iteraţie, deci a proprietătilor de convergenţă ale metodei Jacobi.

Rezultat util: Dacă matricea coeficienţilor este diagonal dominantă, atunci condiţia suficientă de convergenţă este satisfăcută şi algoritmul Jacobi este convergent.

Notes

Metoda Gauss-Seidel: un exemplu simplu

A se descompune astfel încât **B** este triunghiular inferioară.

$$\begin{cases} x + 2y - z = -1 \\ -2x + 3y + z = 0 \\ 4x - y - 3z = -2 \end{cases} \Leftrightarrow \begin{cases} x = -2y + z - 1 \\ -2x + 3y = -z \\ 4x - y - 3z = -2 \end{cases} \quad (45)$$

$$\begin{array}{rcl} x^{(n)} & = & -2y^{(v)} + z^{(v)} - 1 \\ -2x^{(n)} + 3y^{(n)} & = & -z^{(v)} \\ 4x^{(n)} - y^{(n)} - 3z^{(n)} & = & -2 \end{array} \quad (46)$$

$[0, 0, 0]^T$, $[-1, -2, 4]^T$, $[7, 10, -40]^T$, etc.

Notes

Algoritmul metodei Gauss-Seidel

Partiţionarea matricei în metodele iterative:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 \\ a_{31} & a_{32} & 0 & 0 \\ a_{41} & a_{42} & a_{43} & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ 0 & 0 & a_{23} & a_{24} \\ 0 & 0 & 0 & a_{34} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \mathbf{L} + \mathbf{D} + \mathbf{U}$$

$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$
$\mathbf{A} = \mathbf{B} - \mathbf{C}$, unde în metoda Gauss-Seidel

$$\mathbf{B} = \mathbf{L} + \mathbf{D}, \quad \mathbf{C} = -\mathbf{U}, \quad (47)$$

Notes

Algoritmul metodei Gauss-Seidel

Calculul recursiv al noii iteraţii

$$(\mathbf{L} + \mathbf{D})\mathbf{x}^{(k)} = -\mathbf{U}\mathbf{x}^{(k-1)} + \mathbf{b}. \tag{48}$$

Ecuaţia i:

$$\sum_{j=1}^{i-1} a_{ij} x_j^{(k)} + a_{ii} x_i^{(k)} = -\sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} + b_i. \tag{49}$$

$$\sum_{j=1}^{i-1} a_{ij} x_j^{(n)} + a_{ii} x_i^{(n)} = -\sum_{j=i+1}^{n} a_{ij} x_j^{(v)} + b_i, \tag{50}$$

Rezolvarea sistemului: prin substituţie progresivă, conform formulei:

$$\mathbf{x}_i^{(n)} = (b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(n)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(v)})/a_{ii}. \tag{51}$$

Notes

Algoritmul metodei Gauss-Seidel

Observaţii:

- Este respectat principiul lui Seidel, conform căruia o valoare nouă a unei necunoscute trebuie folosită imediat în calcule.
- O componentă nouă nu poate fi calculată independent de celelalte componente noi, motiv pentru care metoda Gauss-Seidel se mai numeşte şi metoda deplasărilor succesive

Notes

Algoritmul metodei Gauss-Seidel

```
procedură Gauss-Seidel(n, a, b, x0, er, maxit, x)
; rezolvă sistemul algebric liniar ax = b, de dimensiune n prin metoda Gauss-Seidel
; declaraţii ca la procedura Jacobi
. . .
pentru i = 1, n
      xv_i = x0_i
•
k = 0                    ; iniţializare contor iteraţii
repetă
      d = 0
      pentru i = 1, n
            s = b_i
            pentru j = 1, n
                  dacă j ≠ i
                        s = s + a_ij * xv_j
                  •
            •
            s = s/a_ii
            p = |x_i − s|
            dacă p > d
                  d = p
            •
            x_i = s
      •
      k = k + 1
cât timp d > er şi k ≤ maxit
retur
```

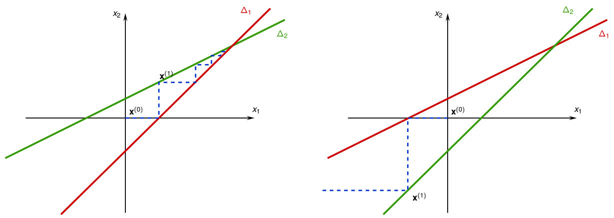

Nu este necesară memorarea soluţiei anterioare (ca la Jacobi). O dată calculată noua valoare, vechea valoare este folosită doar la calculul erorii.

Notes

Convergenţa metodei Gauss-Seidel

Matricea de iteraţie

$$\mathbf{M} = -(\mathbf{L} + \mathbf{D})^{-1}\mathbf{U}. \quad (52)$$



Schimbarea ordinii ecuaţiilor din sistem înseamnă schimbarea matricei de iteraţie, deci a proprietătilor de convergenţă ale metodei Gauss-Seidel.

Rezultate utile:

- Dacă matricea coeficienţilor este diagonal dominantă, atunci algoritmul Gauss-Seidel este convergent.
- În general, dacă metoda Jacobi este convergentă, metoda Gauss-Seidel este mai rapid convergentă.

Notes

Evaluarea algoritmilor Jacobi şi Gauss-Seidel

Efortul total de calcul depinde de numărul de iteraţii m (care depinde de matricea de iteraţie).

- Efortul de calcul pe iteraţie este $O(2n^2)$.
- Efortul total de calcul al algoritmilor Jacobi şi Gauss-Seidel *implementaţi cu matrice pline*: $T = O(2mn^2)$.
- Metoda Jacobi sau Gauss-Seidel este mai eficientă decât metoda Gauss dacă $2mn^2 < 2n^3/3$, deci dacă $m < n/3$.

Necesarul de memorie (matrice pline):

$$M_{GS} = O(n^2 + 2n) \approx O(n^2)$$

$$M_J = O(n^2 + 3n) \approx O(n^2)$$

diferenţa este nesemnificativă

Notes

Evaluarea algoritmilor Jacobi şi Gauss-Seidel

Observaţii:

1. Dacă matricea coeficienţilor este rară (memorată MM, CRS sau CCS), atunci efortul de calcul pe iteraţie se poate diminua.
2. Important: *Nu putem vorbi de umpleri ale matricei coeficienţilor* aşa cum se întâmplă în cazul algoritmului Gauss aplicat pentru matrice rare.

Metodele iterative sunt mai potrivite decât metodele directe pentru rezolvarea sistemelor cu matrice rare, într-un context hardware în care memoria disponibilă nu este suficientă rezolvării prin metode directe.

Notes

Metoda Suprarelaxării succesive (SOR)

Procedeu de accelerare a convergenţei:

$$x_i^{(n)} = x_i^{(v)} + \omega(x_i^{(n\_GS)} - x_i^{(v)}). \quad (53)$$

unde ω se numeşte factor de suprarelaxare
$\omega = 1$ corespunde metodei Gauss-Seidel.
Modificarea în pseudocodul algoritmului Gauss-Seidel:
instrucţiunea $x_i = s$ se înlocuieşte cu $x_i = x_i + \omega(s - x_i)$.

$$\begin{aligned} x_i^{(n)} &= \omega x_i^{(n\_GS)} + (1-\omega)x_i^{(v)} = \\ &= \omega(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(n)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(v)})/a_{ii} + (1-\omega)x_i^{(v)} \quad (54) \end{aligned}$$

$$(\mathbf{D} + \omega\mathbf{L})\mathbf{x}^{(n)} = [(1-\omega)\mathbf{D} - \omega\mathbf{U}]\,\mathbf{x}^{(v)} + \omega b. \quad (55)$$

Notes

Metoda Suprarelaxării succesive (SOR)

Matricea de iteraţie a metodei SOR:

$$\mathbf{M} = (\mathbf{D} + \omega\mathbf{L})^{-1}\left[(1-\omega)\mathbf{D} - \omega\mathbf{U}\right], \quad (56)$$

- Metoda nu converge dacă $\omega \notin (0, 2)$.
- Cazul $\omega \in (0, 1)$ corespunde unei subrelaxări şi se foloseşte atunci când metoda Gauss-Seidel nu converge.
- Dacă matricea este simetrică şi pozitiv definită, SOR este garantat convergentă $(\forall)\ \omega \in (0, 2)$.
- Alegerea valorii lui ω poate afecta în mod semnificativ rata de convergenţă.
- În general, este dificil să se calculeze apriori valoarea optimală a factorului de suprarelaxare.

Notes

Metoda Suprarelaxării succesive (SOR)

- Pentru matricile obţinute prin discretizarea unor ecuaţii cu derivate parţiale prin parcurgerea sistematică a nodurilor reţelei de discretizare se poate demonstra că

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho^2}}, \tag{57}$$

unde ρ este raza spectrală a matricei de iteraţie Jacobi. În practică se folosesc tehnici euristice, ce iau în considerare dimensiunea gridului de discretizare al problemei ["Templates"].
- În cazul matricelor simetrice, o variantă a metodei SOR, numită SSOR, este folosită ca metodă de precondiţionare pentru metode nestaţionare.

Notes

Algoritmul general al metodelor staţionare

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{A} = \mathbf{B} - \mathbf{C}$$

$$\mathbf{Bx}^{(k+1)} = \mathbf{Cx}^{(k)} + \mathbf{b}. \tag{58}$$

$$\mathbf{B}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = \mathbf{b} - \mathbf{Ax}^{(k)}. \tag{59}$$

Reziduul la iteraţia k

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{Ax}^{(k)}, \tag{60}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{B}^{-1}\mathbf{r}^{(k)}, \tag{61}$$

Notes

Algoritmul general al metodelor staţionare

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}, \tag{62}$$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{B}^{-1}\mathbf{r}^{(k)}, \tag{63}$$

Staţionaritatea se referă la faptul că reziduu este întotdeauna înmulţit cu matricea $\mathbf{B}^{-1}$, aceeaşi pe parcursul tuturor iteraţiilor:

- Jacobi: $\mathbf{B} = \mathbf{D}$
- Gauss-Seidel: $\mathbf{B} = \mathbf{D} + \mathbf{L}$
- SOR: $\mathbf{B} = \omega^{-1}(\mathbf{D} + \omega\mathbf{L})$.

Nu se face inversarea propriu zisă, ci se rezolvă un sistem algebric liniar.

Notes

Algoritmul general al metodelor staţionare

```
procedură metodă_iterativă_v2(n, B, A, b, x0, er, maxit, x)
...
xv = x0                          ; iniţializează şirul iteraţiilor
k = 0                            ; iniţializare contor iteraţii
repetă
      r = b − A · xv             ; calculează reziduu
      metodă_directă (n, B, r, z)
      d = ||z||
      x = xv + z
      xv = x                     ; actualizează soluţia veche
      k = k + 1
cât timp d > er şi k ≤ maxit
retur
```


Notes

Algoritmul general al metodelor staţionare

- Metodele iterative sunt eficiente însă pentru matrici rare.
- În cazul matricilor pline, timpul de rezolvare cu metode iterative poate fi comparabil cu timpul de factorizare. Într-o astfel de situaţie, factorizarea este mai utilă deoarece, o dată ce factorii L şi U sunt calculaţi, rezolvarea sistemului poate fi făcută oricând pentru alt termen liber.
- De aceea, un pseudocod simplificat, în care sunt scrise operaţii cu matrice este mai general, putând fi adaptat unor matrice rare.

Notes

Clasa de probleme

Este o metodă iterativă nestaţionară, pentru rezolvarea unor sisteme de ecuaţii algebrice de forma

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \qquad (64)$$

unde $\mathbf{A}$ este simetrică şi pozitiv definită.

- Algoritmul metodei gradienţilor conjugaţi (GC) este eficient pentru matrice rare.

$\Rightarrow$ Pseudocodurile vor fi descrise simplificat, evidenţiind operaţii de algebră liniară, presupuse a fi implementate ţinând cont de raritatea structurilor de date.

Notes

Forma pătratică

Metoda GC este simplu de înţeles dacă ea este formulată ca o problemă de minimizare.
Fie o formă pătratică

$$f : \mathbb{R}^n \to \mathbb{R}$$

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + \mathbf{c}, \tag{65}$$

unde $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{n\times 1}$, $c \in \mathbb{R}$.

Teoremă

Dacă $\mathbf{A}$ este simetrică şi pozitiv definită, atunci funcţia $f(\mathbf{x})$ dată de (65) este minimizată de soluţia ecuaţiei $\mathbf{A}\mathbf{x} = \mathbf{b}$.

Notes

Forma pătratică

Justificarea teoremei:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + \mathbf{c} \tag{66}$$

$$\nabla f(\mathbf{x}) = \frac{1}{2}\mathbf{A}^T\mathbf{x} + \frac{1}{2}\mathbf{A}\mathbf{x} - \mathbf{b}. \tag{67}$$

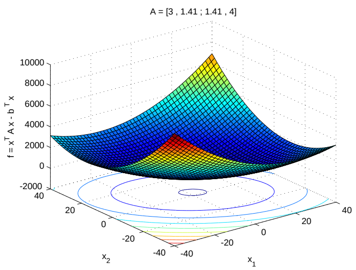
În punctul de minim gradientul este zero

$$\nabla f(\mathbf{x}) = 0 \quad \Rightarrow \quad \frac{1}{2}(\mathbf{A}^T + \mathbf{A})\mathbf{x} = \mathbf{b}. \tag{68}$$

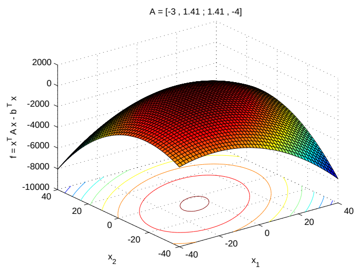
Dacă $\mathbf{A}^T = \mathbf{A}$ (matrice simetrică) atunci (68) $\Leftrightarrow$ $\mathbf{A}\mathbf{x} = \mathbf{b}$.
$\Rightarrow$ soluţia ecuaţiei $\mathbf{A}\mathbf{x} = \mathbf{b}$ este punct critic pentru f.
Dacă în plus, $\mathbf{A}$ pozitiv definită, atunci pct.critic este un minim.

Notes

Matrice pozitiv definită - intuitiv



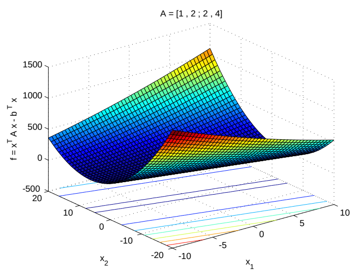
Funcţia pătratică asociată unei matrice pozitiv definite. Minimul funcţiei coincide cu soluţia sistemului. GC functionează.



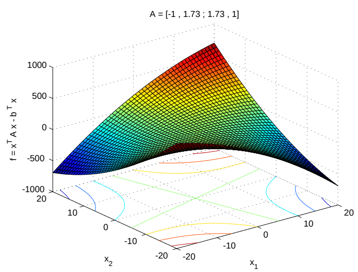
Funcţia pătratică asociată unei matrice negativ definite. Maximul funcţiei coincide cu soluţia sistemului. Algoritmul GC poate fi modificat cu uşurinţă pentru a putea funcţiona.

Notes

Matrice pozitiv semidefinită/indefinită - intuitiv



Funcţia pătratică asociată unei matrice singulare, pozitiv semidefinite. Minimul este atins într-o infinitate de valori. Sistemul are o infinitate de soluţii (problema prost formulată matematic).



Funcţia pătratică asociată unei matrice indefinite. Soluţia sistemului este punct şa pentru f. GC nu funcţionează.

Notes

Metoda celei mai rapide coborâri (a gradientului)

$$\mathbf{Ax} = \mathbf{b}, \tag{69}$$

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{Ax} - \mathbf{b}^T\mathbf{x} + \mathbf{c} \tag{70}$$

$$\nabla f(\mathbf{x}) = \frac{1}{2}\mathbf{A}^T\mathbf{x} + \frac{1}{2}\mathbf{Ax} - \mathbf{b} = \mathbf{Ax} - \mathbf{b}. \tag{71}$$

Ideea: căutarea pe direcţii care corespund ratei celei mai mari de schimbare a funcţiei.
Pentru aproximaţia $\mathbf{x}^{(k)}$, direcţia celei mai rapide coborâri:

$$-\nabla f(\mathbf{x}^{(k)}) = \mathbf{b} - \mathbf{Ax}^{(k)}. \tag{72}$$

Se definesc:

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}, \tag{73}$$

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{Ax}^{(k)}. \tag{74}$$

$\mathbf{e}^{(k)}$ - eroarea la iteraţia k
$\mathbf{r}^{(k)}$ - reziduul la iteraţia k.

Notes

Metoda celei mai rapide coborâri (a gradientului)

$$\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}, \tag{75}$$

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{Ax}^{(k)}. \tag{76}$$

$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}(\mathbf{e}^{(k)} + \mathbf{x}) = \mathbf{b} - \mathbf{Ae}^{(k)} - \mathbf{Ax}.$
Legătura între eroare şi reziduu:

$$\mathbf{r}^{(k)} = -\mathbf{Ae}^{(k)}. \tag{77}$$

$$\nabla f(\mathbf{x}) = \mathbf{Ax} - \mathbf{b}.$$

$$-\nabla f(\mathbf{x}^{(k)}) = \mathbf{b} - \mathbf{Ax}^{(k)}. \tag{78}$$

Reziduul este direcţia celei mai rapide coborâri:

$$\mathbf{r}^{(k)} = -\nabla f(\mathbf{x}^{(k)}). \tag{79}$$

Notes

Metoda celei mai rapide coborâri (a gradientului)

Corecţia primei iteraţii: după direcţia reziduului iniţializării:

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha \mathbf{r}^{(0)}. \tag{80}$$

α se va alege a.î. pe direcţia respectivă funcţia să fie minimă.
$g(\alpha) = f(\mathbf{x}^{(1)})$ minimă, $\Rightarrow$

$$\frac{\mathrm{d}g}{\mathrm{d}\alpha}(\mathbf{x}^{(1)}) = 0 \quad \Rightarrow (\nabla f(\mathbf{x}^{(1)}))^T \cdot \frac{\mathrm{d}\mathbf{x}^{(1)}}{\mathrm{d}\alpha} = 0. \tag{81}$$

Din (79) $\Rightarrow \nabla f(\mathbf{x}^{(1)}) = -\mathbf{r}^{(1)}$;
Din (80) $\Rightarrow \mathrm{d}\mathbf{x}^{(1)}/\mathrm{d}\alpha = \mathbf{r}^{(0)}$.
(81) $\Rightarrow$

$$\begin{aligned}
(\mathbf{r}^{(1)})^T \cdot \mathbf{r}^{(0)} &= 0,\\
(\mathbf{b} - \mathbf{A}\mathbf{x}^{(1)})^T \cdot \mathbf{r}^{(0)} &= 0,\\
[\mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} + \alpha \mathbf{r}^{(0)})]^T \mathbf{r}^{(0)} &= 0,\\
(\mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} - \alpha \mathbf{A}\mathbf{r}^{(0)}))^T \cdot \mathbf{r}^{(0)} &= 0,\\
(\mathbf{b} - \mathbf{A}\mathbf{x}^{(0)})^T \cdot \mathbf{r}^{(0)} - \alpha (\mathbf{A}\mathbf{r}^{(0)})^T \mathbf{r}^{(0)} &= 0,\\
(\mathbf{r}^{(0)})^T \mathbf{r}^{(0)} &= \alpha (\mathbf{r}^{(0)})^T \mathbf{A}^T \mathbf{r}^{(0)}.
\end{aligned}$$

$\mathbf{A}$ simetrică $\Rightarrow$

$$\alpha = \frac{(\mathbf{r}^{(0)})^T \mathbf{r}^{(0)}}{(\mathbf{r}^{(0)})^T \mathbf{A}\mathbf{r}^{(0)}}. \tag{82}$$

Notes

Formularea problemei | Ideea metodei gradienţilor conjugaţi
Metode staţionare | Metoda gradientului
Metoda gradienţilor conjugaţi | Metoda direcţiilor conjugate
Precondiţionare | Metoda gradienţilor conjugaţi

Metoda celei mai rapide coborâri (a gradientului)

Corecţia iteraţiei $k + 1$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}. \tag{83}$$

α_k se va alege a.î.:
$g(\alpha_k) = f(\mathbf{x}^{(k+1)})$ minimă, $\Rightarrow$

$$\frac{\mathrm{d}g}{\mathrm{d}\alpha}(\mathbf{x}^{(k+1)}) = 0 \quad \Rightarrow (\nabla f(\mathbf{x}^{(k+1)}))^T \cdot \frac{\mathrm{d}\mathbf{x}^{(k+1)}}{\mathrm{d}\alpha_k} = 0. \tag{84}$$

Din (79) $\Rightarrow \nabla f(\mathbf{x}^{(k+1)}) = -\mathbf{r}^{(k+1)}$;
Din (83) $\Rightarrow \mathrm{d}\mathbf{x}^{(k+1)}/\mathrm{d}\alpha_k = \mathbf{r}^{(k)}$.
(84) $\Rightarrow$ ortogonalitatea reziduurilor:

$$\begin{aligned}
(\mathbf{r}^{(k+1)})^T \cdot \mathbf{r}^{(k)} &= 0,\\
(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k+1)})^T \cdot \mathbf{r}^{(k)} &= 0,\\
[\mathbf{b} - \mathbf{A}(\mathbf{x}^{(k)} + \alpha \mathbf{r}^{(k)})]^T \mathbf{r}^{(k)} &= 0,\\
(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)} - \alpha \mathbf{A}\mathbf{r}^{(k)}))^T \cdot \mathbf{r}^{(k)} &= 0,\\
(\mathbf{b} - \mathbf{A}\mathbf{x}^{(k)})^T \cdot \mathbf{r}^{(k)} - \alpha (\mathbf{A}\mathbf{r}^{(k)})^T \mathbf{r}^{(k)} &= 0,\\
(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)} &= \alpha (\mathbf{r}^{(k)})^T \mathbf{A}^T \mathbf{r}^{(k)}.
\end{aligned}$$

$\mathbf{A}$ simetrică $\Rightarrow$

$$\alpha = \frac{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{r}^{(k)})^T \mathbf{A}\mathbf{r}^{(k)}}. \tag{85}$$

Notes

Metoda celei mai rapide coborâri (a gradientului)

Algoritm - varianta 0

```
procedură metoda_gradientului_v0(n, A, b, x0, er, maxit, x)
...
xv = x0                              ; iniţializează şirul iteraţiilor
k = 0                                ; iniţializare contor iteraţii
r = b − A · xv                       ; calculează reziduu
cât timp ||r|| > er şi k ≤ maxit
        α = r^T r/(r^T Ar)
        x = xv + αr
        r = b − A · xv
        xv = x                       ; actualizează soluţia veche
        k = k + 1
•
retur
```

Efortul cel mai mare: produsului matrice - vector. (2/iter.).

Notes

Metoda celei mai rapide coborâri (a gradientului)

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}, \tag{86}$$

Înmulţim la stânga cu $-\mathbf{A}$ şi adunăm $\mathbf{b} \Rightarrow$

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{A}\mathbf{r}^{(k)}, \tag{87}$$

Notes

Metoda celei mai rapide coborâri (a gradientului)

```
procedură metoda_gradientului(n, A, b, x0, er, maxit, x)
...
xv = x0                      ; iniţializează şirul iteraţiilor
k = 0                        ; iniţializare contor iteraţii
r = b − A · xv               ; calculează reziduu
e = ||r||
cât timp e > er şi k ≤ maxit
        m = A · r
        α = rᵀr/(rᵀm)
        e = ||r||
        x = xv + αr
        xv = x               ; actualizează soluţia veche
        k = k + 1
        r = r − αm
•
retur
```


Notes

Metoda celei mai rapide coborâri (a gradientului)

Observaţii:

- 1 singur produs matrice vector pe iteraţie;
- Reziduul se calculează recursiv. Se pot acumula erori de rotunjire. Este bine ca periodic $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$.

Notes

Metoda celei mai rapide coborâri (a gradientului)

Analiza convergenţei acestei metode se face pe baza numărului de condiţionare spectrală

$$\kappa = \frac{\max \lambda_i}{\min \lambda_i} \geq 1. \tag{88}$$

- Metoda poate converge rapid, într-un singur pas, dacă punctul de start este ales pe axele elipsoidului sau dacă forma pătratică este sferică (ceea ce corespunde cazului în care toate valorile proprii sunt egale).

Notes

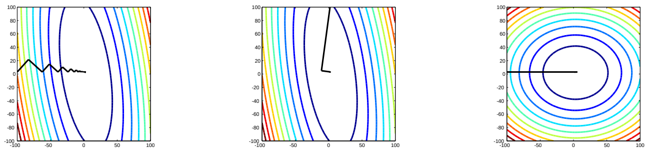
Metoda celei mai rapide coborâri (a gradientului)

- În rest, convergenţa depinde de numărul de condiţionare κ şi de iniţializare.
- Se poate demonstra că eroarea la fiecare iteraţie i este mărginită de

$$\|\mathbf{e}^{(i)}\| \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^i \|\mathbf{e}^{(0)}\| \tag{89}$$

Notes

Metoda celei mai rapide coborâri (a gradientului)



$\kappa = 10$ $\kappa = 10$ $\kappa = 1$

Convergenţa metodei gradientului depinde de iniţializare şi de numărul de condiţionare spectrală. O problemă care are numărul de condiţionare spectrală $\kappa = 1$ (toate valorile proprii sunt egale, iar forma pătratică este sferică) converge într-un singur pas.

Notes

Metoda direcţiilor conjugate

Metoda gradientului converge atât de încet deoarece direcţiile de căutare sunt ortogonale şi se întâmplă ca pe parcursul iteraţiilor direcţiile de căutare să se repete.
Ar fi de dorit să existe exact n direcţii de căutare, $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n-1)}$ astfel încât soluţia să fie găsită după exact n paşi.

Notes

Metoda direcţiilor conjugate

La fiecare pas nou

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}, \tag{90}$$

iar α_k se va alege astfel încât eroarea $\mathbf{e}^{(k+1)}$ să fie ortogonală pe $\mathbf{d}^{(k)}$ şi, în consecinţă, nici o altă corecţie să nu se mai facă în direcţia lui $\mathbf{d}^{(k)}$:

$$(\mathbf{d}^{(k)})^T \mathbf{e}^{(k+1)} = 0. \tag{91}$$

OBS: În cazul particular în care direcţiile $\mathbf{d}^{(k)}$ sunt reziduurile, se obţine exact metoda celei mai rapide coborâri.

Notes

Metoda direcţiilor conjugate

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}, \tag{92}$$

Scădem soluţia exactă din ambii termeni $\Rightarrow$

$$\mathbf{e}^{(k+1)} = \mathbf{e}^{(k)} + \alpha_k \mathbf{d}^{(k)}, \tag{93}$$

Din

$$(\mathbf{d}^{(k)})^T \mathbf{e}^{(k+1)} = 0. \tag{94}$$

$$(\mathbf{d}^{(k)})^T (\mathbf{e}^{(k)} + \alpha_k \mathbf{d}^{(k)}) = 0. \tag{95}$$

$$\alpha_k = -\frac{(\mathbf{d}^{(k)})^T \mathbf{e}^{(k)}}{(\mathbf{d}^{(k)})^T \mathbf{d}^{(k)}}. \tag{96}$$

Nu este utilă deoarece eroarea nu este cunoscută.

Notes

Metoda direcţiilor conjugate

Soluţia constă în a face direcţiile de căutare A-ortogonale şi nu ortogonale:

$$(\mathbf{d}^{(k)})^T \mathbf{A}\mathbf{d}^{(j)} = 0. \qquad j < k. \tag{97}$$

Noua cerinţă: $\mathbf{e}^{(k+1)}$ să fie A-ortogonal pe $\mathbf{d}^{(k)}$

(echivalent cu găsirea unui punct de minim de-a lungul direcţiei $\mathbf{d}^{(k)}$, ca la metoda gradientului).

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\alpha} f(\mathbf{x}^{(k+1)}) &= 0, \\ (\nabla f(\mathbf{x}^{(k+1)}))^T \frac{\mathrm{d}}{\mathrm{d}\alpha}(\mathbf{x}^{(k+1)}) &= 0, \\ -(\mathbf{r}^{(k+1)})^T \mathbf{d}^{(k)} &= 0, \\ (\mathbf{d}^{(k)})^T \mathbf{A}\mathbf{e}^{(k+1)} &= 0. \end{aligned} \tag{98}$$

Notes

Formularea problemei
Metode staţionare
Metoda gradienţilor conjugaţi
Precondiţionare
Ideea metodei gradienţilor conjugaţi
Metoda gradientului
Metoda direcţiilor conjugate
Metoda gradienţilor conjugaţi

Metoda direcţiilor conjugate

Rezultă

$$(\mathbf{d}^{(k)})^T \mathbf{A}(\mathbf{e}^{(k)} + \alpha_k \mathbf{d}^{(k)}) = 0$$

de unde

$$\alpha_k = -\frac{(\mathbf{d}^{(k)})^T \mathbf{A}\mathbf{e}^{(k)}}{(\mathbf{d}^{(k)})^T \mathbf{A}\mathbf{d}^{(k)}} = \frac{(\mathbf{d}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{d}^{(k)})^T \mathbf{A}\mathbf{d}^{(k)}}, \tag{99}$$

expresie ce se poate calcula.

Notes

Metoda direcţiilor conjugate

Teoremă

Metoda direcţiilor conjugate calculează soluţia în exact n paşi.

Pentru demonstraţie consultaţi [Shewchuk94].

Găsirea unui set de direcţii A-ortogonale $\mathbf{d}^{(k)}$ se realizează uşor cu ajutorul procedurii Gram-Schmidt conjugate.

- Se porneşte de la o mulţime de n vectori liniar independenţi $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n-1)}$

$$\mathbf{d}^{(0)} = \mathbf{u}^{(0)}. \tag{100}$$

- A doua direcţie porneşte de la al doilea vector la care se adaugă un vector orientat pe direcţia anterioară, scalat a.î. rezultatul să fie A-ortogonal pe direcţia anterioară.

Notes

Metoda direcţiilor conjugate

$$\mathbf{d}^{(1)} = \mathbf{u}^{(1)} + \beta_{10}\mathbf{d}^{(0)}, \tag{101}$$

unde β_{10} se alege astfel încât

$$(\mathbf{d}^{(1)})^T \mathbf{A}\mathbf{d}^{(0)} = 0. \tag{102}$$

Această relaţie este echivalentă cu

$$(\mathbf{u}^{(1)} + \beta_{10}\mathbf{d}^{(0)})^T \mathbf{A}\mathbf{d}^{(0)} = 0, \tag{103}$$

de unde

$$\beta_{10} = -\frac{(\mathbf{u}^{(1)})^T \mathbf{A}\mathbf{d}^{(0)}}{(\mathbf{d}^{(0)})^T \mathbf{A}\mathbf{d}^{(0)}}. \tag{104}$$

Notes

Metoda direcţiilor conjugate

- Similar, următoarea direcţie este

$$\mathbf{d}^{(2)} = \mathbf{u}^{(2)} + \beta_{20}\mathbf{d}^{(0)} + \beta_{21}\mathbf{d}^{(1)}, \tag{105}$$

unde β_{20} şi β_{21} se aleg astfel încât

$$(\mathbf{d}^{(2)})^T\mathbf{A}\mathbf{d}^{(0)} = 0 \quad \text{şi} \quad (\mathbf{d}^{(2)})^T\mathbf{A}\mathbf{d}^{(1)} = 0, \tag{106}$$

de unde rezultă

$$\beta_{20} = -\frac{(\mathbf{u}^{(2)})^T\mathbf{A}\mathbf{d}^{(0)}}{(\mathbf{d}^{(0)})^T\mathbf{A}\mathbf{d}^{(0)}}, \quad \beta_{21} = -\frac{(\mathbf{u}^{(2)})^T\mathbf{A}\mathbf{d}^{(1)}}{(\mathbf{d}^{(1)})^T\mathbf{A}\mathbf{d}^{(1)}}. \tag{107}$$

Notes

Metoda direcţiilor conjugate

- În general

$$\mathbf{d}^{(k)} = \mathbf{u}^{(k)} + \sum_{j=0}^{k-1} \beta_{kj}\mathbf{d}^{(j)}, \tag{108}$$

unde β_{kj}, definiţi pentru $k > j$, sunt deduşi din condiţiile de A-ortogonalitate impuse direcţiilor, rezultând

$$\beta_{kj} = -\frac{(\mathbf{u}^{(k)})^T\mathbf{A}\mathbf{d}^{(j)}}{(\mathbf{d}^{(j)})^T\mathbf{A}\mathbf{d}^{(j)}}. \tag{109}$$

Dificultate: toate direcţiile de căutare trebuie memorate pentru a construi o direcţie de căutare nouă.

Notes

Metoda gradienţilor conjugaţi

Metoda gradienţilor conjugaţi este metoda direcţiilor conjugate în care direcţiile de căutare sunt construite prin conjugarea reziduurilor:

$$\mathbf{u}^{(k)} = \mathbf{r}^{(k)}. \tag{110}$$

Notes

Metoda gradienţilor conjugaţi

Justificarea acestei alegeri:

1. Intuitiv, inspirat de metoda celei mai rapide coborâri, în care direcţiile de căutare erau direcţiile reziduurilor.
2. Proprietatea reziduului de a fi ortogonal pe direcţiile de căutare anterioare şi pe reziduurile anterioare.

$$(\mathbf{d}^k)^T \mathbf{r}^{(i)} = 0, \quad k < i, \tag{111}$$

$$(\mathbf{r}^k)^T \mathbf{r}^{(i)} = 0, \quad k < i. \tag{112}$$

Notes

Metoda gradienţilor conjugaţi

De asemenea, reziduul $\mathbf{r}^k$ este o combinaţie liniară dintre reziduul anterior şi produsul $\mathbf{Ad}^{(k-1)}$:

$$\mathbf{r}^{(k)} = -\mathbf{Ae}^{(k)} = -\mathbf{A}(\mathbf{e}^{(k)} + \alpha_k \mathbf{d}^{(k)}) = \mathbf{r}^{(k-1)} - \alpha_{k-1}\mathbf{Ad}^{(k-1)}. \quad (113)$$

Notes

Metoda gradienţilor conjugaţi

Coeficienţii β

$$\beta_{kj} = -\frac{(\mathbf{r}^{(k)})^T \mathbf{Ad}^{(j)}}{(\mathbf{d}^{(j)})^T \mathbf{Ad}^{(j)}}. \quad (114)$$

Pentru a explicita termenul de la numărător, vom înmulţi relaţia (113) la stânga cu $(\mathbf{r}^{(i)})^T$:

$$(\mathbf{r}^{(i)})^T \mathbf{r}^{(k+1)} = (\mathbf{r}^{(i)})^T \mathbf{r}^{(k)} - \alpha_k (\mathbf{r}^{(i)})^T \mathbf{Ad}^{(k)}, \quad (115)$$

de unde

$$\alpha_k (\mathbf{r}^{(i)})^T \mathbf{Ad}^{(k)} = (\mathbf{r}^{(i)})^T \mathbf{r}^{(k)} - (\mathbf{r}^{(i)})^T \mathbf{r}^{(k+1)}. \quad (116)$$

Notes

Metoda gradienţilor conjugaţi

Pentru $i \neq k$, membrul drept al acestei relaţii este nenul doar pentru cazul $i = k + 1$, când valoarea lui este $-(\mathbf{r}^{(i)})^T \mathbf{r}^{(i)} / \alpha_{i-1}$. Rezultă că valorile β sunt nenule doar dacă $i = k + 1$:

$$\beta_{kj} = \begin{cases} \frac{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}}{\alpha_{k-1}(\mathbf{d}^{(k-1)})^T \mathbf{A}\mathbf{d}^{(k-1)}} & k = i - 1, \\ 0 & k < i - 1 \end{cases} \tag{117}$$

Notes

Metoda gradienţilor conjugaţi

Nu mai este necesar ca valorile β să fie notate cu doi indici. Vom renota

$$\begin{aligned} \beta_k &= \beta_{k,k-1} = \\ &= \frac{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}}{\alpha_{k-1}(\mathbf{d}^{(k-1)})^T \mathbf{A}\mathbf{d}^{(k-1)}} = \\ &= \frac{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{d}^{(k-1)})^T \mathbf{r}^{(k-1)}} = \\ &= \frac{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{r}^{(k-1)})^T \mathbf{r}^{(k-1)}}. \end{aligned} \tag{118}$$

Notes

Metoda gradienţilor conjugaţi

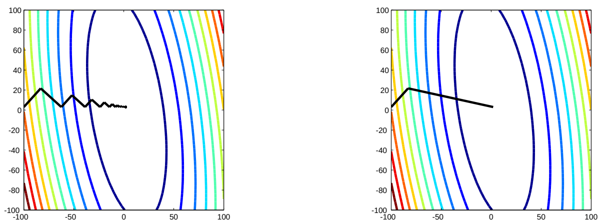
În concluzie, metoda gradienţilor conjugaţi se bazează pe următoarele şase relaţii:

$$\begin{aligned}
\mathbf{d}^{(0)} &= \mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} \\
\alpha_k &= \frac{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}}{(\mathbf{d}^{(k)})^T \mathbf{A}\mathbf{d}^{(k)}} \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)} \\
\mathbf{r}^{(k+1)} &= \mathbf{r}^{(k)} - \alpha_k \mathbf{A}\mathbf{d}^{(k)} \\
\beta_{k+1} &= \frac{(\mathbf{r}^{(k+1)})^T \mathbf{r}^{(k+1)}}{(\mathbf{r}^{(k)})^T \mathbf{r}^{(k)}} \\
\mathbf{d}^{(k+1)} &= \mathbf{r}^{(k+1)} + \beta_{k+1} \mathbf{d}^{(k)}
\end{aligned}$$

Notes

Metoda gradienţilor conjugaţi

Algoritmul este complet după *n* iteraţii.



Convergenţa metodei gradientului depinde de iniţializare şi de numărul de condiţionare spectrală (stânga). Metoda gradienţilor conjugaţi converge în exact *n* paşi, indiferent de iniţializare şi de numărul de condiţionare spectrală (dreapta).

Notes

Metoda gradienţilor conjugaţi

```
procedură metoda_gradienţilor_conjugaţi(n, A, b, x0, er, maxit, x)
...
xv = x0                        ; iniţializează şirul iteraţiilor
k = 0                          ; iniţializare contor iteraţii
r = b − A · xv                 ; calculează reziduu
cât timp ||r|| > er şi k ≤ maxit
        dacă k = 0
                d = r
        altfel
                β = r^T r/(rv^T · rv)
                d = r + β dv
        •
        α = r^T r/(d^T Ad)
        x = xv + α d
        rv = r
        r = rv − α Ad
        xv = x
        dv = d
        k = k + 1
•
retur
```

Algoritmul se poate îmbunătăţi, calculând la o iteraţie o singură dată produsul matrice - vector **Ad** şi produsul scalar $\mathbf{r}^T\mathbf{r}$.

Notes

Metoda gradienţilor conjugaţi

- Algoritmul este complet după n iteraţii. Din acest motiv, se mai spune că metoda este semi-iterativă, şirul iteraţiilor nu tinde către soluţia exactă atunci când numărul iteraţiilor tinde la infinit, ci, într-o aritmetică exactă, soluţia exactă se obţine după exact n iteraţii.
- În practică însă metoda este folosită pentru probleme atât de mari încât nu ar fi fezabil să se execute toate cele n iteraţii. De aceea, iteraţiile în algoritm sunt executate într-un ciclu cu test şi nu într-un ciclu cu contor.

Notes

Precondiţionare

Precondiţionare = transformarea problemei iniţiale într-una echivalentă care are proprietaţi îmbunătăţite considerabil.
Precondiţionare la stânga

$$\mathbf{Ax} = \mathbf{b} \quad (119)$$

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}, \quad (120)$$

M matrice nesingulară, numită *matrice de precondiţionare* sau *precondiţionator*.

- Convergenţa depinde de proprietăţile matricei $\mathbf{M}^{-1}\mathbf{A}$.
- $\mathbf{M}^{-1}\mathbf{A}$ nu se calculează explicit, ci se rezolvă $\mathbf{My} = \mathbf{A}$
- Cazurile extreme: $\mathbf{M} = \mathbf{I}$ şi $\mathbf{M} = \mathbf{A}$ - nu există nici un câştig.
- O condiţie puternică pentru un bun precondiţionator este ca valorile proprii ale matricei $\mathbf{M}^{-1}\mathbf{A}$ să fie apropiate de 1 şi ca norma $\|\mathbf{M}^{-1}\mathbf{A} - \mathbf{I}\|_2$ să fie mică [Trefethen97].

Notes

Precondiţionare

Precondiţionare la dreapta

$$\mathbf{AM}^{-1}\mathbf{y} = \mathbf{b}, \quad (121)$$

unde $\mathbf{x} = \mathbf{M}^{-1}\mathbf{y}$.
Se pot folosi ambele tipuri de precondiţionare simultan.
Precondiţionare matricelor simetrice şi pozitiv definite
Precondiţionarea se face astfel încât să se păstreze această proprietate.
Fie $\mathbf{M} = \mathbf{CC}^T$ - simetrică şi pozitiv definită.

$$\left[\mathbf{C}^{-1}\mathbf{AC}^{-T}\right]\mathbf{C}^T\mathbf{x} = \mathbf{C}^{-1}\mathbf{b}, \quad (122)$$

unde $\mathbf{C}^{-T} = (\mathbf{C}^{-1})^T = (\mathbf{C}^T)^{-1}$, iar matricea din paranteza dreaptă este simetrică şi pozitiv definită.

Notes

Precondiţionare

Metode de precondiţionare celebre

1. *Precondiţionarea Jacobi (sau scalarea diagonală)*: $\mathbf{M} = \mathrm{diag}(\mathbf{A})$, nesingulară. Generalizare: $\mathbf{M} = \mathrm{diag}(\mathbf{c})$, unde $\mathbf{c}$ ales convenabil.
2. *Factorizarea incompletă Cholesky sau LU* în care se aplică procedura de factorizare, dar factorii nu sunt calculaţi complet, valorile care ar umple matricea nu sunt luate în considerare. Matricea de precondiţionare se obţine ca produsul acestor factori incompleţi.

Notes

Precondiţionare

- Aceste două exemple de precondiţionatori nu fac nicio referire la problema iniţială care a generat sistemul (119).
- Cel mai bun sfat general: de a încerca să se construiască precondiţionatorul pe baza unei versiuni mai simple a problemei. Pe această idee se bazează *metodele multigrid* care se bazează şi pe precondiţionatori obţinuţi din Jacobi şi SSOR.

Notes

Referinţe

- [Ciuprina13a] Gabriela Ciuprina - Algoritmi numerici pentru calcule ştiinţifice în ingineria electrică , Editura MatrixROM, 2013, pag. 88-120.
  disponibilă la http://www.lmn.pub.ro/∼gabriela/books/AlgNr_MatrixRom2013.pdf
- [Saad03] Yousef Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.
- [Barrett94] Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June M. Donato, Jack Dongarra, Victor Eijkhout,Roldan Pozo, Charles Romine, and Henk Van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM 1994.
  disponibilă la http://www.netlib.org/templates/templates.pdf
- [Shewchuk94] J.R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*
  disponibilă online aici
- [Trefethen97] Lloyd N. Trefethen, David Bau III *Numerical Linear Algebra*, SIAM 1997.

Notes

Biblioteci existente

Biblioteci existente:

- [Dongarra2016] Freely Available Software for Linear Algebra (September 2016), http://www.netlib.org/utk/people/JackDongarra/la-sw.html
- [Eijkhout97] Overview of Iterative Linear System Solver Packages http://www.netlib.org/utk/papers/iterative-survey/

BDDCML, BILUM, BlockSolve95, CERFACS, DUNE/ISTL, GMM++, HIPS, HYPRE, IML++, ITL, ITPACK, ITSOL, Lis, PARALUTION, pARMS, PETSc, PIM, QMRPACK, SLAP, SOL, SPARSKIT, SPLIB, Templates, Trilinos

Notes

Referinţe

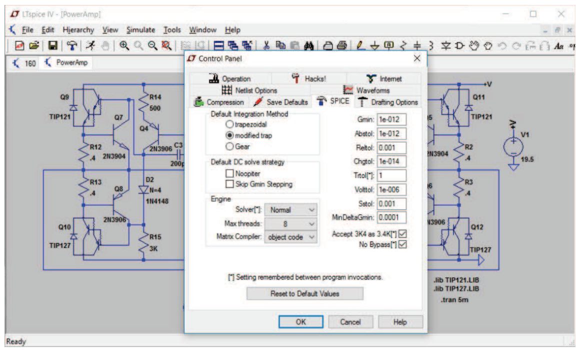
www.netlib.org/utk/people/JackDongarra/la-sw.html

SPARSE ITERATIVE SOLVERS	License	Support	Type		Language			Mode			Dense	Sparse Direct			Sparse Iterative		Sparse Eigenvalue		Last release date
			Real	Complex	F77/ F95	C	C++	Shared	Accel.	Dist		SPD	SI	Gen	SPD	Gen	Sym	Gen	
BDDCML	LGPL	yes	X		X			X		M					X	X			2014-03-12
BILUM	Own	yes	X		X			X							X	X			1998-03-18
BlockSolve95	Own	?	X		X	X	X			M					X	X			1997-07-08
CERFACS	?	yes	X	X	X			X							X	X			2007-07-01
DUNE / ISTL	GPL	yes	X	X			X	X		M					X	X			2014-12-18
GMM++	LGPL	yes	X	X			X	X				X		X	X	X			2014-08-21
HIPS	CeCILL-C	yes	X	X	X	X	X	X		M					X	X			2010-10-13
HYPRE	LGPL	yes	X		X	X		X		M					P	P	P		2015-01-22
IML++	PD	?	X				X	X							X	X			1995-01-05
ITL	Own	yes	X				X	X							X	X			2001-10-26
ITPACK	PD	?	X		X			X							X	X			1989-05-01
ITSOL	GPL	yes	X		X			X								X			2012-10-25
Lis	BSD	yes	X	X	X	X		X		M					P	P	P	P	2015-05-14
PARALUTION	GPL	yes	X				X	X	C/O						X	X			2015-02-27
pARMS	LGPL	yes	X		X	X		X		M					X	X			2011-01-14
PETSc	Own	yes	X	X	X	X		X	C/O	M					P	P			2015-01-31
PIM	Own	yes	X	X	X			X		M					X	X			2003-06-20
QMRPACK (tar.gz)	Own	?	X	X	X			X							X	X	X	X	1996-04-15
SLAP	PD	?	X		X										X	X			1998-07-21
SOL	CPL or BSD	yes	X		X	X		X							X	X			2015-05-14
SPARSKIT	LGPL	yes	X		X			X								X			2009-11-18
SPLIB (tar.gz)	Own	?	X		X			X							X	X			1999-04-01
Templates	BSD	yes	X		X	X		X							X	X			1998-07-21
Trilinos/AztecOO	BSD	yes	X		X	X	X	X		M					X	X			2015-05-07
Trilinos/Belos	BSD	yes	X	X			X	X		M					X	X			2015-05-07

Notes

Pe scurt

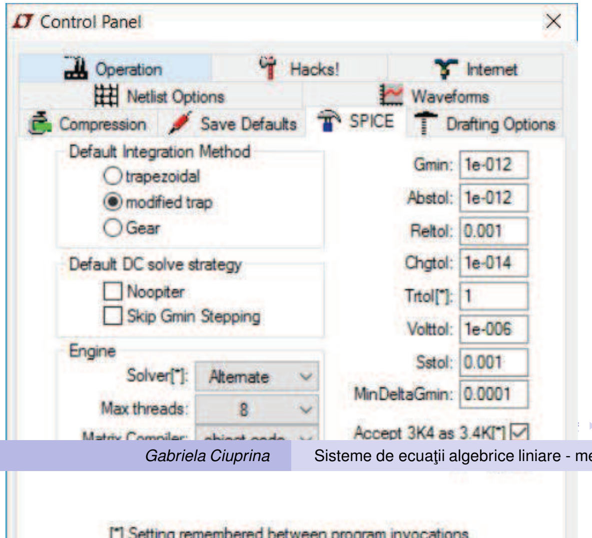
Fiţi atenţi la astfel de informaţii (capturi din LTSPICE).
Consultaţi documentaţia pentru detalii!



Notes

Pe scurt

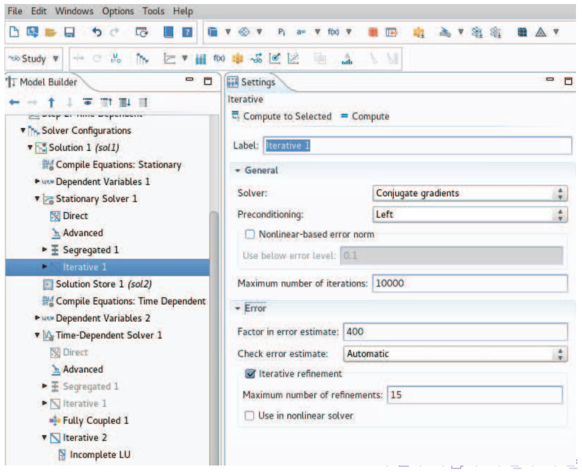
Fiţi atenţi la astfel de informaţii (capturi din LTSPICE).
Consultaţi documentaţia pentru detalii!



Notes

Pe scurt

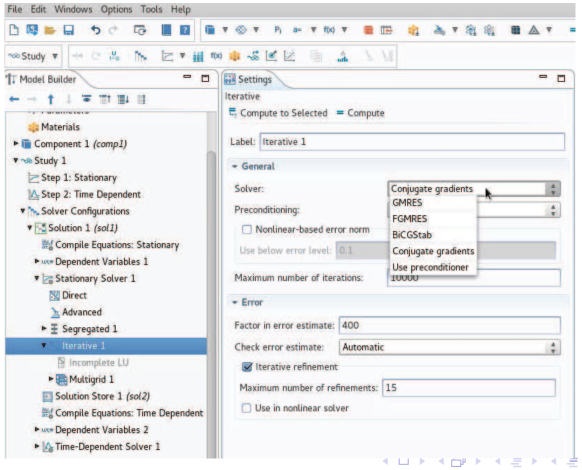
Fiţi atenţi la astfel de informaţii (capturi din COMSOL)



Notes

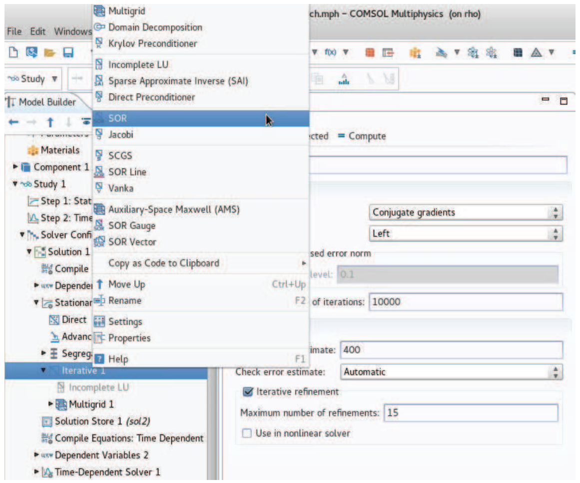
Pe scurt

Fiţi atenţi la astfel de informaţii (capturi din COMSOL)



Notes

Pe scurt



Notes