

Sisteme de ecuații algebrice liniare - metode iterative

Prof.dr.ing. Gabriela Ciuprina

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

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

Cuprins

- 1 Formularea problemei
- 2 Metode staționare
 - Ideea
 - Metoda Jacobi
 - Metoda Gauss-Seidel
 - SOR
- 3 Metoda gradientilor conjugați
 - Ideea metodei gradientilor conjugați
 - Metoda gradientului
 - Metoda direcțiilor conjugate
 - Metoda gradientilor conjugați
- 4 Precondiționare
 - Referințe
 - Biblioteci existente

Formularea problemei

Sistem de n ecuații algebrice liniare cu n necunoscute:

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

Formularea problemei

Se dă matricea coeficienților

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

și vectorul termenilor liberi

$$\mathbf{b} = [b_1 \quad b_2 \quad \cdots \quad b_n]^T \in \mathbb{R}^n, \quad (3)$$

se cere să se rezolve sistemul

$$\mathbf{Ax} = \mathbf{b}, \quad (4)$$

unde \mathbf{x} este soluția

$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_n]^T \in \mathbb{R}^n. \quad (5)$$

Buna formulare matematică

Problema este bine formulată din punct de vedere matematic (soluția există și este unică)



matricea **A** este nesingulară (are determinantul nenul).

Se scrie formal:

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

trebuie citită ca:

"x este soluția sistemului algebric liniar $\mathbf{Ax} = \mathbf{b}$ "

*și **NU** "se calculează inversa matricei **A** care se înmulțește cu vectorul **b**".*

Condiționarea problemei

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

număr de condiționare la inversare al matricei \mathbf{A} .

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

- $\kappa(\mathbf{A}) \geq 1$:

Cazul cel mai favorabil: $n_A = 1$ și $\varepsilon_x = \varepsilon_b$. (matrice ortogonală)

- Numărul de condiționare este o proprietate a matricei și nu are legătură nici cu metoda de rezolvare propriu-zisă, nici cu erorile de rotunjire care apar în mediul de calcul.

În practică:

Dacă $\kappa(\mathbf{A}) > 1/\text{eps}$ problema se consideră slab condiționată.

Clasificarea metodelor

- 1 **Metode directe** - găsesc soluția teoretică a problemei într-un **număr finit de pași**. (Gauss, factorizare LU)
- 2 **Metode iterative** - generează un **șir de aproximații** ale soluției care se dorește a fi convergent către soluția exactă.
 - **staționare**: Jacobi, Gauss-Seidel, SOR, SSOR
 - **nestaționare (semiiterative)**: gradienti conjugați (GC), reziduu minim (MINRES), reziduu minim generalizat (GMRES), gradienti biconjugați (BiGC), etc.

Ideea metodelor staționare

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

se construiește un șir de aproximații
 $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}, \dots$

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

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

Ideea metodelor staționare

Algoritmul are nevoie de

- 1 o inițializare $\mathbf{x}^{(0)}$;
- 2 un mod de generare a șirului de iterații;
- 3 un criteriu de oprire.

1. Inițializarea

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

Ideea metodelor staționare

2. Șirul de iterații se generează *recursiv*:

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

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

\mathbf{x}^* este punct fix pentru aplicația F .

În concluzie, soluția exactă a sistemului de ecuații este și punct fix pentru F . Rezolvarea sistemului de ecuații algebrice liniare se face prin căutarea unui punct fix pentru F .

Ideea metodelor staționare

$$\mathbf{A} = \mathbf{B} - \mathbf{C}. \quad (12)$$

$$\mathbf{B}\mathbf{x} = \mathbf{C}\mathbf{x} + \mathbf{b} \quad (13)$$

$$\mathbf{x} = \mathbf{M}\mathbf{x} + \mathbf{u}, \quad (14)$$

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

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

$$\mathbf{F}(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{u}, \quad (16)$$

Ideea metodelor staționare

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

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

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

$$\mathbf{B}\mathbf{x}^{(k)} = \mathbf{C}\mathbf{x}^{(k-1)} + \mathbf{b}, \quad (20)$$

B are o structură particulară.

Ideea metodelor staționare

3. Criteriul de oprire

Condiție de oprire bazată de criteriul Cauchy de convergență:

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

Se poate întâmpla însă ca șirul iterațiilor 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 iterațiilor;
- un număr maxim de iterații, util pentru a asigura oprirea naturală a procedurii în caz de neconvergență.

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

Util: vectori si valori proprii

Definiție: vectorii proprii \mathbf{v} ai unei matrice pătrate reale \mathbf{M} , de dimensiune n sunt acei vectori nenuli, pentru care există un scalar λ astfel încât

$$\mathbf{M}\mathbf{v} = \lambda\mathbf{v}. \quad (22)$$

Obs:

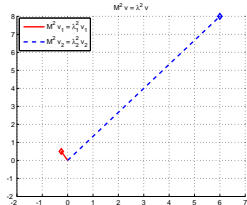
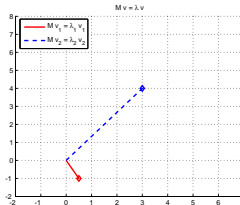
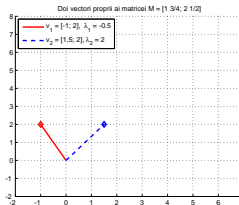
- Reprezentarea geometrică: prin aplicarea \mathbf{M} asupra lui, vectorul nu se rotește;
- Vectorii proprii ai unei matrice nu sunt unici. Dacă \mathbf{v} este un vector propriu, atunci și vectorul scalat $\alpha\mathbf{v}$ este de asemenea vector propriu;
- λ se numește valoare proprie a matricei \mathbf{M} asociată vectorului propriu \mathbf{v} .

Util: vectori si valori proprii

$$\mathbf{M}^k \mathbf{v} = \lambda^k \mathbf{v}. \quad (23)$$

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

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



Înmulțirea repetată dintre o matrice și vectorii ei proprii.

Util: vectori si valori proprii

Dacă \mathbf{M} este simetrică, atunci ea are n vectori proprii liniar independenți:

$$\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots, \mathbf{v}^{(n)}.$$

Această mulțime nu este unică, dar fiecare vector propriu are asociat o valoare proprie unică.

Dacă cei n vectori proprii ai lui \mathbf{M} formează o bază, atunci orice vector $\mathbf{u} = \sum_{i=1}^n \alpha_i \mathbf{v}^{(i)}$.

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

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

Util: vectori si valori proprii

Raza spectrală a unei matrice: proprii

$$\rho(M) = \max_i |\lambda_i|. \quad (25)$$

Valorile proprii sunt rădăcinile polinomului caracteristic.
Deoarece

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

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

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

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

Convergență

Teorema 1: Condiția necesară și suficientă

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

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

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

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

Convergență

Teorema 2: O condiție suficientă

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

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

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

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

Convergență

Mai mult

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

⇒ Utilizarea unui criteriu de oprire Cauchy este pe deplin justificată.

Convergență

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

$$a_k = \|\mathbf{M}\|^k a_0, \quad (33)$$

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

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

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

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

Convergență

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

Rata de convergență = numărul de cifre semnificative corecte ce se câștigă la fiecare iterație.

Exemplu:

- $\|\mathbf{M}\| = 10^{-3}$, rata de convergență este 3, deci la fiecare iterație numărul de cifre semnificative corecte crește cu 3.
- $\|\mathbf{M}\| = 10^{-1}$, la fiecare iterație se câștigă o cifră semnificativă.

OBS:

- Alegerea valorii inițiale nu are nici o influență asupra convergenței sau neconvergenței procesului iterativ;
- În cazul unui proces iterativ convergent, valoarea inițială afectează doar numărul de iterații necesar pentru atingerea unei erori impuse.

Algoritm general

procedură metodă_iterativă($n, B, C, b, x_0, er, maxit, x$)

...

$\mathbf{xv} = \mathbf{x0}$

; inițializează șirul iterațiilor

$k = 0$

; inițializare contor iterații

repetă

$\mathbf{t} = \mathbf{C} * \mathbf{xv} + \mathbf{b}$

metodă_directă (n, B, t, x)

$d = \|\mathbf{xv} - \mathbf{x}\|$

$\mathbf{xv} = \mathbf{x}$

; actualizează soluția veche

$k = k + 1$

cât timp $d > er$ și $k \leq maxit$

retur

Algoritm general

Efortul de calcul

- poate fi făcut doar pentru o singură iterație;
- depinde de structura matricelor în care a fost descompusă matricea coeficienților;
- e consumat mai ales în calculul lui \mathbf{t} și în procedura de rezolvare directă, care în general are o complexitate liniară deoarece \mathbf{B} are o structură rară, particulară.
- este de așteptat ca procedeul iterativ să fie cu atât mai rapid convergent cu cât \mathbf{B} conține mai multă informație din \mathbf{A} .

Metoda Jacobi: un exemplu simplu

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

$$\begin{cases} x + 2y - z = -1 \\ -2x + 3y + z = 0 \\ 4x - y - 3z = -2 \end{cases} \Leftrightarrow \begin{cases} x = -2y + z - 1 \\ 3y = 2x - z \\ -3z = -4x + y - 2 \end{cases} \quad (38)$$

$$\begin{aligned} x^{(n)} &= -2y^{(v)} + z^{(v)} - 1 \\ y^{(n)} &= 2x^{(v)} - z^{(v)} \\ z^{(n)} &= -4x^{(v)} + y^{(v)} - 2 \end{aligned} \quad (39)$$

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

Algoritmul metodei Jacobi

Partiționarea matricei în metodele iterative:

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

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

$\mathbf{A} = \mathbf{B} - \mathbf{C}$ unde, în metoda Jacobi

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

Algoritmul metodei Jacobi

Calculul recursiv al noii iterații

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

Ecuția i :

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

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

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

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 coeficienților, indici de la 1  
tablou real b[n] ; vectorul termenilor liberi  
; mărimi specifice procedurilor iterative  
tablou real x0[n] ; inițializarea soluției  
real er ; eroarea folosită de criteriul de oprire  
întreg maxit ; număr maxim de iterații admis  
tablou real xv[n] ; aproximația anterioară ("veche")
```

Algoritmul metodei Jacobi

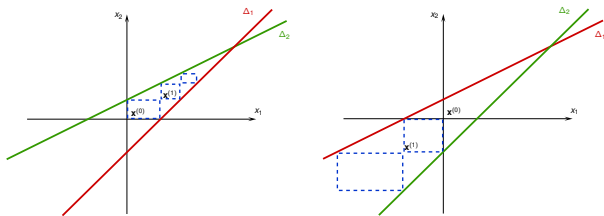
```

procedură Jacobi( $n, a, b, x_0, er, maxit, x$ )
.....
pentru  $i = 1, n$ 
     $xv_i = x_{0i}$ 
    •
     $k = 0$  ; inițializare contor iterații
    repetă
         $d = 0$ 
        pentru  $i = 1, n$ 
             $s = 0$ 
            pentru  $j = 1, n$ 
                dacă  $j \neq i$ 
                     $s = s + a_{ij} * xv_j$ 
                •
             $x_i = (b_i - s) / a_{ii}$ 
             $d = d + (x_i - xv_i)^2$ 
        •
         $d = \sqrt{d}$ 
        pentru  $i = 1, n$ 
             $xv_i = x_i$  ; actualizează soluția veche
        •
         $k = k + 1$ 
    cât timp  $d > er$  și  $k \leq maxit$ 
    retur
    
```

Convergența metodei Jacobi

Matricea de iterație

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



Schimbarea ordinii ecuațiilor din sistem înseamnă schimbarea matricei de iterație, deci a proprietăților de convergență ale metodei Jacobi.

Rezultat util: Dacă matricea coeficienților este diagonal dominantă, atunci condiția suficientă de convergență este satisfăcută și algoritmul Jacobi este convergent.

Metoda Gauss-Seidel: un exemplu simplu

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

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

$$\begin{cases} x^{(n)} = -2y^{(v)} + z^{(v)} - 1 \\ -2x^{(n)} + 3y^{(n)} + z^{(v)} = 0 \\ 4x^{(n)} - y^{(n)} - 3z^{(n)} = -2 \end{cases} \quad (46)$$

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

Algoritmul metodei Gauss-Seidel

Partiționarea matricei în metodele iterative:

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

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

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

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

Algoritmul metodei Gauss-Seidel

Calculul recursiv al noii iterații

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

Ecuția i :

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

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

Rezolvarea sistemului: prin substituție progresivă, conform formulei:

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

Algoritmul metodei Gauss-Seidel

Observații:

- 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 numește și **metoda deplasărilor succesive**

Algoritmul metodei Gauss-Seidel

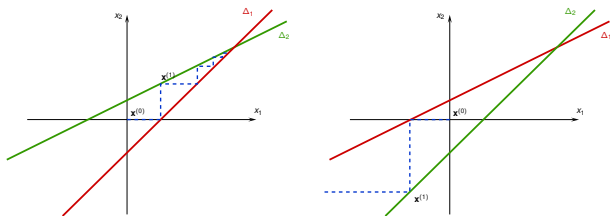
```

procedură Gauss-Seidel( $n, a, b, x0, er, maxit, x$ )
; rezolvă sistemul algebric liniar  $ax = b$ , de dimensiune  $n$  prin metoda Gauss-Seidel
; declarații ca la procedura Jacobi
...
pentru  $i = 1, n$ 
     $xv_i = x0_i$ 
    •
     $k = 0$  ; inițializare contor iterații
    repetă
         $d = 0$ 
        pentru  $i = 1, n$ 
             $s = b_i$ 
            pentru  $j = 1, n$ 
                dacă  $j \neq i$ 
                     $s = s + a_{ij} * xv_j$ 
                    •
                •
             $s = s / a_{ii}$ 
             $p = |x_i - s|$ 
            dacă  $p > d$ 
                 $d = p$ 
            •
             $x_i = s$ 
        •
         $k = k + 1$ 
    cât timp  $d > er$  și  $k \leq maxit$ 
    retur
    
```

Convergența metodei Gauss-Seidel

Matricea de iterație

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



Schimbarea ordinii ecuațiilor din sistem înseamnă schimbarea matricei de iterație, deci a proprietăților de convergență ale metodei Gauss-Seidel.

Rezultate utile:

- Dacă matricea coeficienților este diagonal dominantă, atunci algoritmul Gauss-Seidel este convergent.

Evaluarea algoritmilor Jacobi și Gauss-Seidel

Efortul total de calcul depinde de numărul de iterații m (care depinde de matricea de iterație).

- Efortul de calcul pe iterație este $O(2n^2)$.
- Efortul total de calcul al algoritmilor Jacobi și Gauss-Seidel *implementați 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):

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

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

diferența este nesemnificativă

Evaluarea algoritmilor Jacobi și Gauss-Seidel

Observații:

- 1 Dacă matricea coeficienților este rară (memorată MM, CRS sau CCS), atunci efortul de calcul pe iterație se poate diminua.
- 2 Important: *Nu putem vorbi de umpleri ale matricei coeficienților* așa 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.

Metoda Suprarelaxării succesive (SOR)

Procedeu de accelerare a convergenței:

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

unde ω se numește **factor de suprarelaxare**

$\omega = 1$ corespunde metodei Gauss-Seidel.

Modificarea în pseudocodul algoritmului Gauss-Seidel:

instrucțiunea $x_i = s$ se înlocuiește cu $x_i = x_i + \omega(s - x_i)$.

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

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

Metoda Suprarelaxării succesive (SOR)

Matricea de iterație a metodei SOR:

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

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

Metoda Suprarelaxării succesive (SOR)

- Pentru matricile obținute prin discretizarea unor ecuații cu derivate parțiale prin parcurgerea sistematică a nodurilor rețelei de discretizare se poate demonstra că

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

unde ρ este raza spectrală a matricei de iterație 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 preconditionare pentru metode nestaționare.

Algoritmul general al metodelor staționare

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

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

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

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

Reziduul la iterația k

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

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

Algoritmul general al metodelor staționare

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

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

Staționaritatea se referă la faptul că reziduu este întotdeauna înmulțit cu matricea \mathbf{B}^{-1} , aceeași pe parcursul tuturor iterațiilor:

- 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.

Algoritmul general al metodelor staționare

procedură metodă_iterativă_v2($n, B, A, b, x_0, er, maxit, x$)

...

$\mathbf{xv} = \mathbf{x0}$

; inițializează șirul iterațiilor

$k = 0$

; inițializare contor iterații

repetă

$\mathbf{r} = \mathbf{b} - \mathbf{A} \cdot \mathbf{xv}$

; calculează reziduu

metodă_directă (n, B, r, z)

$d = \|\mathbf{z}\|$

$\mathbf{x} = \mathbf{xv} + \mathbf{z}$

$\mathbf{xv} = \mathbf{x}$

; actualizează soluția veche

$k = k + 1$

cât timp $d > er$ și $k \leq maxit$

retur

Algoritmul general al metodelor staționare

- 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 situație, factorizarea este mai utilă deoarece, o dată ce factorii L și U sunt calculați, rezolvarea sistemului poate fi făcută oricând pentru alt termen liber.
- De aceea, un pseudocod simplificat, în care sunt scrise operații cu matrice este mai general, putând fi adaptat unor matrice rare.

Clasa de probleme

Este o metodă iterativă nestaționară, pentru rezolvarea unor sisteme de ecuații algebrice de forma

$$\mathbf{Ax} = \mathbf{b}, \quad (64)$$

unde \mathbf{A} este **simetrică și pozitiv definită**.

- Algoritmul metodei gradientilor conjugați (GC) este eficient pentru **matrice rare**.

⇒ Pseudocodurile vor fi descrise simplificat, evidențiind operații de algebră liniară, presupuse a fi implementate ținând cont de raritatea structurilor de date.

Forma pătratică

Metoda GC este simplu de înțeles dacă ea este formulată ca o problemă de minimizare.

Fie o formă pătratică

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$
$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{b}^T \mathbf{x} + \mathbf{c}, \quad (65)$$

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

Teoremă

Dacă \mathbf{A} este simetrică și pozitiv definită, atunci funcția $f(\mathbf{x})$ dată de (65) este minimizată de soluția ecuației $\mathbf{A} \mathbf{x} = \mathbf{b}$.

Forma pătratică

Justificarea teoremei:

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

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

În punctul de minim gradientul este zero

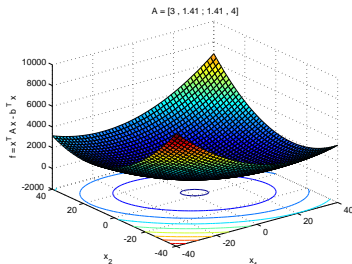
$$\nabla f(\mathbf{x}) = 0 \quad \Rightarrow \quad \frac{1}{2}(\mathbf{A}^T + \mathbf{A})\mathbf{x} = \mathbf{b}. \quad (68)$$

Dacă $\mathbf{A}^T = \mathbf{A}$ (matrice simetrică) atunci (68) $\Leftrightarrow \mathbf{A} \mathbf{x} = \mathbf{b}$.

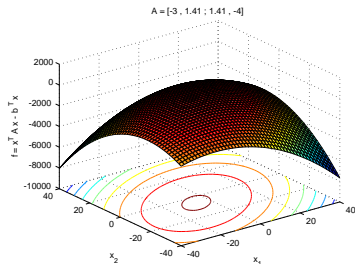
\Rightarrow soluția ecuației $\mathbf{A} \mathbf{x} = \mathbf{b}$ este punct critic pentru f .

Dacă în plus, \mathbf{A} pozitiv definită, atunci pct.critic este un minim.

Matrice pozitiv definită - intuitiv

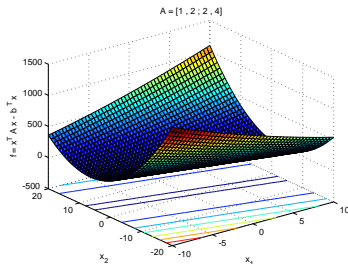


Funcția pătratică asociată unei matrice pozitiv definite. Minimul funcției coincide cu soluția sistemului. GC funcționează.

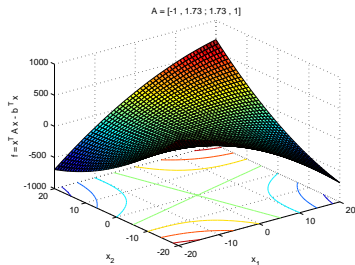


Funcția pătratică asociată unei matrice negativ definite. Maximul funcției coincide cu soluția sistemului. Algoritm GC poate fi modificat cu ușurință pentru a putea funcționa.

Matrice pozitiv semidefinită/indefinită - intuitiv



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



Funcția pătratică asociată unei matrice indefinite. Soluția sistemului este punct șă pentru f . GC nu funcționează.

Metoda celei mai rapide coborâri (a gradientului)

$$\mathbf{Ax} = \mathbf{b}, \quad (69)$$

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

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

Ideea: căutarea pe direcții care corespund ratei celei mai mari de schimbare a funcției.

Pentru aproximația $\mathbf{x}^{(k)}$, **direcția celei mai rapide coborâri:**

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

Se definesc:

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

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

Metoda celei mai rapide coborâri (a gradientului)

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

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}. \quad (76)$$

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}(\mathbf{e}^{(k)} + \mathbf{x}) = \mathbf{b} - \mathbf{A}\mathbf{e}^{(k)} - \mathbf{A}\mathbf{x}.$$

Legătura între eroare și reziduu:

$$\mathbf{r}^{(k)} = -\mathbf{A}\mathbf{e}^{(k)}. \quad (77)$$

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

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

Reziduu este direcția celei mai rapide coborâri:

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

Metoda celei mai rapide coborâri (a gradientului)

Corecția primei iterații: după direcția reziduuului inițializării:

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

α se va alege a.î. pe direcția respectivă funcția să fie minimă.

$g(\alpha) = f(\mathbf{x}^{(1)})$ minimă, \Rightarrow

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

Din (79) $\Rightarrow \nabla f(\mathbf{x}^{(1)}) = -\mathbf{r}^{(1)}$;

Din (80) $\Rightarrow d\mathbf{x}^{(1)}/d\alpha = \mathbf{r}^{(0)}$.

(81) \Rightarrow

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

A simetrică \Rightarrow

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

Metoda celei mai rapide coborâri (a gradientului)

Corecția iterației $k + 1$:

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

α_k se va alege a.î.:

$g(\alpha_k) = f(\mathbf{x}^{(k+1)})$ minimă, \Rightarrow

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

Din (79) $\Rightarrow \nabla f(\mathbf{x}^{(k+1)}) = -\mathbf{r}^{(k+1)}$;

Din (83) $\Rightarrow d\mathbf{x}^{(k+1)}/d\alpha_k = \mathbf{r}^{(k)}$.

(84) \Rightarrow ortogonalitatea reziduurilor:

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

A simetrică \Rightarrow

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

Metoda celei mai rapide coborâri (a gradientului)

Algoritm - varianta 0

procedură metoda_gradientului_v0($n, A, b, x0, er, maxit, x$)

...

$xv = x0$

; inițializează șirul iterațiilor

$k = 0$

; inițializare contor iterații

$r = b - A \cdot xv$

; calculează reziduu

cât timp $\|r\| > er$ și $k \leq maxit$

$$\alpha = r^T r / (r^T A r)$$

$$x = xv + \alpha r$$

$$r = b - A \cdot x$$

$$xv = x$$

; actualizează soluția veche

$$k = k + 1$$

•

retur

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

Metoda celei mai rapide coborâri (a gradientului)

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

Înmulțim la stânga cu $-\mathbf{A}$ și adunăm $\mathbf{b} \Rightarrow$

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

Metoda celei mai rapide coborâri (a gradientului)

procedură metoda_gradientului($n, A, b, x_0, er, maxit, x$)

...

$xv = x_0$

; inițializează șirul iterațiilor

$k = 0$

; inițializare contor iterații

$r = b - A \cdot xv$

; calculează reziduu

$e = \|r\|$

cât timp $e > er$ și $k \leq maxit$

$m = A \cdot r$

$\alpha = r^T r / (r^T m)$

$e = \|r\|$

$x = xv + \alpha r$

; actualizează soluția veche

$xv = x$

$k = k + 1$

$r = r - \alpha m$

•

retur

Metoda celei mai rapide coborâri (a gradientului)

Observații:

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

Metoda celei mai rapide coborâri (a gradientului)

Analiza convergenței acestei metode se face pe baza
numărului de condiționare spectrală

$$\kappa = \frac{\max \lambda_j}{\min \lambda_j} \geq 1. \quad (88)$$

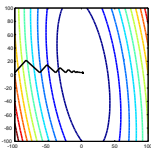
- Metoda poate converge rapid, într-un singur pas, dacă punctul de start este ales pe axele elipsoidului sau dacă forma pătratică este sferică (ceea ce corespunde cazului în care toate valorile proprii sunt egale).

Metoda celei mai rapide coborâri (a gradientului)

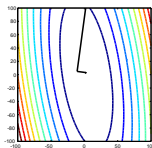
- În rest, convergența depinde de numărul de condiționare κ și de inițializare.
- Se poate demonstra că eroarea la fiecare iterație i este mărginită de

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

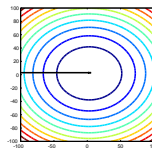
Metoda celei mai rapide coborâri (a gradientului)



$$\kappa = 10$$



$$\kappa = 10$$



$$\kappa = 1$$

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

Metoda direcțiilor conjugate

Metoda gradientului converge atât de încet deoarece direcțiile de căutare sunt ortogonale și se întâmplă ca pe parcursul iterațiilor direcțiile de căutare să se repete.

Ar fi de dorit să existe exact n direcții de căutare, $\mathbf{d}^{(0)}, \mathbf{d}^{(1)}, \dots, \mathbf{d}^{(n-1)}$ astfel încât soluția să fie găsită după exact n pași.

Metoda direcțiilor conjugate

La fiecare pas nou

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

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

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

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

Metoda direcțiilor conjugate

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

Scădem soluția exactă din ambii termeni \Rightarrow

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

Din

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

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

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

Nu este utilă deoarece eroarea nu este cunoscută.

Metoda direcțiilor conjugate

Soluția constă în a face **direcțiile de căutare A-ortogonale** și nu ortogonale:

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

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

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

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

Metoda direcțiilor conjugate

Rezultă

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

de unde

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

expresie ce se poate calcula.

Metoda direcțiilor conjugate

Teoremă

Metoda direcțiilor conjugate calculează soluția în exact n pași.

Pentru demonstrație consultați [Shewchuk94].

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

- Se pornește de la o mulțime de n vectori liniar independenți $\mathbf{u}^{(0)}, \mathbf{u}^{(1)}, \dots, \mathbf{u}^{(n-1)}$

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

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

Metoda direcțiilor conjugate

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

unde β_{10} se alege astfel încât

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

Această relație este echivalentă cu

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

de unde

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

Metoda direcțiilor conjugate

- Similar, următoarea direcție este

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

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

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

de unde rezultă

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

Metoda direcțiilor conjugate

- În general

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

unde β_{kj} , definiți pentru $k > j$, sunt deduși din condițiile de A-ortogonalitate impuse direcțiilor, rezultând

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

Dificultate: toate direcțiile de căutare trebuie memorate pentru a construi o direcție de căutare nouă.

Metoda gradientilor conjugați

Metoda gradientilor conjugați este metoda direcțiilor conjugate în care direcțiile de căutare sunt construite prin conjugarea reziduurilor:

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

Metoda gradientilor conjugați

Justificarea acestei alegeri:

- 1 Intuitiv, inspirat de metoda celei mai rapide coborâri, în care direcțiile de căutare erau direcțiile reziduurilor.
- 2 Proprietatea reziduuului de a fi ortogonal pe direcțiile de căutare anterioare și pe reziduurile anterioare.

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

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

Metoda gradientilor conjugați

De asemenea, reziduul \mathbf{r}^k este o combinație liniară dintre reziduul anterior și produsul $\mathbf{A}\mathbf{d}^{(k-1)}$:

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

Metoda gradientilor conjugați

Coeficienții 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 înmulți relația (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

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

Metoda gradientilor conjugați

Pentru $i \neq k$, membrul drept al acestei relații 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 $i = k + 1$:

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

Metoda gradientilor conjugați

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

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

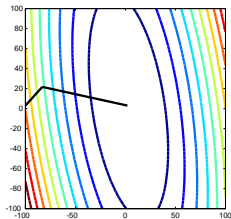
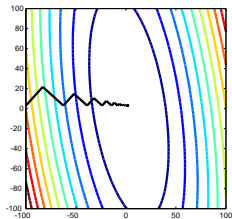
Metoda gradientilor conjugați

În concluzie, metoda gradientilor conjugați se bazează pe următoarele șase relații:

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

Metoda gradientilor conjugați

Algoritmul este complet după n iterații.



Convergența metodei gradientului depinde de inițializare și de numărul de condiționare spectrală (stânga). Metoda gradientilor conjugați converge în exact n pași, indiferent de inițializare și de numărul de condiționare spectrală (dreapta).

Metoda gradientilor conjugați

```

procedură metoda_gradientilor_conjugați(n, A, b, x0, er, maxit, x)
...
xv = x0 ; inițializează șirul iterațiilor
k = 0 ; inițializare contor iterații
r = b - A · xv ; calculează reziduu
cât timp  $\|r\| > er$  și  $k \leq maxit$ 
    dacă  $k = 0$ 
        d = r
    altfel
         $\beta = r^T r / (r^T \cdot r)$ 
        d = r +  $\beta$ dv
    •
     $\alpha = r^T r / (d^T Ad)$ 
    x = xv +  $\alpha$ d
    rv = r
    r = rv -  $\alpha$ Ad
    xv = x
    dv = d
    k = k + 1
    •
retur
    
```

Algoritmul se poate îmbunătăți, calculând la o iterație o singură dată produsul matrice - vector **Ad** și produsul scalar

$r^T r$.

Metoda gradientilor conjugați

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

Precondiționare

Precondiționare = transformarea problemei inițiale într-una echivalentă care are proprietăți îmbunătățite considerabil.

Precondiționare la stânga

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

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

\mathbf{M} matrice nesingulară, numită *matrice de precondiționare* sau *precondiționator*.

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

Precondiționare

Precondiționare la dreapta

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

unde $\mathbf{x} = \mathbf{M}^{-1}\mathbf{y}$.

Se pot folosi ambele tipuri de precondiționare simultan.

Precondiționare matricelor simetrice și pozitiv definite

Precondiționarea se face astfel încât să se păstreze această proprietate.

Fie $\mathbf{M} = \mathbf{C}\mathbf{C}^T$ - simetrică și pozitiv definită.

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

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

Precondiționare

Metode de precondiționare celebre

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

Precondiționare

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

Referințe

- [Ciuprina13a] Gabriela Ciuprina - Algoritmi numerici pentru calcule științifice 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.

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**

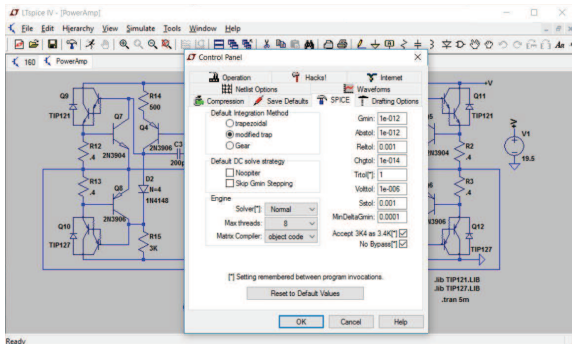
Referințe

← → C 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	
BDDCMIL	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	X			2014-08-21
HIPS	CeCILL-C	yes	X	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
Lia	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
PDM	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

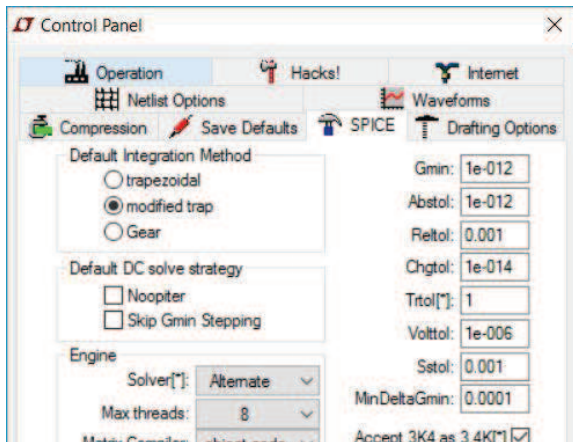
Pe scurt

Fiți atenți la astfel de informații (capturi din LTSPICE).
Consultați documentația pentru detalii!



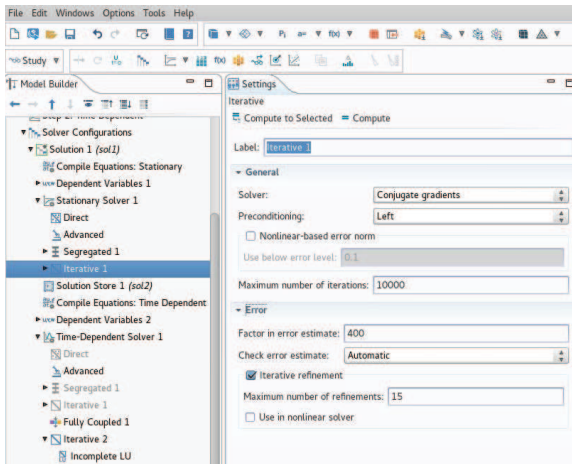
Pe scurt

Fiți atenți la astfel de informații (capturi din LTSPICE).
Consultați documentația pentru detalii!



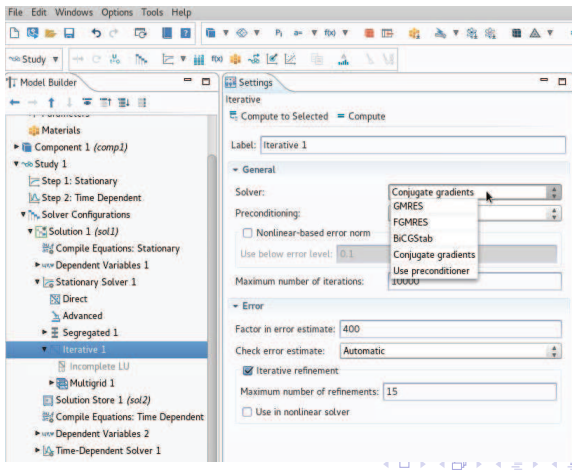
Pe scurt

Fiți atenți la astfel de informații (capturi din COMSOL)



Pe scurt

Fiți atenți la astfel de informații (capturi din COMSOL)



Pe scurt

