

(cursul 4 - S2)

3.4. Metode iterative

- eliminarea gaussiană este o secvență finită de operații care au ca rezultat o soluție
- din acest motiv, eliminarea gaussiană este numită o metodă **directă** de rezolvare a sistemelor de ecuații liniare
- metodele directe, teoretic, dă soluția exactă într-un număr finit de pași
- metodele directe sunt în contrast cu metodele de găsire a soluțiilor descrise în Capitolul 2, care au o formă iterativă
- așa-numitele metode **iterative** pot fi de asemenea aplicate pentru rezolvarea sistemelor de ecuații liniare
- asemenea iterării de punct fix, metodele încep de la o valoare inițială pe care o rafinează la fiecare pas, convergând către vectorul soluție

3.4.1. Metoda lui Jacobi

→ este o formă a iterării de punct fix pt. un sistem de ecuații

- în IPF, primul pas este să rescriem ecuațiile, pentru a găsi necunoscuta
- primul pas în metoda lui Jacobi este să facem acest lucru în următoarea formă standardizată: rezolvăm o ecuație pentru a găsi a-i-a necunoscută
- apoi, se iterează ca în iterăția de punct fix, începând de la valoarea inițială

Exemplul 1

Aplicații Metoda lui Jacobi pentru $\begin{cases} 3u + v = 5 \\ u + 2v = 5 \end{cases}$

$$(u_0, v_0) = (0, 0)$$

$$u = \frac{5 - v}{3}$$

$$v = \frac{5 - u}{2}$$

Cele 2 ecuații iterate sunt

$$\begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} \frac{5 - v_0}{3} \\ \frac{5 - u_0}{2} \end{bmatrix} = \begin{bmatrix} \frac{5 - 0}{3} \\ \frac{5 - 0}{2} \end{bmatrix} = \begin{bmatrix} \frac{5}{3} \\ \frac{5}{2} \end{bmatrix}$$

$$\begin{bmatrix} u_2 \\ v_2 \end{bmatrix} = \begin{bmatrix} \frac{5-v_1}{3} \\ \frac{5-u_1}{2} \end{bmatrix} = \begin{bmatrix} \frac{5-5/2}{3} \\ \frac{5-5/3}{2} \end{bmatrix} = \begin{bmatrix} \frac{5}{6} \\ \frac{5}{3} \end{bmatrix}$$

$$\begin{bmatrix} u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} \frac{5-v_2}{3} \\ \frac{5-u_2}{2} \end{bmatrix} = \begin{bmatrix} \frac{5-5/3}{3} \\ \frac{5-5/6}{2} \end{bmatrix} = \begin{bmatrix} \frac{10}{9} \\ \frac{25}{12} \end{bmatrix}.$$

\Rightarrow Convergență către soluția $[1, 2]^T$

Exemplul 2

Aplicați Metoda lui Jacobi pentru $\begin{cases} u+2v=5 \\ 3u+v=5 \end{cases}$

$$\begin{cases} u = 5 - 2v \\ v = 5 - 3u \end{cases}$$

- cele două ecuații sunt iterate ca mai înainte, dar rezultatele sunt destul de diferite:

$$\begin{aligned} \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} &= \begin{bmatrix} 5 - 2v_0 \\ 5 - 3u_0 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \\ \begin{bmatrix} u_2 \\ v_2 \end{bmatrix} &= \begin{bmatrix} 5 - 2v_1 \\ 5 - 3u_1 \end{bmatrix} = \begin{bmatrix} -5 \\ -10 \end{bmatrix} \\ \begin{bmatrix} u_3 \\ v_3 \end{bmatrix} &= \begin{bmatrix} 5 - 2v_2 \\ 5 - 3u_2 \end{bmatrix} = \begin{bmatrix} 25 \\ 20 \end{bmatrix}. \end{aligned} \quad (4)$$

- În acest caz, metoda lui Jacobi eșuează, deoarece iterarea diverge

! Metoda lui Jacobi nu reușește întotdeauna
Avem multe condiții în care aceasta funcționează

Definiția 1

Matricea $n \times n A = (a_{ij})$ este **strict diagonal dominantă** dacă, pentru orice $1 \leq i \leq n$, $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$. Cu alte cuvinte, fiecare intrare de pe diagonala principală domină rândul pe care se află în sensul în care este mai mare în valoare absolută decât suma valorilor absolute ale celorlalte intrări de pe rândul respectiv.

Teorema 1

Dacă matricea $n \times n A$ este strict diagonal dominantă, atunci (1) A este o matrice nesingulară, și (2) pentru orice vector b și orice valoare inițială, metoda lui Jacobi aplicată ecuației $Ax = b$ converge către soluția unică.

- Teorema 1 spune că, dacă A este strict diagonal dominantă, atunci metoda lui Jacobi aplicată ecuației $Ax = b$ converge către o soluție pentru orice valoare inițială

- În Exemplul 1, matricea coeficientelor este la început

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix},$$

care este strict diagonal dominantă, deoarece $3 > 1$ și $2 > 1$

- convergența este garantată în acest caz
- pe de altă parte, în Exemplul 2, metoda lui Jacobi este aplicată matricii

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix},$$

care nu este diagonal dominantă, și nu există o astfel de garanție

- observăm că strict diagonal dominantă este doar o condiție suficientă
- metoda lui Jacobi s-ar putea totuși să conveargă chiar și în absența acestei condiții

Exemplul 3

- determinați dacă matricile

$$A = \begin{bmatrix} 3 & 1 & -1 \\ 2 & -5 & 2 \\ 1 & 6 & 8 \end{bmatrix} \text{ și } B = \begin{bmatrix} 3 & 2 & 6 \\ 1 & 8 & 1 \\ 9 & 2 & -2 \end{bmatrix}$$

sunt strict diagonal dominante

pt. A : $\left\{ \begin{array}{l} |3| > |1| + |-1| \\ |-5| > |2| + |2| \\ |8| > |1| + |6| \end{array} \right.$ $\Rightarrow A$ este diagonal dominantă

pt. B : $\left\{ \begin{array}{l} |3| < |2| + |6| \\ |8| > |1| + |1| \\ |-2| < |9| + |2| \end{array} \right.$ $\Rightarrow B$ nu este diagonal dominantă

Dacă pt. B $L_1 \leftrightarrow L_3$ $\Rightarrow B$ este diagonal dominantă
 \Rightarrow va converge M. Jacobi

- metoda lui Jacobi este o formă a iterării de punct fix
- fie D diagonala principală a lui A , L triunghiul inferior al lui A (intrările de sub diagonala principală), și U triunghiul superior al lui A (intrările de deasupra diagonalei principale)
- atunci $A = L + D + U$, și ecuația care trebuie rezolvată este $Lx + Dx + Ux = b$
- observăm că această utilizare a lui L și U diferă de cea din factorizarea LU, deoarece toate intrările diagonale pentru acești L și U sunt zero
- sistemul de ecuații $Ax = b$ poate fi reformat sub forma unei iterări de punct fix:

$$\begin{aligned} Ax &= b \\ (D + L + U)x &= b \\ Dx &= b - (L + U)x \\ x &= D^{-1}(b - (L + U)x). \end{aligned} \quad (5)$$

- deoarece D este o matrice diagonală, inversa ei este matricea diagonală a inverselor intrărilor matricii A
- metoda lui Jacobi este doar iterăția de punct fix din (5):

Algoritmul 1 (Metoda lui Jacobi)

$$\begin{aligned}x_0 &= \text{vectorul inițial} \\x_{k+1} &= D^{-1}(b - (L + U)x_k) \text{ for } k = 0, 1, 2, \dots\end{aligned}\quad (6)$$

- pentru Exemplul 1,

$$\begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix},$$

iterația de punct fix (6) cu $x_k = \begin{bmatrix} u_k \\ v_k \end{bmatrix}$ este

$$\begin{aligned}\begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} &= D^{-1}(b - (L + U)x_k) \\&= \begin{bmatrix} 1/3 & 0 \\ 0 & 1/2 \end{bmatrix} \left(\begin{bmatrix} 5 \\ 5 \end{bmatrix} - \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_k \\ v_k \end{bmatrix} \right) \\&= \begin{bmatrix} \frac{5-v_k}{3} \\ \frac{5-u_k}{2} \end{bmatrix},\end{aligned}$$

care este aceeași cu versiunea originală

```
jacobi.m
1 function x=jacobi(A,b,x0,k)
2
3 D=diag(diag(A));
4 L=tril(A)-D;
5 U=triu(A)-D;
6 x=x0;
7
8 for j=1:k
9     x = inv(D)*(b-(L+U)*x);
10    end
```

Command Window

```
>> clear
>> A=[3 1 -1; 2 4 1; -1 2 5];
>> b=[-4; 4; 13];
>> x0=zeros(3,1);
>> x=jacobi(A,b,x0,10)

x =
-0.9981
0.9980
2.0018
```

3.4.2. Metoda Gauss-Seidel și SRS

→ diferență față de Jacobi este că în G-S, cele mai recent actualizate valori ale necunoscutelor sunt utilizate la fiecare pas, chiar dacă actualizarea a avut loc în pașul curent

- întorcându-ne la Exemplul 1, vedem că metoda Gauss-Seidel arată după cum urmează:

$$\begin{aligned}
 \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
 \begin{bmatrix} u_1 \\ v_1 \end{bmatrix} &= \begin{bmatrix} \frac{5-v_0}{3} \\ \frac{5-u_1}{2} \end{bmatrix} = \begin{bmatrix} \frac{5-0}{3} \\ \frac{5-5/3}{2} \end{bmatrix} = \begin{bmatrix} \frac{5}{3} \\ \frac{5}{3} \end{bmatrix} \\
 \begin{bmatrix} u_2 \\ v_2 \end{bmatrix} &= \begin{bmatrix} \frac{5-v_1}{3} \\ \frac{5-u_2}{2} \end{bmatrix} = \begin{bmatrix} \frac{5-5/3}{3} \\ \frac{5-10/9}{2} \end{bmatrix} = \begin{bmatrix} \frac{10}{9} \\ \frac{35}{18} \end{bmatrix} \\
 \begin{bmatrix} u_3 \\ v_3 \end{bmatrix} &= \begin{bmatrix} \frac{5-v_2}{3} \\ \frac{5-u_3}{2} \end{bmatrix} = \begin{bmatrix} \frac{5-35/18}{3} \\ \frac{5-55/54}{2} \end{bmatrix} = \begin{bmatrix} \frac{55}{54} \\ \frac{215}{108} \end{bmatrix}. \tag{7}
 \end{aligned}$$

- observăm diferența dintre metodele Gauss-Seidel și Jacobi: definiția lui v_1 folosește u_1 , nu u_0
- vedem convergența către soluția $[1, 2]^T$, ca în metoda lui Jacobi, dar cu ceva mai mare precizie în același număr de pași
- metoda Gauss-Seidel adesea converge mai repede decât metoda lui Jacobi, dacă este convergentă
- metoda Gauss-Seidel, la fel ca metoda lui Jacobi, converge către soluție dacă matricea coeficientilor este strict diagonal dominantă
- metoda Gauss-Seidel poate fi scrisă în formă matricială și identificată ca o iterație de punct fix în care izolăm ecuația $(L + D + U)x = b$ în forma

$$(L + D)x_{k+1} = -Ux_k + b.$$

- observăm că folosirea intrărilor nou determinate ale lui x_{k+1} este realizată prin includerea triunghiului inferior al lui A în partea stângă a ecuației
- rearanjând această ecuație, obținem metoda Gauss-Seidel

Algoritm 2 (Metoda Gauss-Seidel)

$$\begin{aligned}
 x_0 &= \text{vectorul inițial} \\
 x_{k+1} &= D^{-1}(b - Ux_k - Lx_{k+1}) \text{ for } k = 0, 1, 2, \dots
 \end{aligned}$$

```

gauss_seidel.m  jacobi.m  +
1 function x = gauss_seidel(A, b, x0, k)
2
3 D = diag(diag(A));
4 L = tril(A) - D;
5 U = triu(A) - D;
6
7 x = x0;
8
9 for i = 1 : k
10     x = inv(D + L) * (b - U * x);
11 end

```

```

x =
6.7976
-6.5974
4.7978

>> x=jacobi(A,b,x0,30)

x =
6.8000
-6.5999
4.8000

>> x=jacobi(A,b,x0,40)

x =
6.8000
-6.6000
4.8000

>> x = gauss_seidel(A, b, x0, 20)

x =
6.8000
-6.6000
4.8000

```

Exemplul 4

aplicați metoda Gauss–Seidel sistemului

$$\begin{bmatrix} 3 & 1 & -1 \\ 2 & 4 & 1 \\ -1 & 2 & 5 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 1 \end{bmatrix}.$$

iterația Gauss–Seidel este

$$\begin{aligned} u_{k+1} &= \frac{4 - v_k + w_k}{3} \\ v_{k+1} &= \frac{1 - 2u_{k+1} - w_k}{4} \\ w_{k+1} &= \frac{1 + u_{k+1} - 2v_{k+1}}{5}. \end{aligned}$$

începând cu $x_0 = [u_0, v_0, w_0]^T = [0, 0, 0]^T$, calculăm

$$\begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} = \begin{bmatrix} \frac{4-0+0}{3} \\ \frac{1-2(4/3)-0}{4} \\ \frac{1+(4/3)-2(-5/12)}{5} \end{bmatrix} = \begin{bmatrix} \frac{4}{3} \\ -\frac{5}{12} \\ \frac{19}{30} \end{bmatrix} \approx \begin{bmatrix} 1.3333 \\ -0.4167 \\ 0.6333 \end{bmatrix}$$

și

$$\begin{bmatrix} u_2 \\ v_2 \\ w_2 \end{bmatrix} = \begin{bmatrix} \frac{101}{60} \\ -\frac{3}{4} \\ \frac{251}{300} \end{bmatrix} \approx \begin{bmatrix} 1.6833 \\ -0.7500 \\ 0.8367 \end{bmatrix}.$$

sistemul este strict diagonal dominant, și prin urmare iterăția va converge către soluția $[2, -1, 1]^T$

3.4.2. Metoda Gauss–Seidel și SRS

- metoda numită a **supra-relaxărilor successive (SRS)** preia direcția dată de metoda Gauss–Seidel înspre soluție și „depășește măsura” în încercarea de a accelera convergența
- fie ω un număr real, și definim fiecare componentă a noii aproximări x_{k+1} ca o medie ponderată de ω ori formula Gauss–Seidel și de $1 - \omega$ ori aproximarea curentă x_k
- numărul ω se numește **parametru de relaxare**, și când $\omega > 1$ vorbim despre **supra-relaxare**

Exemplul 5

- aplicați SRS cu $\omega = 1.25$ sistemului din Exemplul 4
- metoda supra-relaxărilor successive ne dă

$$\begin{aligned} u_{k+1} &= (1 - \omega)u_k + \omega \frac{4 - v_k + w_k}{3} \\ v_{k+1} &= (1 - \omega)v_k + \omega \frac{1 - 2u_{k+1} - w_k}{4} \\ w_{k+1} &= (1 - \omega)w_k + \omega \frac{1 + u_{k+1} - 2v_{k+1}}{5}. \end{aligned}$$

începând cu $[u_0, v_0, w_0]^T = [0, 0, 0]^T$, calculăm

$$\begin{bmatrix} u_1 \\ v_1 \\ w_1 \end{bmatrix} \approx \begin{bmatrix} 1.6667 \\ -0.7292 \\ 1.0312 \end{bmatrix}$$

și

$$\begin{bmatrix} u_2 \\ v_2 \\ w_2 \end{bmatrix} \approx \begin{bmatrix} 1.9835 \\ -1.0672 \\ 1.0216 \end{bmatrix}.$$

în acest exemplu, iterăția SRS converge mai repede decât Jacobi și Gauss–Seidel către soluția $[2, -1, 1]^T$

- problema $Ax = b$ poate fi scrisă $(L + D + U)x = b$, și, când înmulțim cu ω și rearanjăm, avem

$$\begin{aligned} (\omega L + \omega D + \omega U)x &= \omega b \\ (\omega L + D)x &= \omega b - \omega Ux + (1 - \omega)Dx \\ x &= (\omega L + D)^{-1}[(1 - \omega)Dx - \omega Ux] + \omega(D + \omega L)^{-1}b. \end{aligned}$$

Algoritm 3 (Metoda supra-relaxărilor successive (SRS))

$$\begin{aligned} x_0 &= \text{vectorul initial} \\ x_{k+1} &= (\omega L + D)^{-1}[(1 - \omega)Dx_k - \omega Ux_k] + \omega(D + \omega L)^{-1}b \text{ for } k = 0, 1, 2, \dots \end{aligned}$$

- SRS cu $\omega = 1$ este exact Gauss-Seidel
- parametrul ω poate de asemenea să fie mai mic decât 1, dând astfel naștere unei metode numită metoda sub-relaxărilor succesive

The screenshot shows the MATLAB environment. In the top window, the code for 'sor.m' is displayed:

```

1 function x = sor(A, b, x0, omega, iter)
2
3 x = x0;
4
5 D = diag(diag(A));
6 L = tril(A) - D;
7 U = triu(A) - D;
8
9 for k = 1 : iter
10
11     x = inv(omega * L + D) * ((1 - omega) * D * x - omega * U * x) + omega * inv(D + omega * L) * b;
12
13 end
14
15 %Exemplul 6 curs 4
16
17 %A = [3 1 -1; 2 4 1; -1 2 5];
18 %b = [4; 1; 1];
19 %omega = 1.2;
20 %x0 = zeros(3,1);
21 %x = sor(A, b, x0, omega, 20)
22
23 %problema 5 lab 3
24
25
26 %problema 9 lab 3

```

In the bottom window, the Command Window shows the execution of the script with specific parameters:

```

>> A = [3 1 -1; 2 4 1; -1 2 5];
b = [4; 1; 1];
omega = 1.2;
x0 = zeros(3,1);
x = sor(A, b, x0, omega, 20)

x =
    2.0000
   -1.0000
    1.0000

```

3.5. Metode pentru matrice simetrice și pozitiv definite

- matricile simetrice au o poziție privilegiată în analiza sistemelor liniare datorită structurii lor speciale și pentru că au doar aproximativ jumătate din numărul de intrări independente al unor matrice generale
- aceasta ridică întrebarea dacă nu cumva factorizări ca factorizarea LU pot fi realizate la jumătate din complexitatea computațională, și folosind doar jumătate din locațiile de memorie
- pentru matricile simetrice și pozitiv definite, acest scop poate fi atins de către factorizarea Cholesky
- matricile simetrice și pozitiv definite permit de asemenea o abordare destul de diferită pentru rezolvarea ecuației $Ax = b$, care nu depinde de o factorizare matricială
- această nouă abordare, numită metoda gradientilor conjugăți, este utilă mai ales pentru matrici mari
- în debutul secțiunii, definim conceptul de pozitiv definitire pentru o matrice simetrică
- apoi, arătăm că orice matrice simetrică și pozitiv definită A poate fi factorizată sub forma $A = R^T R$ pentru o matrice superior triangulară R , fapt ce reprezintă factorizarea Cholesky
- încheiem secțiunea cu algoritmul gradientilor conjugăți

Definiția 2

Matricea $n \times n A$ este **simetrică** dacă $A^T = A$. Matricea A este **pozitiv definită** dacă $x^T A x > 0$ pentru orice vector $x \neq 0$.

Exemplul 6

arătați că matricea $A = \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix}$ este simetrică și pozitiv definită evident, A este simetrică

\rightarrow pt. a arăta că este pozitiv definită

$$x^T A x = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 2x_1^2 + 4x_1x_2 + 5x_2^2 = \\ = 2(x_1 + 2x_2)^2 + 3x_2^2 \geq 0$$

această expresie este întotdeauna nenegativă, și nu poate fi zero decât dacă $x_2 = 0$ și $x_1 + x_2 = 0$, care împreună implică $x = 0$

Exemplul 7

- arătați că matricea simetrică $A = \begin{bmatrix} 2 & 4 \\ 4 & 5 \end{bmatrix}$ nu este pozitiv definită
- calculăm $x^T A x$ prin completarea pătratului:

$$x^T A x = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} 2 & 4 \\ 4 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \\ = 2x_1^2 + 8x_1x_2 + 5x_2^2 = \\ = \underbrace{2(x_1 + 2x_2)^2}_{\geq 0} - \underbrace{3x_2^2}_{\geq 0} \geq 0$$

dor pt $x_1 = -2$ și $x_2 = 1 \Rightarrow x^T A x < 0$

Obs: O matrice simetrică și pozitiv definită trebuie să fie nesingulară ! Este imposibil ca un vector nenul x să satisfacă $A \cdot x = 0$.

Teorema 2

Presupunem că A este o matrice simetrică $n \times n$ cu intrări reale. Atunci valorile proprii ale lui A sunt numere reale, și mulțimea vectorilor proprii unitari ai lui A este o mulțime ortonormală $\{w_1, \dots, w_n\}$ care formează o bază a lui \mathbb{R}^n .

Faptul 1

Dacă matricea $n \times n A$ este simetrică, atunci A este pozitiv definită dacă și numai dacă toate valorile proprii ale sale sunt pozitive.

Teorema 2 ne spune că mulțimea vectorilor proprii este ortonormală și formează o bază a lui \mathbb{R}^n

dacă A este pozitiv definită și $Av = \lambda v$ pentru un vector nenul v , atunci $0 < v^T A v = v^T(\lambda v) = \lambda \|v\|_2^2$, deci $\lambda > 0$

- pe de altă parte, dacă toate valorile proprii ale lui A sunt pozitive, atunci putem scrie orice vector nenul $x = c_1 v_1 + \dots + c_n v_n$, unde v_i sunt vectori unitari ortonormali și nu toți c_i sunt zero
- atunci $x^T A x = (c_1 v_1 + \dots + c_n v_n)^T (\lambda_1 c_1 v_1 + \dots + \lambda_n c_n v_n) = \lambda_1 c_1^2 + \dots + \lambda_n c_n^2 > 0$, deci A este pozitiv definită
- valorile proprii ale lui A din Exemplul 6 sunt 6 și 1
- valorile proprii ale lui A din Exemplul 7 sunt aproximativ 7.77 și -0.77

Faptul 2

Dacă A este o matrice $n \times n$ simetrică și pozitiv definită și X este o matrice $n \times m$ de rang maxim, cu $n \geq m$, atunci $X^T A X$ este o matrice $m \times m$ simetrică și pozitiv definită.

- matricea este simetrică deoarece $(X^T A X)^T = X^T A X$
- considerăm un m -vector nenul v
- observăm că $v^T (X^T A X) v = (Xv)^T A (Xv) \geq 0$, cu egalitate doar dacă $Xv = 0$, datorită faptului că A este pozitiv definită
- deoarece X are rang maxim, coloanele ei sunt liniar independente, astfel că $Xv = 0$ implică $v = 0$

Definiția 3

O submatrice **principală** a matricii pătratice A este o submatrice pătratăcă ale cărei intrări diagonale sunt intrări diagonale ale lui A .

Faptul 3

Orice submatrice principală a unei matrici simetrice și pozitiv definite este de asemenea simetrică și pozitiv definită.

- de exemplu, dacă

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

este simetrică și pozitiv definită, atunci la fel este și

$$\begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}.$$

3.5.2. Factorizarea Cholesky

- pentru a demonstra ideea principală, începem cu cazul 2×2
- toate problemele importante apar în acest caz, extensia la cazul general fiind imediată
- considerăm matricea simetrică și pozitiv definită

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}.$$

- din Faptul 3 despre matricile simetrice și pozitiv definite, știm că $a > 0$
- în plus, știm că determinantul $ac - b^2$ al lui A este pozitiv, deoarece determinantul este produsul valorilor proprii, care sunt toate pozitive, conform Faptului 1
- scriind $A = R^T R$ unde R este o matrice superior triangulară, implică forma

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} \sqrt{a} & 0 \\ u & v \end{bmatrix} \begin{bmatrix} \sqrt{a} & u \\ 0 & v \end{bmatrix} = \begin{bmatrix} a & u\sqrt{a} \\ u\sqrt{a} & u^2 + v^2 \end{bmatrix},$$

și vrem să verificăm dacă acest lucru este posibil

- comparând partea dreaptă și partea stângă, obținem identitățile $u = b/\sqrt{a}$ și $v^2 = c - u^2$
- observăm că $v^2 = c - (b/\sqrt{a})^2 = c - b^2/a > 0$, care rezultă din faptul că determinantul este pozitiv

- deducem că v poate fi definit ca un număr real, și deci factorizarea Cholesky

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix} = \begin{bmatrix} \sqrt{a} & 0 \\ \frac{b}{\sqrt{a}} & \sqrt{c - b^2/a} \end{bmatrix} \begin{bmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \sqrt{c - b^2/a} \end{bmatrix} = R^T R$$

există pentru matrici 2×2 simetrice și pozitiv definite

- factorizarea Cholesky nu este unică: puteam la fel de bine să alegem v ca fiind rădăcina pătrată a lui $c - b^2/a$ cu semnul minus
- următorul rezultat garantează că aceeași idee funcționează și pentru cazul $n \times n$

Teorema 3 (Teorema factorizării Cholesky)

Dacă A este o matrice $n \times n$ simetrică și pozitiv definită, atunci există o matrice $n \times n$ superior triangulară R astfel încât $A = R^T R$.

- construim pe R prin inducție după n
- cazul $n = 2$ a fost făcut mai sus

- considerăm că A este partitioanată ca

$$A = \left[\begin{array}{c|c} a & b^T \\ \hline b & C \end{array} \right]$$

unde b este un $(n - 1)$ -vector și C este o submatrice $(n - 1) \times (n - 1)$

- vom folosi înmulțirea pe blocuri pentru a simplifica argumentul
- luăm $u = b/\sqrt{a}$, ca în cazul 2×2
- luând $A_1 = C - uu^T$ și definind matricea inversabilă

$$S = \left[\begin{array}{c|c} \sqrt{a} & u^T \\ \hline 0 & I \\ \vdots & \\ 0 & I \end{array} \right]$$

- avem că

$$\begin{aligned} S^T \left[\begin{array}{c|c} 1 & 0 & \dots & 0 \\ \hline 0 & A_1 \\ \vdots & & & \\ 0 & & & \end{array} \right] S &= \left[\begin{array}{c|c} \sqrt{a} & 0 & \dots & 0 \\ \hline u & I \\ \vdots & & & \\ 0 & & & \end{array} \right] \left[\begin{array}{c|c} 1 & 0 & \dots & 0 \\ \hline 0 & A_1 \\ \vdots & & & \\ 0 & & & \end{array} \right] \left[\begin{array}{c|c} \sqrt{a} & u^T \\ \hline 0 & I \\ \vdots & \\ 0 & I \end{array} \right] \\ &= \left[\begin{array}{c|c} a & b^T \\ \hline b & uu^T + A_1 \end{array} \right] = A. \end{aligned}$$

- observăm că A_1 este simetrică și pozitiv definită

- aceasta rezultă din faptul că

$$\left[\begin{array}{c|c} 1 & 0 & \dots & 0 \\ \hline 0 & A_1 \\ \vdots & & & \\ 0 & & & \end{array} \right] = (S^T)^{-1} A S^{-1}$$

este simetrică și pozitiv definită din Faptul 2, și la fel este și submatricea principală $(n - 1) \times (n - 1)$ A_1 din Faptul 3

- din ipoteza de inducție, $A_1 = V^T V$, unde V este superior triangulară

- în final, definim matricea superior triangulară

$$R = \left[\begin{array}{c|c} \sqrt{a} & u^T \\ \hline 0 & V \\ \vdots & \\ 0 & V \end{array} \right]$$

și verificăm că

$$R^T R = \left[\begin{array}{c|c} \sqrt{a} & 0 & \dots & 0 \\ \hline u & V^T \\ \vdots & & & \\ 0 & & & \end{array} \right] \left[\begin{array}{c|c} \sqrt{a} & u^T \\ \hline 0 & V \\ \vdots & \\ 0 & V \end{array} \right] = \left[\begin{array}{c|c} a & b^T \\ \hline b & uu^T + V^T V \end{array} \right] = A,$$

încheind astfel demonstrația

- construcția demonstrației se poate face explicit, în ceea ce a devenit algoritmul standard pentru factorizarea Cholesky
- matricea R este construită din exterior înspre interior
- mai întâi, găsim $r_{11} = \sqrt{a_{11}}$ și luăm restul primului rând al lui R ca fiind $u^T = b^T/r_{11}$
- atunci uu^T este scăzut din submatricea principală $(n - 1) \times (n - 1)$ inferioară, și aceiași pași sunt repetați pentru a umple al doilea rând al lui R
- acești pași sunt continuați până când toate rândurile lui R sunt determinante
- potrivit teoremei, noua submatrice principală este pozitiv definită la fiecare pas al construcției, deci, din Faptul 3, rezultă că intrarea din colțul din stânga sus este pozitivă, și extragerea rădăcinii pătrate este validă
- această abordare poate fi pusă direct în următorul algoritm
- folosim notația cu „două puncte” pentru a specifica submatricile respective

Algoritm 4 (Factorizarea Cholesky)

```

for  $k = 1, 2, \dots, n$ 
    if  $A_{kk} < 0$ , stop, end
     $R_{kk} = \sqrt{A_{kk}}$ 
     $u^T = \frac{1}{R_{kk}} A_{k, k+1:n}$ 
     $R_{k, k+1:n} = u^T$ 
     $A_{k+1:n, k+1:n} = A_{k+1:n, k+1:n} - uu^T$ 
end

```

chol(A)

- matricea R rezultată este superior triangulară și satisface $A = R^T R$

```

cholesky.m
1 function R = cholesky(A)
2     n = size(A, 1);
3     U = zeros(n);
4
5     for k = 1 : n
6         if A(k, k) <= 0
7             error('Matricea nu este pozitiv definită.');
8         end
9
10        R(k, k) = sqrt(A(k, k));
11
12        U = (1/ R(k, k)) * A(k, k+1:n);
13
14        R(k, k+1:n) = U';
15
16    A(k+1 : n, k+1 : n) = A(k+1 : n, k+1 : n) - U' * U;
17
18 end

```

Command Window

```

>> A = [4 -2 2; -2 2 -4; 2 -4 11];
>> chol(A)

ans =

```

$$\begin{bmatrix} 2 & -1 & 1 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

```

>> x = cholesky(A)

x =

```

$$\begin{bmatrix} 2 & -1 & 1 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

Exemplul 8

găsiți factorizarea Cholesky a lui $\begin{bmatrix} 4 & -2 & 2 \\ -2 & 2 & -4 \\ 2 & -4 & 11 \end{bmatrix}$

$$R_{11} = \sqrt{a_{11}} = 2$$

$$R_{1,2:3} = [-2, 2] / R_{11} = [-1, 1] \Rightarrow R = \begin{bmatrix} 2 & -1 & 1 \\ | & | & | \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{bmatrix}$$

$$u \cdot u^T = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 1 \end{bmatrix} \stackrel{A_{2:3,2:3}}{\Rightarrow} \begin{bmatrix} 2 & -4 & 11 \\ -4 & 11 \end{bmatrix} - \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -3 \\ -3 & 10 \end{bmatrix}$$

Se repetă pașii pt submatricea 2×2 și găsim $R_{22} = 1$ $R_{23} = -3$

$$R = \left[\begin{array}{c|cc} 2 & -1 & 1 \\ \hline 0 & 1 & -3 \end{array} \right].$$

- submatricea principală 1×1 inferioară a lui A este $10 - (-3)(-3) = 1$, deci $R_{33} = \sqrt{1}$
- factorul Cholesky al lui A este

$$R = \left[\begin{array}{ccc} 2 & -1 & 1 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{array} \right].$$

- rezolvarea sistemului $Ax = b$ pentru matricea simetrică și pozitiv definită A urmează aceeași idee ca și factorizarea LU
- acum că $A = R^T R$ este un produs a două matrici triangulare, trebuie să rezolvăm sistemul inferior triangular $R^T c = b$ și sistemul superior triangular $Rx = c$ pentru a determina soluția x

3.5.3. Metoda gradienților conjugați

→ bazată pe generalizarea ideii cunoscute de produs scalar

- produsul scalar euclidian $(v, w) = v^T w$ este simetric și liniar în intrările v și w , deoarece $(v, w) = (w, v)$ și $(\alpha v + \beta w, u) = \alpha(v, u) + \beta(w, u)$, pentru orice scalari α și β
- produsul scalar euclidian este de asemenea și pozitiv definit, prin aceea că $(v, v) > 0$ dacă $v \neq 0$

Definiția 4

Fie A o matrice $n \times n$ simetrică și pozitiv definită. Pentru doi n -vectori v și w , definim **A -produsul scalar** prin $(v, w)_A = v^T Aw$. Vectorii v și w sunt A -conjuguați dacă $(v, w)_A = 0$.

- observăm că noul produs scalar moștenește proprietățile de simetrie, liniaritate și pozitiv definită de la matricea A
- deoarece A este simetrică, la fel este și A -produsul scalar: $(v, w)_A = v^T Aw = (v^T Aw)^T = w^T Av = (w, v)_A$

- A -produsul scalar este de asemenea liniar, iar pozitiv definită rezultă din faptul că dacă A este pozitiv definită, atunci

$$(v, v)_A = v^T Av > 0, \text{ dacă } v \neq 0$$

- în sens strict, metoda gradienților conjuguați este o metodă directă, și ajunge la soluția x a sistemului simetric și pozitiv definit $Ax = b$ cu următoarea buclă finită:

Algoritmul 5 (Metoda gradienților conjuguați)

```

 $x_0$  = vectorul inițial
 $d_0 = r_0 = b - Ax_0$ 
for  $k = 0, 1, 2, \dots, n - 1$ 
    if  $r_k = 0$ , stop, end
     $\alpha_k = \frac{r_k^T r_k}{d_k^T Ad_k}$ 
     $x_{k+1} = x_k + \alpha_k d_k$ 
     $r_{k+1} = r_k - \alpha_k Ad_k$ 
     $\beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ 
     $d_{k+1} = r_{k+1} + \beta_k d_k$ 
end

```

```

conjgrad.m + 
3 x = x0;
4
5 d = b - A * x0;
6 r = d;
7
8 for k = 1 : k
9 if r == 0
10 break
11 end
12 alfa = (r' * r) / (d' * A * d);
13
14 x = x + alfa * d;
15
16 r_nou = r - alfa * A * d;
17
18 beta = (r_nou' * r_nou) / (r' * r);
19
20 d = r_nou + beta * d;
21
22 r = r_nou;
23 end
24

```

Command Window

```

>> A = [2 2; 2 5];
>> b = [6;3];
>> x0 = zeros(2,1);
>> x = conjgrad(A, b, x0, 20);
>> x = conjgrad(A, b, x0, 20)

x =
4.0000
-1.0000

```

→ iteratia gradientilor conjugati actualizeaza trei vectori diferiti la fiecare pas

→ x_k - aproximarea solutiei la pasul k

→ r_k - rezidualul aproximatiei x_k

- acest lucru este clar pentru r_0 din definicie, si, in timpul iteratiei, observam ca

$$\begin{aligned} Ax_{k+1} + r_{k+1} &= A(x_k + \alpha_k d_k) + r_k - \alpha_k Ad_k \\ &= Ax_k + r_k, \end{aligned}$$

si astfel prin inducție $r_k = b - Ax_k$, pentru orice k

- in final, vectorul d_k reprezinta noua directie de căutare folosita pentru a actualiza aproximarea x_k in versiunea imbunatatita x_{k+1}
- metoda reușește deoarece fiecare rezidual este definit astfel incat sa fie ortogonal pe toți rezidualii anteriori
- dacă acest lucru poate fi făcut, metoda va rămâne fără direcții ortogonale în care să caute și va trebui să ajungă la un rezidual egal cu zero și la o soluție corectă în cel mult n pași
- cheia pentru a realiza ortogonalitatea între reziduali se dovedește a fi alegerea directiilor de căutare d_k ca fiind conjugate două câte două
- conceptul de conjugare generalizează conceptul de ortogonalitate, dând și numele algoritmului
- acum vom explica alegerile lui α_k și β_k
- directiile d_k sunt alese din subspațiul liniar generat de rezidualii anteriori, după cum se poate vedea prin inducție din ultima linie a buclei algoritmului
- pentru a asigura faptul că urmatorul rezidual este ortogonal pe toți rezidualii anteriori, α_k este ales tocmai pentru ca noul rezidual r_{k+1} să fie ortogonal pe directia d_k :

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k d_k \\ b - Ax_{k+1} &= b - Ax_k - \alpha_k Ad_k \\ r_{k+1} &= r_k - \alpha_k Ad_k \\ 0 = d_k^T r_{k+1} &= d_k^T r_k - \alpha_k d_k^T Ad_k \\ \alpha_k &= \frac{d_k^T r_k}{d_k^T Ad_k}. \end{aligned}$$

- aceasta nu este exact forma lui α_k din algoritm, dar observăm că, deoarece d_{k-1} este ortogonal pe r_k , avem

$$\begin{aligned} d_k - r_k &= \beta_{k-1} d_{k-1} \\ r_k^T d_k - r_k^T r_k &= 0, \end{aligned}$$

ceea ce justifică scrierea $r_k^T d_k = r_k^T r_k$

- în al doilea rând, coeficientul β_k este ales pentru a asigura A -conjugarea direcțiilor d_k două câte două:

$$\begin{aligned} d_{k+1} &= r_{k+1} + \beta_k d_k \\ 0 = d_k^T A d_{k+1} &= d_k^T A r_{k+1} + \beta_k d_k^T A d_k \\ \beta_k &= -\frac{d_k^T A r_{k+1}}{d_k^T A d_k}. \end{aligned}$$

- expresia pentru β_k poate fi rescrisă în forma mai simplă folosită în algoritm, după cum se arată în ecuația (9) de mai jos
- Teorema 4 de mai jos verifică faptul că toți rezidualii r_k produși de către iterația gradientilor conjugati sunt ortogonali unii pe alții
- deoarece ei sunt vectori n -dimensionali, cel mult n din rezidualii r_k pot fi ortogonali doi căte doi, deci ori r_n ori un r_k anterior trebuie să fie zero, rezolvând astfel $Ax = b$
- prin urmare, după cel mult n pași, metoda gradientilor conjugati ajunge la o soluție
- în teorie, metoda este una directă, și nu una iterativă
- înainte de a ajunge la teorema care garantează succesul metodei gradientilor conjugati, este instructiv să urmărim un exemplu în aritmetică exactă

Exemplul 9

$$\begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \end{bmatrix} \text{ cu } m. \text{ grad. conjugati}$$

urmând algoritmul de mai sus, avem că

$$\begin{aligned} x_0 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad r_0 = d_0 = \begin{bmatrix} 6 \\ 3 \end{bmatrix} \\ \alpha_0 &= \frac{\begin{bmatrix} 6 \\ 3 \end{bmatrix}^T \begin{bmatrix} 6 \\ 3 \end{bmatrix}}{\begin{bmatrix} 6 \\ 3 \end{bmatrix}^T \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 6 \\ 3 \end{bmatrix}} = \frac{45}{6 \cdot 18 + 3 \cdot 27} = \frac{5}{21} \\ x_1 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \frac{5}{21} \begin{bmatrix} 6 \\ 3 \end{bmatrix} = \begin{bmatrix} 10/7 \\ 5/7 \end{bmatrix} \\ r_1 &= \begin{bmatrix} 6 \\ 3 \end{bmatrix} - \frac{5}{21} \begin{bmatrix} 18 \\ 27 \end{bmatrix} = \begin{bmatrix} 12/7 \\ -24/7 \end{bmatrix} \\ \beta_0 &= \frac{r_1^T r_1}{r_0^T r_0} = \frac{144 \cdot 5/49}{36 + 9} = \frac{16}{49} \\ d_1 &= \begin{bmatrix} 12/7 \\ -24/7 \end{bmatrix} + \frac{16}{49} \begin{bmatrix} 6 \\ 3 \end{bmatrix} = \begin{bmatrix} 180/49 \\ -120/49 \end{bmatrix} \\ \alpha_1 &= \frac{\begin{bmatrix} 12/7 & -24/7 \end{bmatrix}^T \begin{bmatrix} 12/7 \\ -24/7 \end{bmatrix}}{\begin{bmatrix} 180/49 & -120/49 \end{bmatrix}^T \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 180/49 \\ -120/49 \end{bmatrix}} = \frac{7}{10} \\ x_2 &= \begin{bmatrix} 10/7 \\ 5/7 \end{bmatrix} + \frac{7}{10} \begin{bmatrix} 180/49 \\ -120/49 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \end{bmatrix} \\ r_2 &= \begin{bmatrix} 12/7 \\ -24/7 \end{bmatrix} - \frac{7}{10} \begin{bmatrix} 2 & 2 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 180/49 \\ -120/49 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \end{aligned}$$

deoarece $r_2 = b - Ax_2 = 0$, soluția este $x_2 = [4, -1]^T$

Teorema 4

Fie A o matrice $n \times n$ simetrică și pozitiv definită și fie $b \neq 0$ un vector. În metoda gradientilor conjugati, presupunem că $r_k \neq 0$ pentru $k < n$ (dacă $r_k = 0$, ecuația este rezolvată). Atunci, pentru orice $1 \leq k \leq n$,

- (a) următoarele trei subspații liniare ale lui \mathbb{R}^n sunt egale:

$$\langle x_1, \dots, x_k \rangle = \langle r_0, \dots, r_{k-1} \rangle = \langle d_0, \dots, d_{k-1} \rangle,$$

- (b) rezidualii r_k sunt ortogonali doi căte doi: $r_k^T r_j = 0$ pentru $j < k$,
(c) direcțiile d_k sunt A -conjugate două căte două: $d_k^T A d_j = 0$ pentru $j < k$.

- (a) pentru $k = 1$, observăm că $\langle x_1 \rangle = \langle d_0 \rangle = \langle r_0 \rangle$, deoarece $x_0 = 0$
- prin definiție $x_k = x_{k-1} + \alpha_{k-1} d_{k-1}$
- aceasta implică prin inducție faptul că $\langle x_1, \dots, x_k \rangle = \langle d_0, \dots, d_{k-1} \rangle$
- un argument similar folosind $d_k = r_k + \beta_{k-1} d_{k-1}$ arată că $\langle r_0, \dots, r_{k-1} \rangle$ este egal cu $\langle d_0, \dots, d_{k-1} \rangle$
- pentru (b) și (c), folosim inducția
- dacă $k = 0$ nu există nimic de demonstrat

- presupunem că (b) și (c) au loc pentru k , și vom demonstra (b) și (c) pentru $k + 1$
- înmulțim definiția lui r_{k+1} cu r_j^T la stânga:

$$r_j^T r_{k+1} = r_j^T r_k - \frac{r_k^T r_k}{d_k^T A d_k} r_j^T A d_k. \quad (8)$$

- dacă $j \leq k - 1$, atunci $r_j^T r_k = 0$ din ipoteza de inducție (b)
- deoarece r_j poate fi exprimat ca o combinație de d_0, \dots, d_j , avem $r_j^T A d_k = 0$ din ipoteza de inducție (c), și deci (b) are loc
- pe de altă parte, dacă $j = k$, atunci $r_k^T r_{k+1} = 0$ rezultă din nou din (8), deoarece $d_k^T A d_k = r_k^T A d_k + \beta_{k-1} d_{k-1}^T A d_k = r_k^T A d_k$, folosind ipoteza de inducție (c)
- aceasta demonstrează (b)
- acum că $r_k^T r_{k+1} = 0$, ecuația (8) cu $j = k + 1$ ne spune că

$$\frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} = -\frac{r_{k+1}^T A d_k}{d_k^T A d_k}. \quad (9)$$

- aceasta, împreună cu înmulțirea definiției lui d_{k+1} la stânga cu $d_j^T A$, ne dau

$$d_j^T A d_{k+1} = d_j^T A r_{k+1} - \frac{r_{k+1}^T A d_k}{d_k^T A d_k} d_j^T A d_k. \quad (10)$$

- dacă $j = k$, atunci $d_k^T A d_{k+1} = 0$ din (10), folosind simetria lui A
- dacă $j \leq k - 1$, atunci $A d_j = (r_j - r_{j+1})/\alpha_j$ (din definiția lui r_{k+1}) este ortogonal pe r_{k+1} , arătând că primul termen din partea dreaptă a lui (10) este zero, și al doilea termen este zero din ipoteza de inducție, ceea ce completează argumentul pentru (c)
- în Exemplul 9, observăm că r_1 este ortogonal pe r_0 , după cum ne garantează Teorema 4
- acest fapt este cheia succesului metodei gradientilor conjugăți: fiecare nou rezidual r_i este ortogonal pe toți rezidualii r_j anteriori
- dacă unul dintre r_i se dovedește a fi zero, atunci $A x_i = b$ și x_i este soluția
- dacă nu, după n pași ai buclei, r_n este ortogonal pe spațiul liniar generat de ceil n vectori ortogonali doi câte doi r_0, \dots, r_{n-1} , care trebuie să fie egal cu \mathbb{R}^n
- deci r_n trebuie să fie vectorul zero, și $A x_n = b$

3. 6. Sisteme de ecuații neliniare

3.6.1. Metoda lui Newton pentru mai multe variabile

Pentru o singură variabilă: $x_{h+1} = x_h - \frac{f(x_h)}{f'(x_h)}$

- de exemplu, fie

$$\begin{aligned} f_1(u, v, w) &= 0 \\ f_2(u, v, w) &= 0 \\ f_3(u, v, w) &= 0 \end{aligned} \quad (11)$$

trei ecuații neliniare în trei necunoscute u, v, w

- definim funcția cu valori vectoriale $F(u, v, w) = (f_1, f_2, f_3)$, și notăm problema (11) prin $F(x) = 0$, unde $x = (u, v, w)$

- analogul derivatei f' din cazul unei singure variabile este **matricea jacobiana** definită prin

$$DF(x) = \begin{bmatrix} \frac{\partial f_1}{\partial u} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial w} \\ \frac{\partial f_2}{\partial u} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial w} \\ \frac{\partial f_3}{\partial u} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial w} \end{bmatrix}.$$

- dezvoltarea în serie Taylor pentru funcții vectoriale în jurul lui x_0 este

$$F(x) = F(x_0) + DF(x_0) \cdot (x - x_0) + O((x - x_0)^2).$$

- scriem $f(x) = O(g(x))$ dacă și numai dacă există $M > 0$ astfel încât $|f(x)| \leq M|g(x)|$
- de exemplu, dezvoltarea liniară a lui $F(u, v) = (e^{u+v}, \sin u)$ în jurul lui $x_0 = (0, 0)$ este

$$\begin{aligned} F(x) &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} e^0 & e^0 \\ \cos 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + O(x^2) \\ &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} u + v \\ u \end{bmatrix} + O(x^2). \end{aligned}$$

- metoda lui Newton se bazează pe o aproximare liniară, ignorând termenii $O(x^2)$

- ca în cazul unei singure variabile, fie $x = r$ rădăcina, și fie x_0 aproximarea curentă

- atunci

$$0 = F(r) \approx F(x_0) + DF(x_0) \cdot (r - x_0),$$

sau

$$-DF(x_0)^{-1}F(x_0) \approx r - x_0. \quad (12)$$

- prin urmare, o mai bună aproximare a rădăcinii rezultă din rezolvarea ecuației (12) pentru a găsi r

Algoritm 6 (Metoda lui Newton pentru mai multe variabile)

$$\begin{aligned} x_0 &= \text{vectorul inițial} \\ x_{k+1} &= x_k - (DF(x_k))^{-1}F(x_k) \text{ for } k = 0, 1, 2, \dots \end{aligned}$$

```
newton.m x +
1 function x = newton(F, DF, x0, k)
2
3 x = x0;
4
5 for i = 1:k
6 x = x - inv(DF(x)) * F(x);
7 end
8
9
10
11 % F = @(x) [x(1) - x(2)^3; x(1)^2 + x(2)^2 - 1];
12 % DF = @(x) [1, -3*x(2)^2; 2*x(1), 2*x(2)];
13 % x = newton(F, DF, [1;2], 10)
```

```
Command Window
>> F = @(x) [x(1) - x(2)^3; x(1)^2 + x(2)^2 - 1];
>> DF = @(x) [1, -3*x(2)^2; 2*x(1), 2*x(2)];
>> x = newton(F, DF, [1;2], 10)

x =
    0.5636
    0.8260
```

- deoarece calculul inverselor este dificil din punct de vedere computațional, vom folosi un truc pentru a evita acest calcul
- la fiecare pas, în loc să urmăm definiția anterioară literal, luăm $x_{k+1} = x_k - s$, unde s este soluția ecuației $DF(x_k)s = F(x_k)$
- acum, doar eliminarea gaussiană este necesară pentru a realiza un pas al metodei, în loc de calculul inversei, care necesită de trei ori mai multe operații
- prin urmare, pasul de iterare pentru metoda lui Newton pentru mai multe variabile este

$$\begin{cases} DF(x_k)s = -F(x_k) \\ x_{k+1} = x_k + s. \end{cases} \quad (13)$$

Exemplul 10

$$x_0 = [1; 2]$$

$$v - u^3 = 0$$

$$u^2 + v^2 - 1 = 0$$

$$f_1(u, v) = -u^3 + v$$

$$f_2(u, v) = u^2 + v^2 - 1$$

$$\Delta F(u, v) = \begin{bmatrix} -3u^2 & 1 \\ 2u & 2v \end{bmatrix}$$

$$I \quad x_0 = (1, 2)$$

$$\begin{bmatrix} -3 & 1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = - \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

$$s = (0, -1)$$

$$\Rightarrow x_1 = x_0 + s = (1, 1)$$

II

$$\begin{bmatrix} -3 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = - \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\Rightarrow s = (-\frac{1}{8}, -\frac{3}{8})$$

$$\Rightarrow x_2 = x_1 + s = (\frac{7}{8}, \frac{5}{8})$$

Figura 1: Metoda lui Newton pentru Exemplul 10. Cele două rădăcini sunt punctele de pe cerc. Metoda lui Newton produce punctele care converg către soluția aproximativă (0.8260, 0.5636).

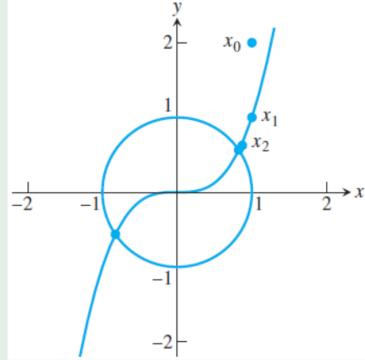
$$-f_1(x_0) = -(-1+2) = -1$$

$$-f_2(x_0) = -(1+4-1) = -4$$

$$-f_1(x_1) = -(1-1) = 0$$

$$-f_2(x_1) = -(1+1-1) = -1$$

pasul	u	v
0	1.000000000000000	2.000000000000000
1	1.000000000000000	1.000000000000000
2	0.875000000000000	0.625000000000000
3	0.82903634826712	0.56434911242604
4	0.82604010817065	0.56361977350284
5	0.82603135773241	0.56362416213163
6	0.82603135765419	0.56362416216126
7	0.82603135765419	0.56362416216126



- familiară dublare a numărului de zecimale exacte la fiecare pas, caracteristică pătratic convergenței, este evidentă în acest tabel
- simetria ecuațiilor arată că dacă (u, v) este o soluție, atunci și $(-u, -v)$ este o soluție, după cum se poate observa în Figura 1
- a două soluție poate fi de asemenea găsită prin aplicarea metodei lui Newton cu un punct initial aflat în vecinătatea acesteia

Exemplul 11

$$f_1(u, v) = 6u^3 + uv - 3v^3 - 4 = 0$$

$$f_2(u, v) = u^2 - 18uv^2 + 16v^3 + 1 = 0$$

$$(u, v) = (1, 1) \rightarrow \text{sol}$$

$$\delta F(u, v) = \begin{bmatrix} 18u^2 + v & u - 9v^2 \\ 2u - 18v^2 & -36uv + 48v^2 \end{bmatrix}$$

$$(u_0, v_0) = (2, 2)$$

pasul	u	v
0	2.000000000000000	2.000000000000000
1	1.37258064516129	1.34032258064516
2	1.07838681200443	1.05380123264984
3	1.00534968896520	1.00269261871539
4	1.00003367866506	1.00002243772010
5	1.00000000111957	1.00000000057894
6	1.000000000000000	1.000000000000000
7	1.000000000000000	1.000000000000000

- alți vectori inițiali conduc la alte două rădăcini, care sunt aproximativ $(0.865939, 0.462168)$ și $(0.886809, -0.294007)$
- metoda lui Newton este o alegere bună dacă jacobianul poate fi calculat
- dacă nu, cea mai bună alternativă este metoda lui Broyden, subiectul subsecțiunii următoare

3.6.2. Metoda lui Broyden

- metoda lui Newton pentru rezolvarea unei ecuații într-o necunoscută necesită cunoașterea derivatei
- dezvoltarea acestei metode în Capitolul 2 a fost urmată de discuția despre metoda secantei, folositoare pentru cazul în care derivata nu este disponibilă sau este prea greu de evaluat
- acum că avem o versiune a metodei lui Newton pentru sisteme de ecuații neliniare $F(x) = 0$, ne confruntăm cu aceeași întrebare: Ce se întâmplă dacă matricea jacobiană DF nu este disponibilă?
- deși nu există o extensie simplă a metodei lui Newton la o metodă a secantei pentru sisteme, Broyden a sugerat o metodă care se apropie cel mai mult de acest deziderat
- să presupunem că A_i este cea mai bună aproximare a matricii jacobiene disponibilă la pasul i , și că a fost folosită pentru a da naștere următoarei aproximări astfel:

$$x_{i+1} = x_i - A_i^{-1} F(x_i). \quad (14)$$

- pentru a actualiza A_i la A_{i+1} pentru pasul următor, am dori să respectăm aspectul de derivată al jacobianului DF , care trebuie să satisfacă

$$A_{i+1} \delta_{i+1} = \Delta_{i+1}, \quad (15)$$

unde $\delta_{i+1} = x_{i+1} - x_i$ și $\Delta_{i+1} = F(x_{i+1}) - F(x_i)$

- pe de altă parte, pentru complementul ortogonal al lui δ_{i+1} , nu avem alte informații noi
- prin urmare, cerem ca

$$A_{i+1} w = A_i w \quad (16)$$

pentru orice w care satisfacă $\delta_{i+1}^T w = 0$

- o matrice care satisfacă atât (15) cât și (16) este

$$A_{i+1} = A_i + \frac{(\Delta_{i+1} - A_i \delta_{i+1}) \delta_{i+1}^T}{\delta_{i+1}^T \delta_{i+1}}. \quad (17)$$

- metoda lui Broyden folosește pasul din metoda lui Newton (14) pentru a găsi aproximarea următoare, și actualizează aproximarea jacobianului ca în (17)
- rezumând, algoritmul începe cu un vector inițial x_0 și o aproximare inițială a jacobianului A_0 , care poate fi aleasă matricea identitate

Algoritmul 7 (Metoda lui Broyden I)

x_0 = vectorul inițial
 A_0 = matricea inițială
for $i = 0, 1, 2, \dots$

$$x_{i+1} = x_i - A_i^{-1} F(x_i)$$

$$A_{i+1} = A_i + \frac{(\Delta_{i+1} - A_i \delta_{i+1}) \delta_{i+1}^T}{\delta_{i+1}^T \delta_{i+1}}.$$
end
unde $\delta_{i+1} = x_{i+1} - x_i$ și $\Delta_{i+1} = F(x_{i+1}) - F(x_i)$.

```
broyden.m
1 function x = broyden(F, x0, k)
2
3 n = length(x0);
4
5 A = eye(n, n);
6
7 for i = 1 : k
8
9 x = x0 - inv(A) * F(x0);
10
11 delta = x - x0;
12
13 Delta = F(x) - F(x0);
14
15 A = A + ((Delta - A * delta) * delta') / (delta' * delta);
16
17 x0 = x;
18 end
% ex lab 3
% F=@(x) [x(2)-x(1)^3;x(1)^2+x(2)^2-1];
% x=broyden(F,[1;1],10)
```

```
Command Window
>> F=@(x) [x(2)-x(1)^3;x(1)^2+x(2)^2-1];
>> x=broyden(F,[1;1],10)

x =
0.8260
0.5636
```

- observăm că pasul de tip Newton se realizează prin rezolvarea ecuației $A_i \delta_{i+1} = F(x_i)$, la fel ca în metoda lui Newton
- tot ca metoda lui Newton, metoda lui Broyden nu oferă garanția convergenței către o soluție
- o a doua abordare a metodei lui Broyden evită pasul relativ dificil din punct de vedere computațional $A_i \delta_{i+1} = F(x_i)$
- deoarece oricum doar aproximăm derivata DF în timpul iterării, mai bine aproximăm inversa lui DF , cea de care avem nevoie în pasul Newton

- refacem deducerea metodei lui Broyden din punctul de vedere al matricii $B_i = A_i^{-1}$

- ne-ar plăcea să avem

$$\delta_{i+1} = B_{i+1} \Delta_{i+1}, \quad (18)$$

unde $\delta_{i+1} = x_{i+1} - x_i$ și $\Delta_{i+1} = F(x_{i+1}) - F(x_i)$, și, pentru orice w care satisfacă $\delta_{i+1}^T w = 0$, să avem $A_{i+1} w = A_i w$, sau

$$B_{i+1} A_i w = w. \quad (19)$$

- o matrice care satisfacă atât (18) cât și (19) este

$$B_{i+1} = B_i + \frac{(\delta_{i+1} - B_i \Delta_{i+1}) \delta_{i+1}^T B_i}{\delta_{i+1}^T B_i \Delta_{i+1}}. \quad (20)$$

- noua versiune a iterării, care nu necesită rezolvarea unei ecuații, este

$$x_{i+1} = x_i - B_i F(x_i). \quad (21)$$

- algoritmul rezultat se numește metoda lui Broyden II

Algoritmul 8 (Metoda lui Broyden II)

x_0 = vectorul inițial
 B_0 = matricea inițială
for $i = 0, 1, 2, \dots$

$$x_{i+1} = x_i - B_i F(x_i)$$

$$B_{i+1} = B_i + \frac{(\delta_{i+1} - B_i \Delta_{i+1}) \delta_{i+1}^T B_i}{\delta_{i+1}^T B_i \Delta_{i+1}}$$
end
unde $\delta_{i+1} = x_{i+1} - x_i$ și $\Delta_{i+1} = F(x_{i+1}) - F(x_i)$.

- la început, un vector inițial x_0 și o aproximare inițială pentru B_0 sunt necesare
- dacă este imposibil să se calculeze derivate, alegerea $B_0 = I$ poate fi folosită

```

broyden2.m + 
1 function x = broyden2(F,x0,k)
2
3 n = length(x0);
4
5 B = eye(n,n);
6
7 for i = 1:k
8     x = x0 - B * F(x0);
9
10    delta = x - x0;
11
12    Delta = F(x) - F(x0);
13
14    B = B + (delta - B * Delta) * delta' * B / (delta' * B * Delta);
15
16    x0 = x;
17
18 end
19 % Lab 3
20 % F=@(x) [x(2)-x(1)^3;x(1)^2+x(2)^2-1];
21 % x=broyden2(F,[1;1],10)

```

Command Window

```

>> F=@(x) [x(2)-x(1)^3;x(1)^2+x(2)^2-1];
>> x=broyden2(F,[1;1],10)

x =
0.8260
0.5636

```

- un dezavantaj al metodei lui Broyden II este că aproximările jacobianului, necesare pentru anumite aplicații, nu sunt ușor disponibile
- matricea B_i este o aproximare a inversei matricii jacobiene
- metoda lui Broyden I, pe de altă parte, actualizează pe A_i , care reprezintă o aproximare a jacobianului
- ambele versiuni ale metodei lui Broyden au o convergență superliniară, fiind ușor mai lente decât convergența pătratică a metodei lui Newton
- dacă o formulă a jacobianului este disponibilă, are loc o accelerare a convergenței dacă se folosește inversa lui $DF(x_0)$ pe post de matrice inițială B_0