Temă la disciplina
Analiza Algoritmilor
Rezolvarea sistemelor de ecuații liniare
Etapa 1

Buzea Alexandru-Mihai-Iulian, grupa 321CA
alexandru.buzea2007@stud.acs.upb.ro

Facultatea de Automatică și Calculatoare
Calculatoare și Tehnologia Informației

Abstract. Prezentarea unor metode folosite pentru soluționarea sistemelor de ecuații liniare, precum și a unor avantaje/dezavantaje ale acestora.

Keywords: GPP · GPPS · Householder · Conjugate Gradient

1 Introducere

1.1 Descrierea problemei rezolvate

Sistemele de ecuații liniare reprezintă una dintre cele mai importante teme din analiza numerică, cu aplicații extrem de răspândite în Computer Science. Majoritatea proceselor din natură ce se doresc a fi modelate cu ajutorul aplicațiilor software pot fi descrise foarte precis prin rezolvarea sistemelor de ecuații liniare. De asemenea, multe modele neliniare sau dinamice (descrise prin ecuații diferențiale) pot fi descrise local sau global cu ajutorul sistemelor de ecuații liniare.
La modul general, un sistem de ecuații liniare are m ecuații și n necunoscute. În forma sa desfășurată, acesta se scrie sub următoarea formă:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \cdots + a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \cdots + a_{2n} \cdot x_n = b_2 \\ a_{31} \cdot x_1 + a_{32} \cdot x_2 + \cdots + a_{3n} \cdot x_n = b_3 \\ \cdots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \cdots + a_{mn} \cdot x_n = b_m \end{cases} \tag{1}$$

În mod similar, un sistem de ecuații liniare se poate scrie și în forma sa matriceală (mai compactă):

$$A \cdot x = b \tag{2}$$

unde $A \in \mathbb{R}^{m \times n}$ este matricea coeficienților (matricea sistemului), $x \in \mathbb{R}^{n \times 1}$ este vectorul necunoscutelor, iar $b \in \mathbb{R}^{m \times 1}$ este vectorul termenilor liberi.

În domeniul metodelor numerice, se întâlnesc în majoritatea situațiilor probleme bine formulate: sisteme de ecuații liniare în care matricea A este pătratică (sistem bine dimensionat) și nesingulară, caz în care vectorul necunoscutelor se poate calcula prin metoda trivială:

$$x = A^{-1} \cdot b \tag{3}$$

Principala problemă întâmpinată în acest caz este instabilitatea numerică și efortul computațional substanțial adus pentru a calcula inversa unei matrice ($O(n^3)$ pentru metoda Gauss-Jordan, cea mai folosită pentru inversarea de matrice). De aceea, este preferată folosirea unor metode exacte de rezolvare a sistemelor de ecuații (cum ar fi eliminarea gaussiană și factorizarea QR cu ajutorul transformărilor ortogonale Householder) sau a unor metode iterative (cum ar fi metoda gradientului, Gradient Descent sau metoda gradientului conjugat), care ne furnizează o soluție cât mai apropiată de cea reală.

Aplicație practică 1: Rezolvarea circuitelor electrice

Primul exemplu de aplicație practică a rezolvării sistemelor de ecuații liniare îl reprezintă rezolvarea de circuite electrice și electronice reale în cadrul unor simulatoare, cum ar fi **LTSpice XVII**. Acest simulator permite folosirea unor elemente de circuit cum ar fi: surse de tensiune/curent, rezistori, bobine, condensatoare, diode (cu o mare varietate de modele matematice pentru fiecare element în parte). În cazul simplu al unui circuit de curent continuu, cu elemente liniare de circuit, circuitul se poate reprezenta ca un graf orientat, în care nodurile circuitului corespund nodurilor din graf, iar fiecare latură este caracterizată de tensiunea sursei de tensiune (dacă aceasta există), curentul sursei de curent (dacă aceasta există), rezistența echivalentă etc. Apoi, folosind metoda potențialelor nodurilor, obținem un sistem de ecuații liniare cu $n - 1$ ecuații, unde n este numărul de noduri (un nod este considerat cu potențial 0). Acest sistem poate fi rezolvat (mai eficient) prin orice metodă exactă sau iterativă cunoscută.

Aplicație practică 2: Predicție bazată pe regresie liniară

Una dintre cele mai folosite metode de predicție în Machine Learning pentru output care poate lua orice valoare reală este regresia liniară. Spre exemplu, dorim să estimăm costul unui articol de îmbrăcăminte. Asociem cu un articol de îmbrăcăminte generic mai multe **calități/features**, cum ar fi mărimea, tipul de articol etc. Fiecare feature va contribui în mod semnificativ (sau nu) la costul articolului nostru, de aceea prețul unui articol va fi dat de o combinație liniară de valorile acestor calități/features luate în considerare, plus un termen constant (**bias**). Dacă notăm cu $X \in \mathbb{R}^{m \times (n+1)}$ [1] matricea tuturor calităților ale tuturor

[1] Apare n + 1 datorită faptului că avem n features pentru fiecare exemplu, plus un feature constant (bias). Astfel, matricea X va avea pe prima coloană numai 1.

exemplelor de antrenament ale modelului nostru de predicție, cu $\theta \in \mathbb{R}^{n+1}$ vectorul greutăților (weights) și cu $y \in \mathbb{R}^m$ prețurile reale pentru exemplele de antrenament, noi ne dorim să minimizăm următoarea funcție de cost:

$$J(\theta) = \frac{1}{2} \cdot \|X \cdot \theta - y\|^2 \tag{4}$$

Se poate demonstra analitic că funcția de cost dată de formula de mai sus are un minim global, dat de următoarea ecuație matriceală:

$$X^T X \cdot \theta = X^T y \tag{5}$$

Relația de mai sus reprezintă un sistem de ecuații liniar, în care matricea sistemului este $A = X^T X$, iar vectorul termenilor liberi este $X^T y$. În situația cea mai probabilă ca cele $n+1$ calități/features să fie liniar independente, matricea $X^T X$ este inversabilă, deci sistemul nostru are soluție unică.

Un mare avantaj al acestui sistem este însă faptul că matricea sistemului A este **simetrică și pozitiv-definită**. Astfel, sistemul nostru de ecuații poate fi rezolvat cu ajutorul metodei **gradientului conjugat**, ajungându-se la soluția exactă în exact n pași.

Aplicație practică 3: Rezolvarea sistemelor de ecuații neliniare

Sistemele neliniare de ecuații pot fi descrise generic de o funcție de forma $F : \mathbb{R}^n \to \mathbb{R}^n$. Rezolvarea sistemului de ecuații neliniare presupune a găsi un vector $x \in \mathbb{R}^n$ astfel încât:

$$F(x) = 0 \tag{6}$$

Pentru rezolvarea unui astfel de sistem de ecuații (în anumite condiții), se poate folosi metoda iterativă Newton, astfel: se pleacă de la o aproximație inițială x_0 a soluției sistemului și se folosește următoarea recurență (similară cu metoda lui Newton pentru rezolvarea ecuațiilor neliniare):

$$x_{k+1} = x_k - J(x_k)^{-1} \cdot F(x_k) \tag{7}$$

unde $J(x_k)$ este matricea jacobiană corespunzătoare transformării F în punctul x_k. Relația de mai sus poate fi adusă la forma:

$$J(x_k) \cdot (x_k - x_{k+1}) = F(x_k) \tag{8}$$

Relația de mai sus reprezintă un sistem de ecuații liniare, rezolvabil pentru a putea astfel determina o aproximare mai bună a soluției sistemului.

1.2 Specificarea soluțiilor alese

Pentru analiză, am ales patru algoritmi de rezolvare a sistemelor de ecuații liniare: eliminarea gaussiană (în cele două variante principale - eliminarea gaussiană cu pivotare parțială (GPP) și eliminarea gaussiană cu pivotare parțială cu pivot scalat (GPPS)), factorizarea QR folosind transformări ortogonale Householder și metoda gradientului conjugat/Conjugate Gradient.

Eliminarea gaussiană

Eliminarea gaussiană este cel mai simplu și intuitiv algoritm de determinare a soluției unui sistem de ecuații liniare. În cadrul acestui algoritm, se pot realiza următoarele operații pe ecuațiile unui sistem (respectiv, pe liniile matricei sistemului A), astfel:

- Adunarea la o ecuație (linie) a oricărei alte ecuații (linii).
- Înmulțirea unei ecuații (linii) cu orice număr real nenul.
- Adunarea la o ecuație (linie) a oricărei alte ecuații (linii), înmulțită cu un număr real (operație derivată din primele 2 operații).
- Permutarea ecuațiilor (liniilor) în orice ordine (caz particular: interschimbarea a două ecuații (linii)).

Principalele etape ale algoritmului de eliminare gaussiană simplă sunt:

- Cât timp mai există linii/coloane de prelucrat, ne alegem un **pivot** (situat pe diagonala principală).
- Scădem din toate liniile care urmează după linia pivotului linia pe care se află pivotul, scalată (astfel încât coeficientul corespunzător necunoscutei pivotului în aceste ecuații să fie 0, adică sub pivot să avem numai elemente egale cu 0).
- Trecem la următorul element de pe diagonala principală (pivot).
- La finalul algoritmului, obținem un sistem superior triunghiular echivalent cu cel inițial, pe care îl putem analiza cu scopul de a-i stabili caracterul (are/nu are soluție unică), precum și a-i determina soluția (dacă aceasta există).
- După terminarea algoritmului, problema originală s-a redus la rezolvare aunui sistem superior triunghiular, care poate fi rezolvat foarte simplu prin substituție înapoi.

O problemă semnificativă este dată de posibilitatea apariției a unor erori foarte mari pe anumite cazuri particulare datorită unui pivot cu valoare absolută mică (astfel că la calcularea factorului de scalare putem avea erori foarte mari de aproximare și instabilitate numerică). De aceea, se folosesc strategii de scalare, dintre care putem menționa eliminarea gaussiană cu **pivotare parțială** și eliminarea gaussiană cu **pivotare parțială cu pivot scalat**.

În cadrul eliminării gaussiene cu pivotare parțială(**GPP**), înainte de a realiza eliminarea propriu-zisă, se determină cel mai mare element de pe coloana pivotului, începând cu acesta (cu alte cuvinte, cel mai mare element de sub pivot). Apoi, se interschimbă linia pivotului cu linia elementului descoperit, astfel aducând pe poziția pivotului cel mai mare element (micșorând eroarea generată de scalarea la un pivot mic).

În cadrul eliminării gaussiene cu pivotare parțială cu pivot scalat (**GPPS**), se dorește găsirea unui nou pivot (în mod asemănător cu GPP). De această

dată, se va alege cel mai mare element de sub pivot raportat (scalat) la maximul (în valoare absolută) de pe linia respectivului element. Astfel, îmbunătățim stabilitatea numerică a algoritmului nostru cu costul unui efort computațional suplimentar depus.

Factorizarea QR folosind Householder

Una dintre cele mai folosite metode pentru soluționarea sistemelor de ecuații liniare este factorizarea QR. Această metodă de rezolvare are marele avantaj de a fi eficientă mai ales asupra sistemelor de ecuații cu matrice rare, unde folosirea unor metode tradiționale precum factorizarea LU nu este de dorit (nu exploatăm calitatea matricei sistemului de a fi matrice rară). Am ales factorizarea QR folosind transformări ortogonale Householder deoarece, spre deosebire de celelalte variante existente (aplicarea algoritmului Gram-Schmidt și factorizarea folosind matrice de rotație - Givens) timpul de execuție este mai redus și stabiitatea numerică este mai mare, deși complexitatea temporală este aceeași ($O(n^3)$ pentru matrice pătratice).

Ideea algoritmului este de a construi, pentru fiecare vector coloană din matricea ce se dorește a fi factorizată, un reflector (matrice ortogonală, simetrică, notată cu H, cu proprietatea că $H^2 = I_n$) denumit și reflector elementar Householder. Acest reflector, aplicat unui vector coloană z are efectul de a reflecta acest vector pe un subspațiu vectorial (mai exact, de a păstra primele k componente din cele n ale vectorului nostru nenule, restul fiind nule). Pentru un vector coloană z, se parcurg următorii pași pentru a factoriza matricea noastră:

- Pe coloana numărul k din matrice, se alege elementul de pe poziția k (de pe diagonala principală) drept pivot.
- Se extrage z, vectorul coloană din matrice ce are pivotul drept prim element.
- Se completează vectorul z cu zerouri (deasupra pivotului) pentru a obține un vector de dimensiune n, care va reprezenta vectorul director al reflectorului elementar (hiperplan față de care se reflectă toți vectorii din matricea A).
- Se calculează reflectorul elementar Householder conform următoarei formule:

$$H = I_n - 2\frac{vv^T}{v^T v} \tag{9}$$

- Se actualizează matricea ortogonală Q și matricea superior-triunghiulară R:

$$Q \leftarrow H \cdot Q \tag{10}$$

$$A \leftarrow H \cdot A \tag{11}$$

- Se continuă cu următoarea coloană.

Alogritmul continuă cât timp există coloane și linii disponibile pentru a efectua factorizarea. Odată realizată factorizarea QR, la fel ca la eliminarea

gaussiană, rezolvarea presupune rezolvarea sistemului superior triunghiular astfel obținut prin substituție înapoi, astfel:

$$A \cdot x = b \tag{12}$$

$$Q \cdot R \cdot x = b \tag{13}$$

Matricea Q este ortogonală, deci:

$$Q \cdot Q^T = Q^T \cdot Q = I_n \tag{14}$$

Astfel, inversa se calculează mult mai ușor, sistemul de rezolvat fiind dat de:

$$R \cdot x = Q^T \cdot b \tag{15}$$

Prin analiza corespunzătoare a matricei R, se poate astfel da un răspuns exact cu privire la compatibilitatea și determinarea sistemului.

Metoda gradientului conjugat

Spre deosebire de metodele exacte prezentate până acum, metoda gradientului conjugat iese în evidență prin câteva dezavantaje pe care le are:

- Pentru a putea folosi metoda gradientului conjugat, matricea sistemului trebuie să fie simetrică și pozitiv-definită (o condiție destul de restrictivă în general, dar care nu reprezintă o problemă în domenii unde astfel de sisteme sunt des întâlnite, cum ar fi Machine Learning).
- Spre deosebire de eliminarea gaussiană și Householder, metoda gradientului conjugat nu oferă un mecanism de determinare a compatibilității/determinării unui sistem (se aplică exclusiv pe sisteme bine definite, compatibil determinate).

Cu toate acestea, metoda este extrem de eficientă pentru sisteme pozitiv-definite (complexitate $O(n^2)$, unde n este dimensiunea matricei sistemului), putând fi optimizată pentru matrice rare (se poate demonstra că în acest caz complexitatea este de $O(m\sqrt{k})$, unde m este numărul de elemente nenule, iar k este numărul de condiționare). De asemenea, metoda poate fi extinsă în anumite situații pentru matrice nesimetrice (gradienți biconjugați).

Metoda noastră se bazează pe ideea de **vectori conjugați** în raport cu matricea A. Doi vectori x și y se numesc conjugați dacă $x^T Ay = 0$. Principala idee a algoritmului este de a genera direcții conjugate, care pentru o matrice A inversabilă sunt liniar independente. În acest mod, soluția noastră este căutată într-un subspațiu de tip Krylov, adică:

$$x_{sol} \in span(b, A \cdot b, A^2 \cdot b \cdots A^{n-1} \cdot b) \tag{16}$$

Deoarece direcțiile de căutare conjugate acoperă tot spațiul $\mathbb{R}^n$, algoritmul nostru va determina soluția exactă în n pași. Cu toate acestea, algoritmul poate fi oprit mai devreme dacă aproximația soluției sistemului este suficient de bună.

1.3 Criteriile de evaluare pentru soluția propusă

Ne propunem să avem un număr de aproximativ 50 de teste generate cu ajutorul GNU Octave (limbaj care va fi folosit și pentru implementarea algoritmilor), distribuite astfel:

- 20 de teste pe sisteme pozitiv definite, dintre care 10 pe matrice foarte mari (peste 1000 de linii), tocmai pentru a arăta avantajul metodei gradientului comjugat în fața celorlalte metode. Aceste teste sunt pe sisteme bine definite (matricea sistemului este pătratică și se garantează că sistemul nostru are soluție).
- 30 de teste pe matrice normale, dintre care 20 teste pe sisteme de ecuații în care matricea sistemului este pătratică și inversabilă. Alte 10 teste vor fi pentru sisteme care nu au soluție unică sau care nu au soluție (aici sunt incluse și cazuri particulare, cum ar fi cazul în care matricea A este pătratică, dar singulară, precum și cazul în care matricea A nu este pătratică, iar sistemul este subdimensionat/supradimensionat).

Vom rula metodele exacte pe toate testele create, iar metoda gradientului conjugat va fi rulată doar pe cele 20 de teste (menționate anterior) din totalul de 50.

Verificarea de corectitudine și de precizie va fi făcută folosind operatorul existent în sintaxa limbajului Octave și care permite rezolvarea sistemelor de ecuații folosind cele mai eficiente metode, astfel:

$$A \cdot x = b \Longrightarrow x = A \backslash b \tag{17}$$

Astfel, putem compara soluția dată de metoda folosită cu soluția returnată prin intermediul acestui operator.

Principalele metrici pentru compararea algoritmilor de rezolvare a sistemelor de ecuații vor fi timpul de execuție (măsurabil în environment-ul din Octave) și corectitudinea răspunsului (eroarea în raport cu cea mai bună soluție; se poate folosi una din cele variante de calcul ale erorii de mai jos:)

$$\epsilon_1(x) = \|b - A \cdot x\| \tag{18}$$

$$\epsilon_2(x) = \|x - x_0\| \tag{19}$$

În formula de mai sus, x_0 reprezintă soluția sistemului returnată de operatorul \. Prima variantă de eroare este mai bună, deoarece reprezintă o metrică obiectivă ce indică corectitudinea soluției (spre deosebire de varianta 2, unde rezultatul nu este comparat cu cel real, ci cu cel aproximat de operator).

1.4 Referințe

– Multivariate linear regression in Machine Learning
– A simple approach to Conjugate Gradient
– Newton method for solving nonlinear equation systems
– Typical Gaussian elimination
– An interesting approach to Householder transformation