Metode iterative pentru rezolvarea sistemelor de ecuații liniare: Jacobi, Gauss-Seidel, Suprarelaxare

Cuprins

1 Obiective laborator

În urma parcurgerii acestui laborator, studentul va fi capabil să rezolve sisteme de ecuații liniare utilizând metode iterative.

2 Noțiuni teoretice

Metodele exacte de rezolvare a sistemelor de ecuații liniare, având complexitate $O(n^3)$, au aplicabilitate limitată la sisteme de ordin relativ mic. Pentru sisteme de dimensiuni mai mari se utilizează metode iterative, cu complexitate $O(n^2)$. Acestea utilizează relații de recurență, care prin aplicare repetată furnizează aproximații, cu precizie controlată, a soluției sistemului.

2.1 Metode iterative pentru funcții

Vom dezvolta ideea de *metode iterative* pornind de la un caz mai simplu, cel al metodelor iterative pentru funcții.

Exemplu grafic. Numărul de aur. Începem cu orice număr real pozitiv x și calculăm

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

La un moment dat, $|x^{(k+1)} - x^{(k)}| < \varepsilon$, unde ε este o valoare mică dată. Asta înseamnă $x^{(k+1)} \approx x^{(k)}$.

$$\begin{aligned}
x &= \sqrt{1+x} \\
x^2 &= 1 + x \\
x^2 - x - 1 &= 0 \\
x &= \frac{1 \pm \sqrt{5}}{2} \\
\phi &= \frac{1 + \sqrt{5}}{2} \approx 1.618033988749895
\end{aligned}$$

Soulția sistemului este dată de intersecția a două funcții: $f(x) = x$ și $g(x) = \sqrt{1+x}$. Dacă trasăm grafic aceste puncte, obținem un punct de intersecție, care este soluția sistemului.

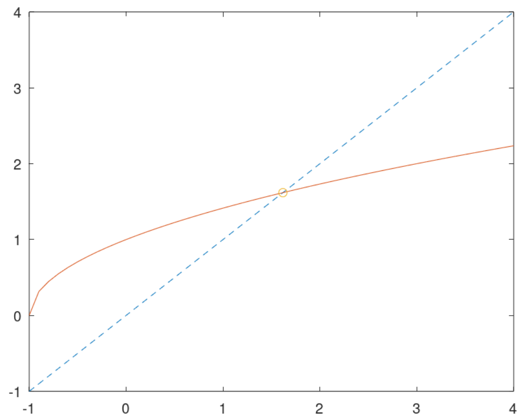

Figura 1: Numărul de aur

2.1.1 Puncte fixe

Definiție. Fie $f : M \to M$ o funcție continuă și un spațiu metric (M, d). f este o contracție dacă există un număr real $k \in [0, 1)$ astfel încât

$$d(f(x), f(y)) \leq kd(x, y), \quad \forall x, y \in M$$

Cel mai mic număr k pentru care această relație este adevărată se numește *constantă Lipschitz*. Pentru că $k < 1$, putem spune despre f că este continuă.

Teorema lui Banach. Dacă $f : M \to M$ este o contracție pe un spațiu metric complet (M, d), atunci există un unic punct fix $x^* \in M$ pentru care $f(x^*) = x^*$.

Pe $\mathbb{R}$, distanța o putem considera ca fiind $d(x, y) = |x - y|$.

$$|f(x) - f(y)| \leq k|x - y|$$
$$\frac{|f(x) - f(y)|}{|x - y|} \leq k$$
$$|\frac{f(x) - f(y)}{x - y}| \leq k$$

dar, din teorema lui Lagrange, știm că există un $c \in (a, b)$ astfel încât

$$|f'(c)| \leq k$$

Astfel, putem spune că iterația $x^{(k+1)} = f(x^{(k)})$ converge către un punct fix x^* dacă $|f'(c)| < 1$, pentru orice $c \in (a, b)$.

Exemplu. Fie $f(x) = \sqrt{1+x}$. Calculăm derivata funcției:

$$f'(x) = \frac{1}{2}(1+x)^{-\frac{1}{2}}$$
$$f'(x) = \frac{1}{2\sqrt{1+x}}$$

Pentru orice $x \in \mathbb{R}, \quad x \geq 0$, avem $|f'(x)| < 1$, deci funcția este o contracție pe $\mathbb{R}_+$ și are un punct fix unic.

2.2 Metode iterative pentru sisteme de ecuații liniare

Folosind teorema de mai sus, putem spune despre transformarea liniară $T : \mathbb{R}^n \to \mathbb{R}^n$ că este o contracție dacă "derivata" transformării este mai mică decât 1. Pentru că T este o funcție vectorială, derivata se calculează folosind *matricea Jacobiană*. Ne dorim ca această matrice să aibe norma subunitară, lucru care se întâmplă dacă matricea are raza spectrală mai mică decât 1.

Considerăm sistemul $Ax = b$. Ne dorim să îl rescriem pentru a ajunge la forma $x = T(x)$. Metoda prin care vom face asta e să descompunem matricea A în $A = M - N$.

$$\begin{aligned} (M - N)x &= b \\ Mx &= Nx + b \\ x &= M^{-1}Nx + M^{-1}b \end{aligned}$$

Astfel, $T(x) = M^{-1}Nx + M^{-1}b$. Notăm $G = M^{-1}N$ și $c = M^{-1}b$. G este *matricea de iterație*, iar c este *vectorul de iterație*. Jacobiana lui T este chiar matricea G, ceea ce înseamnă că, pentru convergență, avem nevoie ca raza spectrală a lui G să fie mai mică decât 1. Acest lucru se poate vedea și din analiza erorii:

$$\begin{aligned} x - x^{(k)} &= (Gx + c) - (Gx^{(k-1)} + c) = G(x - x^{(k-1)}) \\ e^{(k)} &= G^k e^{(0)} \end{aligned}$$

Pentru convergență, avem nevoie ca $\lim_{k\to\infty} e^{(k)} = 0 \equiv \lim_{k\to\infty} G^k = 0$. Pentru o demonstrație mai amplă, se poate consulta [1].

Pentru alegerea lui M și N, vom partiționa matricea A punând în evidență o matrice diagonală D, o matrice strict triunghiular inferioară L și o matrice strict triunghiular superioară U:

$$A = D - L - U$$

$$\underset{A}{\boxed{\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}}} = \underset{D}{\boxed{\begin{matrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{matrix}}} - \underset{L}{\boxed{\begin{matrix} & & \\ -a_{21} & & \\ -a_{31} & -a_{32} & \end{matrix}}} - \underset{U}{\boxed{\begin{matrix} & -a_{12} & -a_{13} \\ & & -a_{23} \\ & & \end{matrix}}}$$

Diferența dintre următoarele metode constă în modul în care se asociază aceste matrice.

Condiția de oprire. Pentru a finaliza execuția metodelor iterative din cadrul acestui laborator vom introduce doua condiții: **toleranța** (notată în continuare cu ϵ) și **numărul maxim de iterații** (notat în continuare cu N_{iter}). Toleranța ne asigură faptul ca algoritmul nu efectuează iterații mai mult decât este necesar, practic nu se continuă execuția dacă "diferența" soluțiilor a două iterații consecutive nu este semnificativă. Numărul maxim de pași garantează că algoritmul se încheie, indiferent dacă alegerea inițială pentru $x^{(0)}$ a fost una bună sau nu.

2.2.1 Metoda Jacobi

În metoda Jacobi se aleg:

$$\begin{gathered} M = D \\ N = L + U \\ G = D^{-1}(L+U) \\ c = D^{-1}b \end{gathered}$$

Pentru că M este o matrice diagonală, inversa sa este foarte ușor de calculat. Pasul de itarație este:

$$\begin{aligned} x^{(k+1)} &= D^{-1}[(L+U)x^{(k)} + b] \\ x_i^{(p+1)} &= \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(p)}}{a_{ii}} \end{aligned}$$

Algoritmul în MATLAB poate fi gândit în două moduri. Cel mai simplu este să folosim prima relație de mai sus.

Algorithm 1 Metoda Jacobi

$D \leftarrow \mathrm{diag}(\mathrm{diag}(A))$ ▷ Extragem Matricea diagonală
$L \leftarrow -\mathrm{tril}(A, -1)$ ▷ Extragem matricea inferior triunghiulară
$U \leftarrow -\mathrm{triu}(A, 1)$ ▷ Extragem matricea superior triunghiulară
$G \leftarrow D^{-1}(L+U)$ ▷ Matricea de iterație
$c \leftarrow D^{-1}b$ ▷ Vectorul de iterație
$x \leftarrow \mathrm{zeros}(\mathrm{length}(b), 1)$ ▷ Inițializăm vectorul soluție
$i \leftarrow 1$
while $i \leq \mathrm{max\_iter}$ **do**
    $x_{\mathrm{prev}} \leftarrow x$
    $x \leftarrow G \cdot x + c$
    if $\|x - x_{\mathrm{prev}}\| < \mathrm{tol}$ **then**
        break
    end if
    $i \leftarrow i + 1$
end while

2.2.2 Metoda Gauss-Seidel

În metoda Gauss-Seidel se aleg:

$$\begin{gathered} M = D - L \\ N = U \\ G = (D-L)^{-1}U \\ c = (D-L)^{-1}b \end{gathered}$$

În acest caz, matricea M este o matrice inferior triunghiulară, iar inversa nu mai este așa ușor de calculat.

Pasul de iterație este:

$$x^{(k+1)} = (D-L)^{-1}(Ux^{(k)} + b)$$
$$x_i^{(p+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(p+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(p)}}{a_{ii}}$$
$$x_i = \frac{b_i - \sum_{j\neq i} a_{ij}x_j}{a_{ii}}$$

Diferența dintre metoda Jacobi și Gauss-Seidel constă în faptul că în metoda Gauss-Seidel, la calculul lui $x_i^{(p+1)}$ se folosesc valorile deja calculate pentru $x_j^{(p+1)}$ cu $j < i$.

Algorithm 2 Metoda Gauss-Seidel

$x \leftarrow \text{zeros}(\text{length}(b), 1)$ ▷ Inițializăm vectorul soluție
for $i = 1$ to max_iter **do**
  $x_{\text{prev}} \leftarrow x$
  for $j = 1$ to length(x) **do**
    $x[j] \leftarrow \frac{b[j] - \sum_{k\neq j} A[j,k]x[k]}{A[j,j]}$
  end for
  if $\|x - x_{\text{prev}}\| < \text{tol}$ **then**
    break
  end if
end for

De observat la metodele Jacobi și Gauss-Seidel este că o condiție suficientă dar nu necesară pentru convergența acestora este ca matricea A să fie diagonal dominantă. Metodele Jacobi și Gauss-Seidel au proprietatea că ori sunt ambele convergente, ori niciuna nu este convergentă (teorema Stein-Rosenberg). Atunci când converg, Gauss-Seidel converge mai rapid decât Jacobi, $\rho(GS) < \rho(J) < 1$.

2.2.3 Metoda suprarelaxării

O variantă a metodei Gauss-Seidel este metoda suprarelaxării succesive (SOR). Se introduce un parametru de relaxare ω și se obține:

$$A = M - N$$
$$A = M - \omega M - N + \omega M$$
$$A = (1-\omega)M - (N - \omega M)$$
$$A = M(\omega) - N(\omega)$$

Dacă notăm cu GS formula pentru $x_i^{(p+1)}$ de la Gauss-Seidel, atunci formula pentru SOR este:

$$x_i^{(p+1)} = (1-\omega)x_i^{(p)} + \omega GS$$

Dacă A este simetrică și pozitiv definită, atunci pentru $\omega \in (0,2)$ metoda SOR converge. Pentru $\omega = 1$ obținem metoda Gauss-Seidel.

Algorithm 3 Metoda SOR

$x \leftarrow \text{zeros}(\text{length}(b), 1)$ ▷ Inițializăm vectorul soluție
for $i = 1$ to max_iter **do**
    $x_{\text{prev}} \leftarrow x$
    for $j = 1$ to length(x) **do**
        $x[j] \leftarrow \frac{b[j] - \sum_{k \neq j} A[j,k]x[k]}{A[j,j]}$
    end for
    $x \leftarrow \omega \cdot x + (1 - \omega) \cdot x_{\text{prev}}$ ▷ Aplicăm relaxarea
    if $\|x - x_{\text{prev}}\| < \text{tol}$ **then**
        break
    end if
end for

3 Probleme

1. Să se implementeze în MATLAB funcțiile pentru metodele iterative Jacobi, Gauss-Seidel și SOR.
2. Desenați grafic, pentru fiecare metodă, evoluția erorii în funcție de numărul de iterații. Testați parametrii diferiți pentru ω în metoda SOR.
3. Folosiți metoda Jacobi pentru a aproxima soluția sistemului:

$$\begin{aligned} 10x_1 - 5x_2 + x_3 &= 1 \\ x_1 + 4x_2 + 3x_3 &= 4 \\ 4x_1 - 3x_2 - 9x_3 &= 6 \end{aligned}$$

4. Fie sistemul liniar:

$$\begin{aligned} 2x + y + z &= 4 \\ x + 2y + z &= 4 \\ x + y + 2z &= 4 \end{aligned}$$

Stabiliți convergența metodelor Jacobi și Gauss-Seidel și razele spectrale corespunzătoare. În caz de convergență, calculați soluția iterativă după trei pași. Alegeți voi aproximația inițială.

Referințe

[1] Anders C. Hansen. Iterative methods for linear algebraic systems.