Introducere în calcul ştiinţific

Suport de curs şi laborator

pentru studenţii anului al II-lea

Ionel-Dumitrel Ghiba

Facultatea de Matematică, Universitatea "Alexandru Ioan Cuza" din Iaşi

dumitrel.ghiba@uaic.ro



ALEXANDRU IOAN CUZA
UNIVERSITY of IAŞI

Prefaţă

Calculul ştiinţific se află la intersecţia dintre matematică, algoritmi şi calculul efectiv pe calculator. El porneşte de la o întrebare simplă: *cum transformăm un model matematic într-un rezultat numeric în care putem avea încredere?* Răspunsul nu constă doar în aplicarea unei formule. Trebuie să alegem o reprezentare adecvată a problemei, un algoritm suficient de eficient, să înţelegem erorile introduse de aritmetica în virgulă mobilă şi să putem decide dacă rezultatul calculat este relevant pentru problema iniţială.

Acest curs este conceput ca o primă introducere în aceste idei. Algebra liniară numerică apare în mare parte a cursului, deoarece numeroase probleme de calcul ştiinţific se reduc, exact sau după discretizare, la operaţii cu vectori şi matrice: rezolvarea sistemelor liniare, aproximarea datelor, problema celor mai mici pătrate, calculul valorilor şi vectorilor proprii, optimizarea unor funcţii pătratice, aproximarea de rang mic prin descompunerea în valori singulare sau reducerea dimensiunii datelor prin analiza componentelor principale.

Scopul cursului nu este memorarea unei colecţii de algoritmi. Pentru fiecare metodă ne interesează patru întrebări: *de unde provine metoda?*, *de ce funcţionează?*, *cât costă?* şi *cât de mult putem avea încredere în rezultatul obţinut?* Aceste întrebări vor însoţi metodele prezentate pe parcursul cursului.

De la formulă la algoritm

În matematică, două formule echivalente descriu acelaşi obiect. Pe calculator, două proceduri algebric echivalente pot avea costuri şi comportamente numerice foarte diferite. De exemplu, pentru sistemul $Ax = b$, formula $x = A^{-1}b$ este corectă matematic, dar calcularea explicită a inversei este în general o alegere numerică mai slabă decât rezolvarea sistemului printr-o factorizare adecvată. În calculul ştiinţific contează, aşadar, nu numai formula, ci şi modul în care ea este transformată într-un algoritm.

Manualul este destinat studenţilor anului al II-lea şi presupune cunoştinţe de bază de algebră liniară şi analiză. Expunerea combină demonstraţia matematică cu interpretarea numerică, exemple complet calculate, exerciţii, sarcini de programare şi mini-proiecte.

Cum se citeşte o secţiune

La prima lectură urmăriţi ideea metodei şi rolul fiecărei ipoteze. La a doua lectură refaceţi demonstraţiile şi calculele. În final, implementaţi algoritmul şi testaţi-l pe exemple pentru care cunoaşteţi răspunsul. O metodă numerică este înţeleasă cu adevărat atunci când puteţi explica de ce funcţionează, când poate eşua, ce eroare produce şi cum verificaţi rezultatul.

Cum lucrăm la seminar şi laborator

În acest manual, calculul manual şi programarea nu sunt activităţi separate. O implementare este considerată înţeleasă numai dacă studentul poate explica formulele din care provine, ordinea operaţiilor, ipotezele în care algoritmul funcţionează şi un criteriu prin care rezultatul este verificat.

Pentru fiecare algoritm nou se recomandă aceeaşi succesiune. Mai întâi se execută cu pixul pe hârtie unul sau două cazuri mici, fără a sări etape. Apoi se scrie pseudocodul şi se identifică dependenţele dintre paşi. Numai după aceea se trece la MATLAB. Funcţiile MATLAB de bibliotecă sunt folosite în primul rând ca referinţă pentru verificare; scopul laboratorului este construirea algoritmului, nu înlocuirea lui printr-o singură comandă.

Ce înseamnă “a explica un algoritm”?

Studentul trebuie să poată răspunde la întrebări precum: de ce această buclă începe de la indicele $k+1$? de ce este necesar pivotul nenul? ce mărime scade de la o iteraţie la alta? de ce folosim QR şi nu ecuaţiile normale? ce se întâmplă dacă o valoare singulară este foarte mică? Un cod care produce un număr, dar ale cărui etape nu pot fi justificate, nu este încă o soluţie completă.

Proiectele au nivel mediu şi urmăresc integrarea noţiunilor. Nu se cere construirea unui sistem software complex. Se cere însă ca modelul, algoritmul, implementarea MATLAB, verificarea şi interpretarea rezultatului să formeze un argument coerent.

Bibliografie de bază şi materiale didactice consultate

Expunerea foloseşte rezultate standard din calcul ştiinţific şi algebră liniară numerică şi a fost calibrată didactic şi prin consultarea unor suporturi moderne de curs. Principalele repere sunt:

- Lucian Maticiuc, *Introducere în MATLAB*, Iaşi, 2019, material disponibil online la https://www.math.uaic.ro/~maticiuc/didactic/MATLAB_Course.pdf.
- A. Quarteroni, R. Sacco, F. Saleri, *Numerical Mathematics*, Springer.
- W. Gander, M. J. Gander, F. Kwok, *Scientific Computing – An Introduction Using Maple and MATLAB*, Springer.
- A. Beck, *Introduction to Nonlinear Optimization – Theory, Algorithms, and Applications with MATLAB*, SIAM.
- L. Page, S. Brin, R. Motwani, T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Stanford Digital Library Technologies Project, 1998/1999.
- Y. Koren, R. Bell, C. Volinsky, "Matrix Factorization Techniques for Recommender Systems", *Computer*, 42(8), 30–37, 2009.
- J. Calder, *Math 5467: Introduction to the Mathematics of Image and Data Analysis*, University of Minnesota, School of Mathematics, 2022.
- J. Gallier, J. Quaintance, *Linear Algebra for Computer Vision, Robotics, and Machine Learning*, University of Pennsylvania, 2025.
- O. Çeliker, *MIT 6.057: Introduction to MATLAB*, IAP 2019.

Organizarea şi evaluarea cursului

Evaluarea la această disciplină este pe parcurs şi este alcătuită din 2 componente de evaluare continuă. Evaluarea continuă constă în

- 70% verificarea înţelegerii rezolvărilor unor probleme la laborator, asemătoare cu cele din fişele propuse săptămânal. Această verificare se va face cerând la laborator ca studenţii să rezolve individual în 20 minute o sarcină de laborator. Aceste activităţi au caracter continuu şi vor fi notaţi, desigur, doar studenţii care le vor îndeplini. Nu se repetă, fiind considerate activităţi curente de laborator. Nu se repetă pe parcursul semestrului sau în sesiunea de restanţe!
- 30% verificarea cunoştinţelor teoretice printr-un test scurt de 20 minute în ultima săptămâna, la începutul ultimului curs. Se poate repeta în sesiunea de restanţe.

Ce se evaluează

- Accentul nu este pus numai pe obţinerea unui rezultat numeric. Studentul trebuie să poată justifica alegerea metodei, să explice ipotezele teoretice, să estimeze costul şi eroarea şi să interpreteze rezultatul unui experiment numeric.
- Nu se punctează prezenţa, însă absentarea de la orele de laborator vor conduce la imposibilitatea îndeplinirii sarcinilor de lucru, neîndeplinirea lor notându-se cu nota 1/10.
- Prezenţa la curs nu este obligatorie, însă multe explicaţii teoretice şi de îndeplinire a sarcinilor de laborator vor fi date doar la curs şi vor fi foarte utile, evaluarea insistând pe aceste aspecte comunicate doar în sala de curs.

Notaţii

Unele litere sunt folosite cu sensuri diferite în capitole diferite. Pentru a evita confuziile, convenţiile principale sunt reunite mai jos.

Simbol	Sens	Capitol
P	matrice de permutare	metode directe / pivotare
P	matrice de splitting sau precondiţionare	metode iterative
P_L	proiector ortogonal pe subspaţiul L	PCA
P	matrice stocastică de tranziţie	PageRank
U	factor superior triunghiular în LU	metode directe
U	matricea vectorilor singulari stângi	SVD
$r = b - Ax$	reziduul pentru sisteme liniare	convenţie folosită în manual

Sensul unui simbol este precizat din nou acolo unde poate apărea o confuzie.

Cuprins

Prefaţă **i**

Cum lucrăm la seminar şi laborator **ii**

Bibliografie de bază şi materiale didactice consultate **iii**

Organizarea şi evaluarea cursului **iv**

Notaţii **v**

1 Introducere în calcul ştiinţific **1**

1.1 Ce este calculul ştiinţific? . . . 1

1.2 De ce apar vectorii şi matricile în aplicaţii? . . . 1

1.3 Problema model: sistemul liniar . . . 4

1.4 Erori, stabilitate şi condiţionare . . . 4

1.5 Cum este organizat cursul? . . . 4

Test de capitol . . . 7

2 Recapitulare: matrice, valori proprii şi pozitiva definire **8**

Extensie didactică: interpretarea spectrală a matricelor . . . 11

Fişă de seminar: geometria matricelor simetrice . . . 13

Test de capitol . . . 18

3 Metode directe pentru sisteme liniare **19**

3.1 Rezolvarea sistemelor triunghiulare . . . 19

3.2 Eliminarea gaussiană şi factorizarea LU . . . 23

3.3 Forma compactă a factorizării LU . . . 33

3.4 Factorizarea LDM^T . . . 36

3.5 Pivotare . . . 40

3.6 Factorizarea Cholesky . . . 47

3.7 Matrice cu structură de bandă . . . 52

Fişă de seminar şi laborator: metode directe în ansamblu . . . 57

Test de capitol 62

4 Norme, condiţionare şi analiza erorii 63

Aritmetica în virgulă mobilă 63

Extensie didactică: norme, perturbaţii şi numărul de condiţionare 73

4.1 Îmbunătăţirea acurateţei eliminării gaussiene 76

Fişă de seminar: eroare, condiţionare şi stabilitate 80

Test de capitol 85

5 Metode iterative pentru sisteme liniare 86

Extensie didactică: metode iterative ca probleme de punct fix 89

5.1 Jacobi, Gauss-Seidel şi metode relaxate 92

5.2 Convergenţa metodelor Jacobi şi Gauss-Seidel 96

Fişă de seminar şi laborator: metode iterative 102

Test de capitol 107

6 Sisteme dreptunghiulare şi aproximaţii 108

Extensie didactică: ce înseamnă un sistem dreptunghiular? 108

6.1 "Soluţia" sistemelor supradeterminate 109

6.2 Problema celor mai mici pătrate şi ajustarea datelor 114

6.3 Ajustarea datelor (data fitting) 117

Fişă de seminar: sisteme dreptunghiulare şi modelarea datelor 122

Test de capitol 126

7 Regularizarea problemei celor mai mici pătrate şi eliminarea zgomotului 127

Extensie didactică: regularizarea ca informaţie a priori 130

Test de capitol 135

8 Factorizarea QR 136

Extensie didactică: ortogonalizarea ca instrument numeric 139

8.1 Sisteme dreptunghiulare rezolvate cu factorizarea QR 142

Fişă de seminar şi laborator: factorizarea QR 146

Test de capitol 150

9 Metode de descreştere şi metoda gradientului 151

Extensie didactică: un sistem SPD ca problemă de minimizare 153

9.1 Metoda direcţiilor de descreştere 155

9.2 Metoda gradientului 159

9.3 Numărul de condiţionare pentru funcţii pătratice. Rescalarea problemei. 163

9.4 Rescalare 166

Fişă de seminar: metode de descreştere şi gradient . . . 174
Extensie: metoda gradientului conjugat . . . 175
Test de capitol . . . 179

10 Metoda Gauss–Newton pentru cele mai mici pătrate neliniare 180
Extensie didactică: Gauss–Newton ca secvenţă de probleme LS liniare . . . 182
Fişă de seminar şi laborator: Gauss–Newton . . . 184
10.1 Problema celor mai mici pătrate pentru modele neliniare: ajustarea cercurilor şi problema localizării . . . 185
Test de capitol . . . 194

11 Descompunerea în valori singulare şi pseudoinversa 195
Extensie didactică: geometria şi puterea descompunerii SVD . . . 200
Fişă de seminar: SVD şi pseudoinversa . . . 202
Test de capitol . . . 206

12 Aplicaţii SVD în compresia imaginilor 207
12.1 Descompunerea în valori singulare (SVD) şi tensorii de ordinul al doilea . . . 207
12.2 Compresia imaginilor în tonuri de gri . . . 213
12.3 Compresia imaginilor color în spaţiul RGB . . . 217
12.4 Compresia imaginilor color în spaţiul YCbCr . . . 222
Test de capitol . . . 228

13 Analiza componentelor principale (PCA) 229
13.1 Motivaţie: cum descriem date de dimensiune mare prin puţine direcţii? . . . 229
13.2 Centrare, varianţă şi covarianţă . . . 229
13.3 Cel mai bun subspaţiu afin în sensul celor mai mici pătrate . . . 230
13.4 Principiul Rayleigh–Ritz şi alegerea direcţiilor principale . . . 232
13.5 PCA ca maximizare a varianţei . . . 234
13.6 Legătura exactă dintre PCA şi SVD . . . 234
13.7 Reducerea dimensiunii şi reconstrucţia . . . 235
13.8 Câte componente păstrăm? Varianţa explicată . . . 236
13.9 Exemplu complet calculat . . . 237
13.10 Centrare sau standardizare? . . . 238
13.11 Algoritmul PCA în MATLAB . . . 238
13.12 Aplicaţie: o colecţie de imagini ca nor de puncte . . . 240
13.13 Limitări şi situaţii în care PCA trebuie folosită cu atenţie . . . 241
13.14 Exerciţii . . . 242
13.15 Rezumatul capitolului . . . 243

Test de capitol 244

14 Aplicaţii integratoare: PageRank şi sisteme de recomandare 245

14.1 De ce introducem aceste aplicaţii abia acum? 245

14.2 PageRank ca problemă de calcul ştiinţific 245

14.2.1 De la un graf orientat la o matrice stocastică 245

14.2.2 Modelul surferului aleator şi teleportarea 246

14.2.3 Forma spectrală 247

14.2.4 Exemplu complet cu patru pagini 247

14.3 Un model de recomandare prin factorizare de rang mic 250

14.4 Ce noţiuni din curs au fost folosite? 252

Test de capitol 253

În loc de încheiere 254

Tabel de referinţă: costuri dominante 254

Fişă finală: compresia imaginilor ca proiect de calcul ştiinţific 255

Capitolul 1

Introducere în calcul ştiinţific

1.1 Ce este calculul ştiinţific?

Calculul ştiinţific studiază transformarea unui model matematic într-un calcul efectiv, controlabil şi reproductibil. Într-o problemă reală nu este suficient să ştim că o soluţie există sau că poate fi scrisă formal printr-o formulă. Trebuie să construim un algoritm, să estimăm costul lui, să înţelegem efectul erorilor de rotunjire şi al perturbaţiilor din date şi, în final, să putem verifica dacă rezultatul calculat este relevant pentru problema iniţială.

Un exemplu elementar este sistemul liniar $Ax = b$. Formula $x = A^{-1}b$ este corectă atunci când A este inversabilă, dar nu descrie metoda numerică pe care ar trebui să o folosim. În practică, alegerea depinde de structură: un sistem algebric având o matrice triunghiulară se rezolvă prin substituţie, un sistem algebric având o matrice generală prin eliminare Gauss şi factorizare LU cu pivotare, un sistem algebric având o matrice simetrică pozitiv definită (adică $A = A^T$ şi $x^T Ax > 0$ pentru orice $x \neq 0$) prin Cholesky, iar o problemă foarte mare cu o matrice rară – adică având majoritatea elementelor nule – poate favoriza o metodă iterativă. Aceeaşi problemă matematică poate conduce astfel la algoritmi foarte diferiţi.

Trei niveluri care nu trebuie confundate

În calculul ştiinţific avem permanent *problema matematică*, *algoritmul* şi *implementarea*. Problema poate fi bine sau prost condiţionată; algoritmul poate fi stabil sau instabil; implementarea poate fi corectă sau poate conţine erori. Un rezultat numeric bun cere înţelegerea tuturor celor trei niveluri menţionate.

1.2 De ce apar vectorii şi matricile în aplicaţii?

De exemplu, după discretizarea unei ecuaţii diferenţiale, valorile necunoscute ale unei temperaturi, deplasări sau presiuni devin componentele unui vector, iar relaţiile dintre ele sunt descrise de o matrice. În ajustarea datelor experimentale, liniile unei matrice conţin evaluările unui model în punctele măsurate. În procesarea imaginilor, o imagine în tonuri de gri este chiar o matrice de intensităţi, iar o imagine color poate fi privită ca un ansamblu de trei astfel de matrici. În probleme de reţea, matricile descriu conexiunile dintre noduri. În sisteme de recomandare, matricile pot codifica interacţiunile dintre utilizatori şi obiecte.

Pentru a vedea mai concret cum apar aceste matrice, să considerăm câteva situaţii simple. Dacă discretizăm problema de ordinul întâi

$$u'(x) = g(x), \qquad u(0) = 0,$$

pe o reţea uniformă $x_i = ih$, atunci aproximaţia cu diferenţă înainte

$$\frac{u_i - u_{i-1}}{h} \simeq g(x_i), \qquad i = 1, \ldots, n,$$

conduce, pentru necunoscutele $u_1, \ldots, u_n$, la sistemul

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ 0 & \cdots & -1 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = h \begin{pmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_n) \end{pmatrix}.$$

Astfel, matricea apare direct din relaţiile dintre valorile vecine ale soluţiei pe reţea.

În ajustarea unei drepte $y \simeq a+bt$ la datele (t_i, y_i), fiecare observaţie furnizează relaţia $a+bt_i \simeq y_i$. Reunind toate relaţiile obţinem

$$\begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} \simeq \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix},$$

deci matricea apare direct prin evaluarea funcţiilor 1 şi t în punctele măsurate.

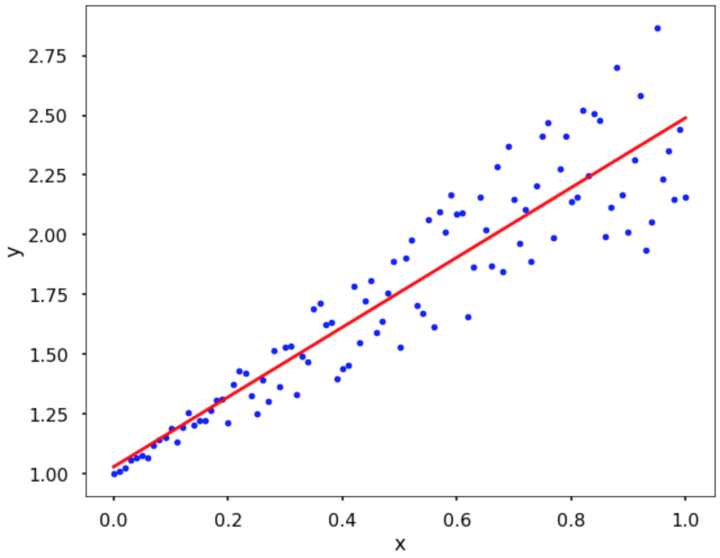


Figura 1.1: Date experimentale şi o dreaptă de ajustare. Fiecare observaţie contribuie cu o relaţie liniară aproximativă pentru parametrii dreptei.

Într-o reţea cu n noduri, o descriere elementară este matricea de adiacenţă $A = (a_{ij})$, unde $a_{ij} = 1$ dacă nodurile i şi j sunt conectate şi $a_{ij} = 0$ în caz contrar. Astfel, chiar structura conexiunilor este codificată într-o matrice.

Aceste exemple au o caracteristică comună: dimensiunea problemei poate fi mare, datele pot conţine zgomot, iar structura matricei este adesea la fel de importantă ca valorile numerice ale elementelor sale. De aceea, cursul nu tratează algebra liniară doar ca pe un set de identităţi, ci ca pe un limbaj pentru construirea şi analiza algoritmilor.

Un exemplu practic important: o imagine ca matrice

O imagine în tonuri de gri cu $m \times n$ pixeli poate fi reprezentată printr-o matrice $A \in \mathbb{R}^{m \times n}$. Elementul a_{ij} descrie intensitatea pixelului din poziţia (i, j): valori mici corespund unor tonuri mai închise, iar valori mari unor tonuri mai deschise. Dacă această matrice admite o aproximare bună de rang mic, ea poate fi descrisă cu mult mai puţine date. Mai târziu vom demonstra că descompunerea în valori singulare furnizează, într-un sens precis, cea mai bună aproximaţie de rang fix. Pentru moment este suficient să observăm ordinul de mărime: o imagine 4000×3000 conţine $12\,000\,000$ de valori, în timp ce o reprezentare de rang 50 necesită aproximativ $50(4000 + 3000 + 1)$ valori, adică aproximativ $350\,000$.

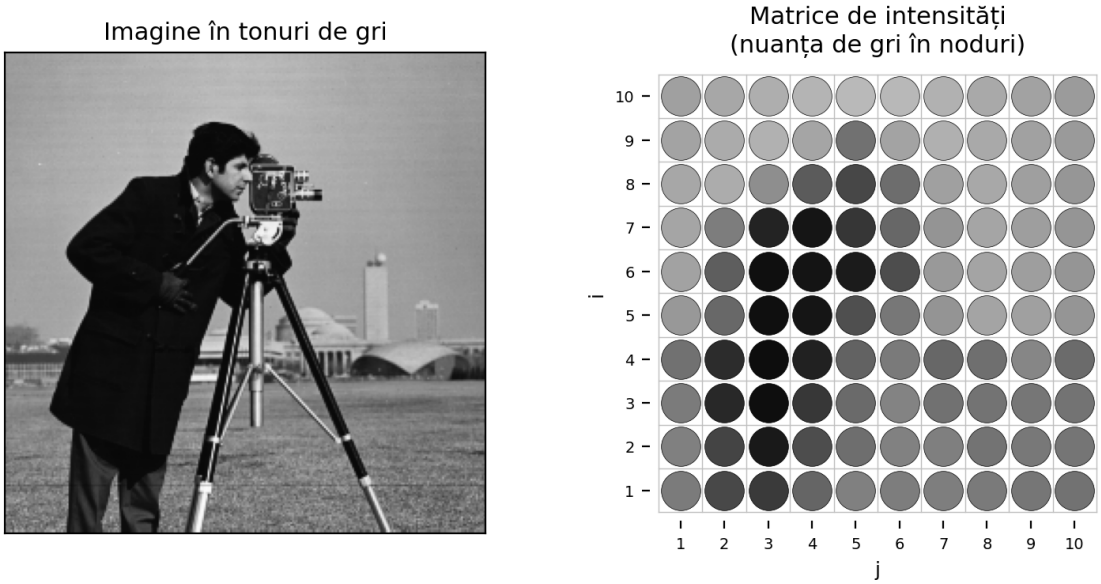


Figura 1.2: O imagine alb-negru şi o reprezentare matricială grosieră. Fiecărui nod din grilă îi corespunde o nuanţă de gri, deci o valoare a matricei de intensităţi.

Acest exemplu arată de ce noţiuni aparent abstracte precum rangul, ortogonalitatea şi valorile singulare au consecinţe practice directe. În capitolele finale vom reveni la imagini şi vom construi efectiv algoritmi MATLAB de compresie.

Alte aplicaţii care vor reapărea în curs

Problema celor mai mici pătrate apare în calibrare, ajustarea unor modele la date şi identificarea parametrilor. Regularizarea permite stabilizarea problemelor sensibile la zgomot. Metodele de gradient şi Gauss–Newton apar în optimizare şi ajustări neliniare. Matricile rare şi metodele iterative apar în discretizări ale ecuaţiilor diferenţiale şi în probleme de reţea de dimensiuni foarte mari.

Spre sfârşitul manualului vom reveni la mai multe dintre aceste noţiuni prin algoritmul PageRank pentru analiza reţelelor. El va fi introdus numai după ce vom avea la dispoziţie normele induse, raza spectrală, metodele iterative, analiza convergenţei şi reprezentarea rară a matricelor. În acel moment vom putea nu doar să scriem o iteraţie, ci să demonstrăm de ce converge, să explicăm de ce este eficientă şi să o implementăm corect în MATLAB. În aceeaşi parte finală vom discuta şi un model simplificat de recomandare bazat pe factorizări de rang mic şi pe problema celor mai mici pătrate.

1.3 Problema model: sistemul liniar

Unul dintre ideile centrale ale cursului este sistemul $Ax = b$, unde $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$ şi $b \in \mathbb{R}^m$. În cazul pătratic, $m = n$, iar dacă A este inversabilă, soluţia este unică. Din punct de vedere teoretic putem scrie $x = A^{-1}b$ sau putem invoca regula lui Cramer. Din punct de vedere numeric, întrebarea relevantă este însă cum obţinem x cu un cost rezonabil şi cu o eroare controlată.

Regula lui Cramer ilustrează bine diferenţa dintre formulă şi algoritm. Calcularea determinanţilor prin dezvoltare Laplace are un cost care creşte factorial cu dimensiunea. Eliminarea Gauss şi factorizarea LU au, pentru matrice dense, cost de ordin n^3 şi permit apoi rezolvarea eficientă pentru mai mulţi termeni liberi. Pentru matrici rare sau structurate costul poate fi redus substanţial.

Direct sau iterativ?

Metodele directe, precum LU, Cholesky şi QR, produc în aritmetică exactă rezultatul după un număr finit de paşi. Metodele iterative construiesc o succesiune $x^{(0)}, x^{(1)}, \ldots$ care trebuie să tindă la soluţie şi sunt potrivite pentru multe probleme foarte mari şi rare. Alegerea nu se face numai după dimensiunea matricei: contează simetria, pozitiva definire, raritatea, condiţionarea şi numărul de sisteme care trebuie rezolvate cu aceeaşi matrice.

1.4 Erori, stabilitate şi condiţionare

Calculatorul nu efectuează operaţii cu numere reale exacte. Numerele sunt reprezentate cu precizie finită, iar fiecare operaţie poate introduce o eroare de rotunjire. În plus, datele unei probleme provin adesea din măsurători şi au propriile erori. Din acest motiv vom distinge două întrebări.

Prima este o proprietate a problemei: *cât de mult se modifică soluţia dacă datele se modifică puţin?* Aceasta este problema condiţionării. A doua este o proprietate a algoritmului: *cât de mult amplifică procedura numerică erorile inevitabile ale calculului?* Aceasta este problema stabilităţii. Un algoritm foarte bun nu poate transforma o problemă extrem de prost condiţionată într-una insensibilă, dar un algoritm instabil poate degrada inutil chiar şi o problemă bine condiţionată.

1.5 Cum este organizat cursul?

Ordinea capitolelor urmăreşte dependenţa logică dintre noţiuni. Începem cu recapitularea matricelor simetrice, a valorilor proprii şi a pozitivei definiri. Continuăm cu metode directe pentru sisteme liniare: substituţie, eliminare Gauss, pivotare, factorizările LU, LDL^T şi Cholesky, precum şi matricile cu structură de bandă. Abia apoi introducem normele, raza spectrală, condiţionarea şi analiza erorilor.

Pe această bază studiem metodele iterative Jacobi, Gauss–Seidel, JOR şi SOR şi demonstrăm criterii de convergenţă. Urmează sistemele dreptunghiulare, problema celor mai mici pătrate, regularizarea şi factorizarea QR. Problemele liniare sunt apoi reinterpretate ca probleme de optimizare, ceea ce conduce la metode de descreştere, metoda gradientului şi metoda Gauss–Newton. Ultimele capitole dezvoltă descompunerea în valori singulare, pseudoinversa şi aproximaţiile de rang mic, cu aplicaţii în compresia imaginilor. Pe această bază introducem analiza componentelor principale (PCA), unde aceleaşi instrumente spectrale sunt folosite pentru

reducerea dimensiunii, aproximarea datelor prin subspaţii afine şi analiza variaţiei. Numai după parcurgerea acestor instrumente revenim la aplicaţii integratoare de tipul analizei reţelelor şi al recomandării.

Ce trebuie să poată face studentul

La finalul cursului, studentul trebuie să poată trece în ambele direcţii între matematică şi algoritm: să deducă paşii unei metode din relaţiile matematice, dar şi să explice matematic un algoritm scris în MATLAB. La problemele mici, paşii trebuie să poată fi executaţi cu pixul pe hârtie; la problemele mai mari, implementarea trebuie însoţită de criterii de oprire, verificări prin reziduu, estimări de eroare şi o discuţie privind costul şi stabilitatea.

Întrebări de început

1. De ce formula $x = A^{-1}b$ nu specifică încă un algoritm numeric?
2. Daţi un exemplu în care structura matricei ar trebui să schimbe succesiunea de calcule folosită pentru rezolvarea sistemului.
3. De ce trebuie să distingem între problema matematică, algoritm şi implementare?
4. De ce o matrice rară nu ar trebui, în general, stocată ca şi cum toate elementele sale ar fi nenule?
5. Ce informaţii despre date, dimensiune şi structură ar trebui notate înainte de a începe implementarea MATLAB?

Sarcini de aprofundare şi orientare

Probleme de început: model, date şi calcul

1. Alegeţi una dintre aplicaţiile discutate în capitol (procesare de imagini, recomandări, analiza reţelelor, ajustarea datelor). Identificaţi separat: datele de intrare, necunoscutele, rezultatul urmărit şi etapa în care apare calculul numeric.
2. Pentru un sistem liniar simplu $Ax = b$, explicaţi diferenţa dintre a scrie o formulă matematică pentru soluţie şi a descrie o succesiune efectivă de operaţii care o calculează.
3. Daţi câte un exemplu de matrice diagonală, triunghiulară, simetrică şi rară. Pentru fiecare, indicaţi ce informaţie structurală este vizibilă înainte de efectuarea calculelor.
4. Comparaţi stocarea unei matrice 1000×1000 dense cu stocarea unei matrice de aceeaşi dimensiune având numai trei diagonale nenule. Număraţi elementele care trebuie memorate în fiecare caz.
5. Construiţi un exemplu în care aceeaşi problemă poate fi formulată cu 10, 100 şi 1000 de necunoscute. Discutaţi ce se schimbă în volumul de calcul şi ce rămâne neschimbat în model.
6. Formulaţi, în maximum cinci rânduri, diferenţa dintre model matematic, date, algoritm şi implementare.

Laborator de orientare: aceeaşi structură la dimensiuni diferite

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă

este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Observaţi rolul structurii şi al dimensiunii înainte de introducerea instrumentelor de analiză a erorilor.

1. Construiţi în MATLAB matricea tridiagonală cu 2 pe diagonală şi -1 pe diagonalele vecine, pentru $n = 10, 100, 1000$.
2. Stocaţi matricea atât dens, cât şi cu `sparse` şi comparaţi memoria raportată de `whos`.
3. Reprezentaţi structura cu `spy(A)` şi explicaţi ce informaţie oferă imaginea.
4. Alegeţi vectorul $x_\star = \mathbf{1}$, formaţi $b = Ax_\star$ şi folosiţi `A\b` pentru a obţine un vector calculat $\widehat{x}$.
5. Comparaţi direct componentele lui $\widehat{x}$ cu cele ale lui $x_\star$ şi raportaţi cea mai mare diferenţă absolută dintre componente, fără a introduce încă noţiuni de normă sau condiţionare.
6. Măsuraţi timpul pentru forma densă şi forma rară şi explicaţi diferenţele observate.

Test de capitol

Test – Introducere în calcul ştiinţific

1. Analiză şi justificare. Pentru fiecare dintre situaţiile următoare precizaţi dacă afirmaţia se referă în primul rând la *problema matematică*, la *algoritm* sau la *implementare* şi justificaţi răspunsul:

a) o mică modificare a datelor produce o modificare mare a soluţiei;

b) două programe care implementează aceeaşi metodă furnizează rezultate diferite deoarece unul conţine o eroare de indexare;

c) pentru aceeaşi problemă, o metodă exploatează faptul că matricea este triunghiulară, iar alta tratează matricea ca fiind generală.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi.

a) Pentru un sistem $Ax = b$, formula $x = A^{-1}b$ este:

A) întotdeauna un algoritm numeric complet;

B) o formulă matematică ce nu precizează în sine succesiunea efectivă de calcule;

C) o formulă care exploatează automat faptul că A poate fi rară sau structurată;

D) aplicabilă şi dacă A nu este inversabilă.

b) O implementare numerică serioasă trebuie să ţină seama de:

A) structura datelor;

B) dimensiunea problemei;

C) verificarea rezultatului;

D) numai numărul de linii ale programului.

Capitolul 2

Recapitulare: matrice, valori proprii şi pozitiva definire

Matrice simetrice pozitiv definite

Pozitiva definire a unei matrice

- Spunem că matricea $A \in \mathbb{R}^{n\times n}$ este simetrică dacă $A = A^T$,
- Spunem că matricea simetrică $A \in \mathbb{R}^{n\times n}$ este **pozitiv semidefinită**, notăm $A \succeq 0$, dacă $\langle A\,x, x\rangle \geq 0$ pentru orice $x \in \mathbb{R}^n$, unde $\langle \cdot, \cdot \rangle$ reprezintă produsul scalar standard din $\mathbb{R}^n$.
- Spunem că matricea simetrică $A \in \mathbb{R}^{n\times n}$ este **pozitiv definită**, notăm $A \succ 0$, dacă $\langle A\,x, x\rangle > 0$ pentru orice $x \in \mathbb{R}^n$, $x \neq 0$.

Valori proprii şi vectori proprii

Fie $A \in \mathbb{R}^{n\times n}$. Un vector nenul $v \in \mathbb{C}^n$ se numeşte vector propriu pentru A dacă există $\lambda \in \mathbb{C}$ astfel încât

$$A\,v = \lambda\,v.$$

Scalarul λ se numeşte valoarea proprie corespunzătoare vectorului propriu v. În general, o matrice reală poate avea valori proprii complexe. Pentru o matrice simetrică reală, toate valorile proprii sunt însă reale.

Teoremă 2.0.1. *Dacă $A \in \mathbb{R}^{n\times n}$ este simetrică şi $Av = \lambda v$ pentru un vector $v \in \mathbb{C}^n \setminus \{0\}$, atunci $\lambda \in \mathbb{R}$.*

Demonstraţie. Pe $\mathbb{C}^n$ folosim produsul scalar hermitian standard

$$\langle z, w\rangle_{\mathbb{C}^n} = z^H w = \sum_{j=1}^{n} \overline{z_j}\, w_j,$$

unde $z^H = \overline{z}^T$ notează transpusa conjugată (sau adjuncta hermitiană) a vectorului z. Mai general, pentru o matrice complexă M, notaţia $M^H = \overline{M}^T$ reprezintă transpusa conjugată a lui M. Deoarece A este reală şi simetrică, avem $A^H = A$.

Din $Av = \lambda v$ rezultă

$$v^H A v = \lambda v^H v.$$

Scalarul $v^H Av$ este real: într-adevăr,

$$\overline{v^H Av} = v^H A^H v = v^H Av.$$

Cum $v^H v = \|v\|_2^2 > 0$, obţinem

$$\lambda = \frac{v^H Av}{v^H v} \in \mathbb{R}.$$

□

Teoremă 2.0.2 (Teorema spectrală pentru matrice reale simetrice)**.** *Fie $A \in \mathbb{R}^{n\times n}$ simetrică. Atunci există o matrice ortogonală $Q \in \mathbb{R}^{n\times n}$ şi o matrice diagonală reală $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ astfel încât*

$$A = Q\Lambda Q^T.$$

Echivalent, $\mathbb{R}^n$ admite o bază ortonormală formată din vectori proprii ai lui A.

Demonstraţie. Demonstrăm prin inducţie după n. Pentru $n = 1$ afirmaţia este evidentă. Presupunem rezultatul adevărat în dimensiunea $n-1$. Polinomul caracteristic al lui A are o rădăcină în $\mathbb{C}$, deci A are un vector propriu complex $v \neq 0$. Teorema precedentă arată că valoarea proprie corespunzătoare λ este reală. Scriind $v = x + iy$, cu $x, y \in \mathbb{R}^n$, din $Av = \lambda v$ obţinem $Ax = \lambda x$ şi $Ay = \lambda y$; cel puţin unul dintre x, y este nenul. Există deci un vector propriu real unitar q_1.

Considerăm subspaţiul ortogonal $q_1^\perp$. Acesta este invariant la acţiunea lui A: dacă $z \perp q_1$, atunci, folosind simetria,

$$q_1^T Az = (Aq_1)^T z = (\lambda q_1)^T z = 0,$$

deci $Az \in q_1^\perp$. Restricţia lui A la $q_1^\perp$ este din nou simetrică. Prin ipoteza de inducţie există o bază ortonormală $q_2, \ldots, q_n$ a lui $q_1^\perp$ formată din vectori proprii ai restricţiei, deci şi ai lui A. Matricea $Q = (q_1 | \cdots | q_n)$ este ortogonală, iar $Q^T AQ$ este diagonală cu valorile proprii pe diagonală. Rezultă $A = Q\Lambda Q^T$. □

Valorile proprii ale unei matrice simetrice $A \in \mathbb{R}^{n\times n}$ vor fi notate cu

$$\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_n(A).$$

Cea mai mare valoare proprie va fi notată cu $\lambda_{\max}(A) = \lambda_1(A)$ şi cea mai mică valoare proprie va fi notată cu $\lambda_{\min}(A) = \lambda_n(A)$.

Teoremă 2.0.3 (Pozitiva definire şi valorile proprii)**.** *Fie $A \in \mathbb{R}^{n\times n}$ simetrică. Atunci A este pozitiv semidefinită dacă şi numai dacă toate valorile sale proprii sunt nenegative, iar A este pozitiv definită dacă şi numai dacă toate valorile sale proprii sunt strict pozitive.*

Demonstraţie. Prin teorema spectrală, $A = Q\Lambda Q^T$, cu Q ortogonală şi $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Pentru $x \in \mathbb{R}^n$ punem $y = Q^T x$. Deoarece Q este ortogonală, $x \neq 0$ dacă şi numai dacă $y \neq 0$, iar

$$x^T Ax = y^T \Lambda y = \sum_{i=1}^n \lambda_i y_i^2.$$

Dacă toate $\lambda_i \geq 0$, suma este nenegativă pentru orice x, deci A este pozitiv semidefinită; dacă toate $\lambda_i > 0$, suma este strict pozitivă pentru orice $x \neq 0$, deci A este pozitiv definită. Reciproc, alegând $x = q_i$, coloana a i-a a lui Q, obţinem $x^T Ax = \lambda_i$. Prin urmare pozitiva semidefinire implică $\lambda_i \geq 0$, iar pozitiva definire implică $\lambda_i > 0$ pentru orice i. □

Teoremă 2.0.4 (Criteriul lui Sylvester)**.** *Fie $A \in \mathbb{R}^{n\times n}$ simetrică şi notaţi $A_k = A(1:k, 1:k)$. Atunci A este pozitiv definită dacă şi numai dacă*

$$\det A_k > 0, \qquad k = 1, \ldots, n.$$

Demonstraţie. Dacă A este pozitiv definită, atunci fiecare submatrice principală A_k este pozitiv definită: pentru $z \in \mathbb{R}^k \setminus \{0\}$ completăm z cu zerouri până la un vector $x \in \mathbb{R}^n$ şi avem $z^T A_k z = x^T A x > 0$. Valorile proprii ale lui A_k sunt deci pozitive, iar $\det A_k$, produsul lor, este pozitiv.

Pentru reciproca folosim inducţia după n. Cazul $n = 1$ este imediat. Presupunem că toţi minorii principali conducători ai lui A sunt pozitivi. Atunci, pentru blocul $B = A_{n-1}$, toţi minorii principali conducători sunt pozitivi; prin ipoteza de inducţie, B este pozitiv definită şi, în particular, inversabilă. Scriem

$$A = \begin{pmatrix} B & c \\ c^T & a \end{pmatrix}.$$

Cantitatea

$$s = a - c^T B^{-1} c$$

se numeşte complementul Schur al blocului B în A. Pentru a vedea de unde apare, efectuăm o eliminare pe blocuri:

$$\begin{pmatrix} I & 0 \\ -c^T B^{-1} & 1 \end{pmatrix} \begin{pmatrix} B & c \\ c^T & a \end{pmatrix} = \begin{pmatrix} B & c \\ 0 & s \end{pmatrix}.$$

Matricea din stânga cu care am înmulţit are determinantul 1, iar matricea din dreapta este triunghiulară pe blocuri. Prin urmare,

$$\det A = \det B \, s = \det B \left(a - c^T B^{-1} c\right).$$

Din ipoteză, $\det A > 0$ şi $\det B > 0$, deci

$$s = a - c^T B^{-1} c > 0.$$

Rămâne să arătăm că această informaţie implică pozitiva definire a lui A. Pentru $x = (y, t)^T \in \mathbb{R}^n$ avem

$$x^T A x = y^T B y + 2t\, c^T y + a t^2.$$

Completând pătratul şi folosind simetria lui B, obţinem

$$\begin{aligned}(y + tB^{-1}c)^T B (y + tB^{-1}c) + s\, t^2 &= y^T B y + 2t\, c^T y + t^2 c^T B^{-1} c \\ &\quad + t^2\left(a - c^T B^{-1} c\right) \\ &= x^T A x.\end{aligned}$$

Deoarece B este pozitiv definită, primul termen este nenegativ, iar $s > 0$. Dacă $t \neq 0$, termenul st^2 este strict pozitiv. Dacă $t = 0$ şi $x \neq 0$, atunci $y \neq 0$, iar $y^T B y > 0$. Aşadar

$$x^T A x > 0 \qquad \text{pentru orice } x \neq 0,$$

deci A este pozitiv definită. □

O matrice $A \in \mathbb{R}^{n\times n}$ se numeşte diagonal dominantă pe linii dacă

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad \text{pentru} \quad i = 1, \cdots, n. \tag{2.1}$$

O matrice $A \in \mathbb{R}^{n\times n}$ se numeşte diagonal dominantă pe coloane dacă

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ji}|, \quad \text{pentru} \quad i = 1, \cdots, n, \tag{2.2}$$

Dacă inegalităţile sunt stricte, spunem că A este strict diagonal dominantă (pe linii, respectiv, pe coloane).

Teoremă 2.0.5. *O matrice simetrică strict diagonal dominantă cu elemente strict pozitive pe diagonală este pozitiv definită.*

Demonstraţie. Fie $A = (a_{ij})$ simetrică, cu $a_{ii} > \sum_{j\neq i} |a_{ij}|$ pentru fiecare i. Pentru $x \in \mathbb{R}^n$ avem, prin simetrie,

$$x^T Ax = \sum_{i=1}^{n} a_{ii}x_i^2 + 2 \sum_{1\leq i<j\leq n} a_{ij}x_i x_j.$$

Folosind $2|x_i x_j| \leq x_i^2 + x_j^2$, obţinem

$$2a_{ij}x_i x_j \geq -2|a_{ij}||x_i x_j| \geq -|a_{ij}|(x_i^2 + x_j^2).$$

Prin însumare,

$$x^T Ax \geq \sum_{i=1}^{n} \left(a_{ii} - \sum_{j\neq i} |a_{ij}| \right) x_i^2.$$

Fiecare coeficient dintre paranteze este strict pozitiv. Pentru $x \neq 0$ există cel puţin un $x_i \neq 0$, deci partea dreaptă este strict pozitivă. Prin urmare A este pozitiv definită. □

Reciproca nu este valabilă. De exemplu,

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix}$$

este simetrică şi pozitiv definită, deoarece minorii principali conducători sunt $1 > 0$ şi $\det A = 1 > 0$, dar prima linie nu este diagonal dominantă: $|a_{11}| = 1 < 2 = |a_{12}|$. Aşadar, diagonal-dominanţa strictă cu diagonală pozitivă este o condiţie suficientă, nu necesară, pentru pozitiva definire.

Extensie didactică: interpretarea spectrală a matricelor

Pentru metodele numerice, valorile proprii nu sunt doar obiecte algebrice. Ele descriu direcţii în care acţiunea unei matrice are o formă simplă şi vor reapărea în capitolele următoare în analiza mai multor algoritmi.

Matrice simetrică pozitiv definită

Pentru o matrice simetrică A, condiţia $x^T Ax > 0$ pentru orice $x \neq 0$ înseamnă că forma pătratică asociată are curbură pozitivă în orice direcţie. Prin teorema spectrală, $A = Q\Lambda Q^T$, unde Q este ortogonală şi $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Dacă scriem $y = Q^T x$, atunci $x^T Ax = \sum_i \lambda_i y_i^2$. De aici se vede imediat echivalenţa dintre pozitiva definire şi condiţia $\lambda_i > 0$ pentru toate valorile proprii.

Raportul Rayleigh $R_A(x) = \frac{x^T A x}{x^T x}$, definit pentru $x \neq 0$, verifică $\lambda_{\min}(A) \leq R_A(x) \leq \lambda_{\max}(A)$ pentru orice matrice simetrică. El permite astfel să legăm proprietăţile geometrice ale formei pătratice de spectrul matricei. În cazul unei funcţii pătratice $f(x) = \frac{1}{2}x^T A x - b^T x$, aceste două valori proprii extreme vor determina cât de alungite sunt curbele de nivel şi, ulterior, cât de repede converge metoda gradientului.

Exemplu ghidat

Fie $A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$. Vectorii $q_1 = (1,1)^T/\sqrt{2}$ şi $q_2 = (1,-1)^T/\sqrt{2}$ sunt vectori proprii pentru valorile proprii 3, respectiv 1. Pentru $x = \alpha q_1 + \beta q_2$ avem $x^T A x = 3\alpha^2 + \beta^2$. Pozitivitatea este transparentă în aceste coordonate. În plus, raportul dintre cea mai mare şi cea mai mică valoare proprie este 3; pentru moment îl interpretăm numai ca pe o măsură geometrică a diferenţei dintre cele două axe ale elipsei.

De ce contează ortogonalitatea?

Transformările ortogonale păstrează produsul scalar şi lungimile euclidiene. Această proprietate geometrică va fi exploatată ulterior de factorizări numerice bazate pe transformări ortogonale; aici ne limităm la interpretarea ei algebrică şi geometrică.

Laborator: spectru şi formă pătratică

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Alegeţi o matrice simetrică 2×2 pozitiv definită. Calculaţi valorile proprii cu `eig`, apoi reprezentaţi curbele de nivel ale lui $f(x) = \frac{1}{2}x^T A x$. Rotiţi sistemul de coordonate cu matricea vectorilor proprii şi verificaţi că în noile coordonate curbele de nivel sunt elipse cu axele aliniate cu axele coordonatelor. Modificaţi raportul $\lambda_{\max}/\lambda_{\min}$ şi observaţi cum se alungesc elipsele.

Exerciţii şi verificarea înţelegerii

1. Demonstraţi că o matrice simetrică A este pozitiv definită dacă şi numai dacă toate valorile sale proprii sunt strict pozitive.
2. Pentru $A = \begin{pmatrix} 4 & -1 \\ -1 & 1 \end{pmatrix}$ verificaţi pozitiva definire prin două metode: criteriul minorilor principali şi calculul valorilor proprii.
3. Arătaţi că, pentru $A = A^T$, maximul raportului Rayleigh este $\lambda_{\max}(A)$ şi minimul este $\lambda_{\min}(A)$.
4. Construiţi o matrice nesimetrică ale cărei valori proprii sunt pozitive, dar pentru care $x^T A x$ nu este pozitiv pentru orice $x \neq 0$. Explicaţi de ce ipoteza de simetrie este necesară în afirmaţia anterioară.

Mini-proiect: un model elementar cu arcuri

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Considerăm n puncte materiale care se pot deplasa numai pe o dreaptă, cu deplasări $x_1, \dots, x_n$, legate succesiv prin arcuri identice. Pentru un arc, presupunem legea liniară $F = -k\delta$: forţa este opusă alungirii δ şi proporţională cu ea. Capetele lanţului sunt fixate. Scrierea echilibrului forţelor pentru fiecare punct conduce la un sistem $Kx = f$. În acest context, numim *matrice de rigiditate* matricea K ce leagă deplasările de forţele elastice. După împărţirea prin k, ea este tridiagonală, cu 2 pe diagonală şi -1 pe diagonalele vecine. Deduceţi această matrice din ecuaţiile de echilibru, apoi studiaţi numeric valorile proprii pentru mai multe valori ale lui n şi verificaţi pozitiva definire. Interpretarea fizică a vectorilor proprii este opţională; din punct de vedere matematic, urmăriţi forma lor şi variaţia valorilor proprii.

Rezolvare orientativă

Pozitiva definire prin două metode

Pentru $A = \begin{pmatrix} 4 & -1 \\ -1 & 1 \end{pmatrix}$, criteriul minorilor principali dă $\Delta_1 = 4 > 0$ şi $\Delta_2 = 4 - 1 = 3 > 0$, deci A este SPD. Valorile proprii sunt rădăcinile lui $\lambda^2 - 5\lambda + 3 = 0$, adică $\lambda_{1,2} = (5 \pm \sqrt{13})/2$, ambele pozitive. Cele două verificări exprimă aceeaşi proprietate, dar din perspective diferite. Pentru $x = (x_1, x_2)^T$, forma pătratică este $4x_1^2 - 2x_1x_2 + x_2^2$. Completând pătratul, obţinem $3x_1^2 + (x_2 - x_1)^2 > 0$ pentru $x \neq 0$, o a treia demonstraţie. În aplicaţii, alegerea demonstraţiei depinde de informaţia disponibilă: structură explicită, spectru sau factorizare.

Situaţie practică: energie elastică discretă

În modelul elementar cu arcuri descris mai sus, x_i reprezintă deplasările punctelor, iar f_i forţele exterioare. Matricea K care apare în relaţia de echilibru $Kx = f$ a fost numită matrice de rigiditate. Pentru trei deplasări interioare şi arcuri identice (după o scalare cu constanta arcului), energia poate avea forma $E(x) = \frac{1}{2}x^TKx - f^Tx$, unde K este tridiagonală cu 2 pe diagonală şi -1 pe vecini. Termenul x^TKx se poate rescrie ca sumă de pătrate ale diferenţelor dintre deplasări şi ale deplasărilor faţă de capetele fixate. De aici se vede direct că K este SPD. Condiţia de staţionaritate a acestei funcţii pătratice conduce la $Kx = f$. Deoarece K este pozitiv definită, sistemul are soluţie unică. Cholesky şi, ulterior, metoda gradientului vor exploata exact această structură.

Fişă de seminar: geometria matricelor simetrice

Problema rezolvată 1: pozitiva definire prin trei criterii

Considerăm $A = \begin{pmatrix} 4 & 2 \\ 2 & 3 \end{pmatrix}$. Putem verifica pozitiva definire în trei moduri. Prin criteriul Sylvester, minorii principali sunt $4 > 0$ şi $\det A = 8 > 0$. Prin completare de pătrate, $x^T Ax = 4x_1^2 + 4x_1x_2 + 3x_2^2 = (2x_1 + x_2)^2 + 2x_2^2 > 0$ pentru $x \neq 0$. Prin valori proprii, polinomul caracteristic este $\lambda^2 - 7\lambda + 8$, ale cărui rădăcini $(7 \pm \sqrt{17})/2$ sunt pozitive. Cele trei teste exprimă aceeaşi proprietate, dar sunt utile în contexte diferite.

Pentru calcul numeric, nu vom calcula de obicei valorile proprii doar pentru a decide dacă o matrice SPD admite Cholesky; algoritmul Cholesky însuşi poate servi drept test practic. În demonstraţii, însă, perspectiva spectrală explică direct geometria formei pătratice.

Problema rezolvată 2: o schimbare ortogonală de coordonate

Dacă $Q^TQ = I$, atunci produsul scalar este păstrat: $(Qx)^T(Qy) = x^Ty$. În particular, unghiurile şi lungimile euclidiene nu se schimbă. Această observaţie explică de ce diagonalizarea ortogonală a unei matrice simetrice păstrează fidel geometria problemei.

Laborator integrator 2: spectru şi curbe de nivel

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Generaţi matrici simetrice 2×2 cu valori proprii prescrise şi vectori proprii rotiţi. Reprezentaţi curbele $x^TAx = 1$. Modificaţi raportul valorilor proprii de la 1 la 10^4 şi observaţi elipsele. Conectaţi forma geometrică cu raportul dintre valorile proprii extreme.

Probleme pentru seminar

1. Arătaţi că o matrice SPD este nesingulară.
2. Demonstraţi că suma a două matrici SPD este SPD.
3. Dacă A este SPD şi Q ortogonală, arătaţi că Q^TAQ este SPD şi are aceleaşi valori proprii.
4. Construiţi o matrice simetrică indefinită cu determinant pozitiv. De ce determinantul singur nu decide pozitiva definire în dimensiune mai mare?
5. Pentru $A = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$ determinaţi intervalul lui ρ pentru care A este SPD.
6. Calculaţi valorile proprii în problema precedentă şi studiaţi ce se întâmplă cu cea mai mică valoare proprie când $|\rho| \to 1$.
7. Pentru o matrice simetrică diagonalizată ortogonal, explicaţi cum se citeşte acţiunea matricei pe fiecare direcţie proprie.
8. Explicaţi de ce o matrice Gram $G = X^TX$ poate fi singulară chiar dacă toate elementele diagonale sunt pozitive.

Mini-proiect: atlas de matrici

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi un script MATLAB cu exemple pentru cel puţin opt clase: diagonală, ortogonală, SPD, semidefinită, indefinită, strict diagonal dominantă, singulară şi aproape singulară. Pentru fiecare afişaţi determinantul, rangul şi valorile proprii şi verificaţi proprietăţile structurale relevante folosind numai criteriile introduse în acest capitol.

Sarcini de aprofundare şi sinteză

Probleme de aprofundare: spectru, forme pătratice şi geometrie

1. Pentru $A = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}$ calculaţi valorile proprii şi o bază ortonormală de vectori proprii. Scrieţi diagonalizarea ortogonală $A = Q\Lambda Q^T$.
2. Construiţi o matrice simetrică 2×2 cu valori proprii prescrise 1 şi 100, ale cărei direcţii proprii sunt obţinute printr-o rotaţie de 30°. Scrieţi explicit Q, Λ şi $A = Q\Lambda Q^T$.
3. Demonstraţi că pentru orice matrice B cu rang coloană complet, matricea Gram B^TB este pozitiv definită.
4. Daţi o matrice reală 2×2 cu valori proprii complexe. Explicaţi de ce aceasta nu contrazice teorema spectrală pentru matrici reale simetrice.
5. Demonstraţi, folosind teorema spectrală, că o matrice simetrică este pozitiv definită dacă şi numai dacă toate valorile proprii sunt strict pozitive.
6. Pentru o matrice simetrică A, arătaţi că valorile expresiei x^TAx pe vectori unitari sunt cuprinse între cea mai mică şi cea mai mare valoare proprie.
7. Verificaţi criteriul lui Sylvester pentru trei matrici simetrice alese astfel încât una să fie pozitiv definită, una pozitiv semidefinită şi una indefinită. Comparaţi cu semnele valorilor proprii.
8. Construiţi o matrice simetrică strict diagonal dominantă cu diagonală pozitivă şi verificaţi direct pozitiva definire. Construiţi apoi o matrice pozitiv definită care nu este diagonal dominantă.
9. Consideraţi matricea
$$L = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}.$$
Calculaţi valorile proprii şi identificaţi vectorul propriu asociat valorii proprii zero. Explicaţi algebric de ce suma elementelor de pe fiecare linie fiind zero implică existenţa acestui vector propriu. Interpretări ale unei asemenea matrice prin grafuri vor fi introduse numai ulterior.
10. Comparaţi două matrici similare A şi SAS^{-1}. Verificaţi că au aceleaşi valori proprii, dar vectorii proprii se transformă prin S.

Laborator: matrice simetrice cu spectru prescris

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Construiţi exemple pentru care spectrul este cunoscut dinainte şi verificaţi teorema spectrală.

1. În dimensiunea 2, luaţi explicit matricea de rotaţie $Q(\theta)$ şi $\Lambda = \mathrm{diag}(1, 10)$. Formaţi $A = Q\Lambda Q^T$ pentru mai multe valori ale lui θ.
2. Verificaţi numeric că A este simetrică şi că $AQ = Q\Lambda$, comparând element cu element cele două matrici.
3. Reprezentaţi curbele $x^TAx = 1$ pentru cel puţin trei rapoarte diferite între valorile proprii şi explicaţi schimbarea geometrică.

4. Generaţi mai mulţi vectori unitari x şi calculaţi $x^T Ax$. Verificaţi experimental intervalul determinat de valorile proprii extreme.
5. Repetaţi construcţia în dimensiunea 3 folosind o matrice ortogonală simplă construită din rotaţii plane succesive.
6. Formulaţi o concluzie despre relaţia dintre valorile proprii, direcţiile proprii şi geometria formei pătratice.

Mini-proiect: axele unui nor de puncte

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Consideraţi punctele

$$(-3,-2),\ (-2,-1),\ (-1,0),\ (1,0),\ (2,1),\ (3,2),$$

al căror centru este originea, şi formaţi matricea $X \in \mathbb{R}^{6\times 2}$ având aceste puncte pe linii. Definiţi $C = X^T X$.

1. Calculaţi C şi verificaţi că este simetrică şi pozitiv semidefinită. Justificaţi direct relaţia $z^T C z = \|Xz\|_2^2$.
2. Calculaţi valorile proprii şi o bază ortonormală de vectori proprii ai lui C.
3. Reprezentaţi punctele în plan şi trasaţi prin origine cele două axe determinate de vectorii proprii. Comparaţi orientarea lor cu forma norului de puncte.
4. Rotiţi toate punctele cu acelaşi unghi, folosind o matrice ortogonală Q, şi formaţi noua matrice $\widetilde{C}$. Verificaţi că valorile proprii nu se schimbă, iar axele se rotesc odată cu punctele.
5. Modificaţi coordonatele astfel încât norul să fie mai alungit şi urmăriţi raportul dintre cea mai mare şi cea mai mică valoare proprie.

Test de capitol

Test – Matrice, valori proprii şi pozitiva definire

1. Calcul explicit. Se consideră

$$A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

a) Determinaţi valorile proprii şi câte un vector propriu pentru fiecare dintre ele.

b) Construiţi o bază ortonormală de vectori proprii şi scrieţi descompunerea spectrală $A = Q\Lambda Q^T$.

c) Decideţi, justificat în două moduri diferite, dacă A este pozitiv definită.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi. Pentru o matrice reală simetrică A:

A) toate valorile proprii sunt reale;

B) vectorii proprii corespunzători unor valori proprii distincte pot fi aleşi ortogonali;

C) A este pozitiv definită dacă şi numai dacă toate valorile proprii sunt strict pozitive;

D) orice matrice cu determinant pozitiv este pozitiv definită.

Capitolul 3

Metode directe pentru sisteme liniare

3.1 Rezolvarea sistemelor triunghiulare

Rezolvarea sistemelor inferior triunghiulare

Pentru exemplificare să considerăm sistemul

$$\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}, \tag{3.1}$$

unde $l_{ii} \neq 0$, $i = 1, 2, 3$.

Ultima condiţie ne asigură că matricea sistemului este inversabilă, soluţia fiind dată de

$$\begin{aligned} x_1 &= \frac{b_1}{l_{11}}, \\ x_2 &= \frac{b_2 - l_{21}x_1}{l_{22}}, \\ x_3 &= \frac{b_3 - l_{31}x_1 - l_{32}x_2}{l_{33}}. \end{aligned}$$

Acest algoritm se numeşte metoda substituţiei înainte (*forward substitution*).

În general, pentru $n \geq 2$ avem

$$\begin{aligned} x_1 &= \frac{b_1}{l_{11}}, \\ x_i &= \frac{b_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}, \qquad i = 2, \ldots, n. \end{aligned} \tag{3.2}$$

Numărarea operaţiilor aritmetice dă un cost de ordin n^2 pentru substituţia înainte. Pentru comparaţia dintre metode este suficient, de regulă, termenul dominant. Aplicarea regulii lui Cramer prin dezvoltarea determinantelor are, în schimb, un cost de ordin factorial.

Rezolvarea sistemelor superior triunghiulare

Pentru exemplificare să considerăm sistemul

$$\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}, \tag{3.3}$$

unde $u_{ii} \neq 0$, $i = 1, 2, 3$.

Ultima condiţie ne asigură că matricea sistemului este inversabilă, soluţia în cazul general fiind dată de

$$x_n = \frac{b_n}{u_{nn}},$$
$$x_i = \frac{b_i - \sum_{j=i+1}^{n} u_{ij} x_j}{u_{ii}}, \qquad i = n-1, ..., 1. \tag{3.4}$$

Acest algoritm se numeşte metoda substituţiei înapoi (*backward substitution*).

Avem tot n^2 operaţii.

Pentru implementare

Pentru implementare ar fi indicat să stocăm doar elementele nenule atunci când avem de rezolvat sistemele triunghiulare.

Completare didactică: substituţia înainte şi înapoi

Rezolvarea sistemelor triunghiulare este "motorul" ascuns al multor metode directe. Factorizările LU, Cholesky şi QR reduc o problemă aparent complicată la una sau mai multe rezolvări triunghiulare. De aceea algoritmul trebuie înţeles şi implementat fără calcularea vreunei inverse.

Pentru un sistem inferior triunghiular $Lx = b$, ecuaţia i conţine necunoscutele $x_1, \dots, x_i$, iar primele $i-1$ au fost deja calculate. Rezultă formula scurtă $x_i = (b_i - \sum_{j=1}^{i-1} l_{ij} x_j)/l_{ii}$. În cazul unui sistem superior triunghiular se procedează în ordine inversă.

Exemplu ghidat

Considerăm

$$L = \begin{pmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & 2 & 4 \end{pmatrix}, \qquad b = \begin{pmatrix} 4 \\ 1 \\ 10 \end{pmatrix}.$$

Prima ecuaţie dă $x_1 = 2$. A doua devine $-2 + x_2 = 1$, deci $x_2 = 3$. Ultima ecuaţie este $6 + 6 + 4x_3 = 10$, de unde $x_3 = -\frac{1}{2}$. Observaţi că nu a fost nevoie de nicio operaţie asupra unei linii deja rezolvate; fiecare pas utilizează doar valorile calculate anterior.

Pentru o matrice triunghiulară densă de ordin n, numărul de termeni utilizaţi în sume este $1 + 2 + \cdots + (n-1) = n(n-1)/2$. Costul este deci de ordin $O(n^2)$, mult mai mic decât costul unei factorizări dense, de ordin $O(n^3)$.

Implementare fără funcţii de rezolvare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o funcţie MATLAB pentru substituţia înainte.

```
function x = forward_sub(L,b)
    n = length(b);
    x = zeros(n,1);
    for i = 1:n
        s = L(i,1:i-1)*x(1:i-1);
        x(i) = (b(i)-s)/L(i,i);
    end
end
```

Scrieţi apoi funcţia analogă `back_sub(U,b)`. Testaţi ambele funcţii pe matrice triunghiulare aleatoare şi comparaţi cu operatorul \. Introduceţi explicit un test care opreşte algoritmul atunci când un element diagonal este zero sau prea mic în raport cu scala problemei.

Exerciţii şi verificarea înţelegerii

1. Rezolvaţi manual sistemul cu $U = \begin{pmatrix} 1 & -2 & 1 \\ 0 & 3 & 6 \\ 0 & 0 & 2 \end{pmatrix}$ şi $b = (0, 9, 4)^T$.
2. Calculaţi exact numărul de înmulţiri, adunări/scăderi şi împărţiri necesare substituţiei înainte pentru o matrice densă $n \times n$.
3. Dacă L este triunghiulară inferior şi toate elementele diagonale sunt nenule, demonstraţi unicitatea soluţiei fără a invoca formula determinantului.
4. Modificaţi algoritmul pentru cazul în care L are diagonala identică cu 1. Ce operaţii dispar?
5. Explicaţi de ce rezolvarea separată a n sisteme triunghiulare cu termenii liberi $e_1, \ldots, e_n$ produce inversa matricei, dar este inutilă atunci când avem nevoie de un singur vector $L^{-1}b$.

Mini-proiect: influenţa structurii rare

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi matrice triunghiulare dense şi rare de dimensiuni până la limita rezonabilă a calculatorului folosit. Comparaţi memoria şi timpul de rezolvare. Pentru matricea rară, folosiţi doar câteva diagonale nenule. Explicaţi de ce o implementare care parcurge toate elementele unei matrice rare poate pierde avantajul structurii rare.

Rezolvare orientativă

Substituţie înapoi, fără scurtături

Pentru $U = \begin{pmatrix} 1 & -2 & 1 \\ 0 & 3 & 6 \\ 0 & 0 & 2 \end{pmatrix}$ şi $b = (0, 9, 4)^T$, ultima ecuaţie dă $2x_3 = 4$, deci $x_3 = 2$. A doua devine $3x_2 + 12 = 9$, deci $x_2 = -1$. Prima este $x_1 - 2(-1) + 2 = 0$, de unde $x_1 = -4$. Verificarea este obligatorie: $Ux = (-4 + 2 + 2, -3 + 12, 4)^T = (0, 9, 4)^T$.
Observaţi că ordinea este impusă de structură. Dacă am încerca să calculăm x_1 mai întâi, prima ecuaţie conţine încă două necunoscute. Algoritmul nu este doar o formulă, ci o ordine de dependenţă între calcule.

Situaţie practică: mulţi termeni liberi

Dacă L este fixă şi avem matricea $B = (b^{(1)}, \ldots, b^{(p)})$, putem rezolva $LX = B$ coloană cu coloană. În MATLAB, operaţiile pot fi vectorizate astfel încât la pasul i să actualizăm simultan toate cele p coloane. Costul creşte aproximativ proporţional cu p, dar citirea factorului L poate fi reutilizată eficient din memorie. Acest scenariu apare după o factorizare LU când acelaşi model este excitat de mai multe forţe.

Exemplu rezolvat suplimentar: substituţia înainte fără salturi de calcul

Considerăm sistemul $Lx = b$, cu

$$L = \begin{pmatrix} 2 & 0 & 0 \\ -1 & 3 & 0 \\ 4 & 2 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 4 \\ 5 \\ 3 \end{pmatrix}.$$

Prima ecuaţie conţine o singură necunoscută, deci $2x_1 = 4$ şi $x_1 = 2$. A doua ecuaţie devine $-x_1 + 3x_2 = 5$, de unde $3x_2 = 7$ şi $x_2 = 7/3$. Abia după ce primele două componente sunt cunoscute folosim ultima ecuaţie: $4x_1 + 2x_2 + x_3 = 3$. Rezultă $x_3 = 3 - 8 - 14/3 = -29/3$. Observaţi că fiecare pas foloseşte numai componente deja calculate; aceasta este proprietatea structurală care face posibilă substituţia înainte.

Verificarea nu trebuie omisă. Pentru un exemplu mic calculăm Lx şi verificăm dacă recuperăm vectorul b. În implementările numerice putem compara direct componentele celor doi vectori; instrumentele sistematice pentru măsurarea acestei diferenţe vor fi introduse în capitolul despre norme şi analiza erorii.

De ce costul este de ordin n^2?

La pasul i trebuie evaluată o sumă cu $i - 1$ termeni. Numărul total de înmulţiri din aceste sume este $0 + 1 + \cdots + (n - 1) = n(n - 1)/2$. Acelaşi ordin apare pentru adunări, iar cele n împărţiri nu schimbă ordinul dominant. De aceea substituţia triunghiulară costă $O(n^2)$, mult mai puţin decât factorizarea unei matrice dense, care costă $O(n^3)$.

Sarcină de implementare: `forwardSubstitution`

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă

este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o funcţie MATLAB care primeşte L şi b şi întoarce x. Funcţia trebuie să verifice dacă dimensiunile sunt compatibile şi să semnaleze un pivot diagonal nul sau prea mic. Nu folosiţi operatorul \ sau o rutină de rezolvare existentă. Testaţi funcţia astfel:

1. generaţi un vector $x_\star$ aleator şi o matrice triunghiulară L cu diagonală depărtată de zero;
2. construiţi $b = Lx_\star$;
3. calculaţi $\widehat{x}$ cu funcţia proprie;
4. comparaţi componentă cu componentă $\widehat{x}$ cu $x_\star$ şi $L\widehat{x}$ cu b; raportaţi cea mai mare diferenţă în valoare absolută în fiecare comparaţie;
5. repetaţi pentru $n = 10, 100, 1000$ şi măsuraţi timpul.

Explicaţi de ce această metodă de testare este mai informativă decât alegerea arbitrară a unui termen liber pentru care soluţia exactă nu este cunoscută.

Exerciţii suplimentare

1. Rezolvaţi prin substituţie înapoi sistemul cu $U = \begin{pmatrix} 2 & -1 & 3 \\ 0 & 4 & 2 \\ 0 & 0 & -5 \end{pmatrix}$ şi $b = (6, 6, -10)^T$. Verificaţi soluţia prin înmulţire.
2. Arătaţi că o matrice triunghiulară este inversabilă dacă şi numai dacă toate elementele diagonale sunt nenule. Folosiţi argumentul pentru a justifica existenţa fiecărui pas al substituţiei.
3. Modificaţi algoritmul pentru cazul în care L este triunghiulară unitară. Câte împărţiri se economisesc?
4. Pentru o matrice triunghiulară cu bandă de lăţime p, arătaţi că substituţia poate fi făcută în $O(np)$ operaţii, nu în $O(n^2)$.
5. Construiţi o matrice triunghiulară cu elemente diagonale de ordine foarte diferită şi urmăriţi numeric substituţia. Ce dificultăţi observaţi când unul dintre elementele diagonale este foarte mic?

3.2 Eliminarea gaussiană şi factorizarea LU

Eliminarea gaussiană

Eliminarea gaussiană ne ajută să reducem un sistem

$$A\,x = b$$

la un sistem triunghiular (sau la două sisteme triunghiulare) prin transformări succesive ale sistemului în sisteme echivalente

$$A^{(1)}\,x = b^{(1)} \to A^{(2)}\,x = b^{(2)} \to \cdots \to A^{(k)}\,x = b^{(k)}.$$

Presupunem că la fiecare pas elementul $a_{kk}^{(k)}$ al matricei $A^{(k)}$ este nenul.

Acest element va fi numit pivot.

Presupunem că A este inversabilă, adică sistemul admite soluţie unică.

Pasul 1

Plecăm de la sistemul

$$\underbrace{\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & \cdots & a_{2n}^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & \cdots & \cdots & a_{3n}^{(1)} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & \cdots & a_{nn}^{(1)} \end{pmatrix}}_{\equiv A} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{pmatrix}$$

Definim multiplicatorii $m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$, $i = 2, 3, ..., n$, înmulţim prima linie cu m_{i1}, pe rând, şi scădem rezultatele din linia $i = 2, 3, ..., n$, respectiv.

Pasul 1

Se obţine astfel un nou sistem, echivalent cu cel iniţial

$$\underbrace{\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & \cdots & a_{2n}^{(2)} \\ 0 & a_{32}^{(2)} & \cdots & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & \cdots & a_{nn}^{(2)} \end{pmatrix}}_{:=A^{(2)}} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1^{(2)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{pmatrix}.$$

Pasul k

Repetând procedeul, la pasul k se obţine astfel un nou sistem, echivalent cu cel iniţial

$$\underbrace{\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & \cdots & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}}_{:=A^{(k)}} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \underbrace{\begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_k^{(k)} \\ \vdots \\ b_n^{(k)} \end{pmatrix}}_{:=b^{(k)}}.$$

Pasul $n-1$

După $n-1$ paşi se obţine astfel un nou sistem, echivalent cu cel iniţial dar **superior triunghiular**

$$\underbrace{\begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & \cdots & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & a_{nn}^{(n)} \end{pmatrix}}_{:=A^{(n)}} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \underbrace{\begin{pmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_k^{(k)} \\ \vdots \\ b_n^{(n)} \end{pmatrix}}_{:=b^{(n)}}.$$

Algoritm

În concluzie, formulele după care se modifică sistemul de la pasul k în sistemul de la pasul $k+1$ sunt

$$\begin{aligned} m_{ik} &= \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, && i = k+1, ..., n, \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, && i, j = k+1, ..., n, \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)}, && i = k+1, \ldots, n. \end{aligned} \tag{3.5}$$

Numărul de operaţii

- Triangularizarea prin eliminare Gauss are costul dominant

$$\#\text{op(triangularizare)} = \frac{2}{3}n^3 + O(n^2).$$

- Substituţia înapoi necesită $n^2 + O(n)$ operaţii.
- Prin urmare, costul total rămâne

$$\#\text{op(GEM)} = \frac{2}{3}n^3 + O(n^2).$$

Pentru compararea algoritmilor ne interesează în primul rând termenul dominant; formulele exacte depind de convenţia folosită pentru numărarea împărţirilor, înmulţirilor şi adunărilor.

GEM funcţionează dacă $a_{kk}^{(k)} \neq 0$, $k = 1, 2, ..., n-1$.

Din păcate, faptul că diagonala iniţială este nenulă nu garantează că la un pas ulterior k vom avea $a_{kk}^{(k)} \neq 0$.

De exemplu, considerând matricea

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{pmatrix}$$

după primul pas găsim

$$A^{(2)} = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{pmatrix}.$$

Ce e de făcut?

Nu abandonăm metoda, ci identificăm cauza eşecului şi construim o strategie care acoperă toate cazurile: pivotarea.

Regândim teoretic problema fără a recurge la “peticiri” de moment.

Anticipăm

Poate putem şti de la bun început dacă pentru o matrice este potrivit sau nu să folosim GEM?

Într-adevăr sunt criterii care ne asigură că putem folosi GEM, de exemplu

- Matrice strict diagonal dominante pe linii sau pe coloane.
- Matrice simetrice pozitiv definite.

Însă toate aceste afirmaţii trebuie verificate şi argumentate; simplele bănuieli nu constituie justificări matematice.

În continuare urmărim să rescriem matricea $A \in \mathbb{R}^{n\times n}$ sub forma $A = LU$, unde L este triunghiulară inferioară, iar U este triunghiulară superioară.

Facem acest lucru deoarece după ce vom reuşi, sistemul iniţial va putea fi rescris sub forma a două sisteme triunghiulare, adică

$$Ax = b \qquad \leftrightarrow \qquad LUx = b \qquad \leftrightarrow \qquad \begin{cases} Ly = b \\ Ux = y \end{cases} \tag{3.6}$$

ce se vor rezolva pe rând.

GEM prin multiplicare de matrice

Operaţiile de eliminare din prima coloană se pot scrie prin înmulţire la stânga cu

$$M_1 = I_n - m_1 e_1^T = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -m_{21} & 1 & 0 & \cdots & 0 \\ -m_{31} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -m_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix}, \qquad m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}.$$

Astfel $M_1 A^{(1)} = A^{(2)}$. La pasul următor folosim analog

$$M_2 = I_n - m_2 e_2^T, \qquad M_2 A^{(2)} = A^{(3)},$$

şi, repetând procedeul,

$$M_{n-1} \cdots M_2 M_1 A = A^{(n)} =: U,$$

unde U este triunghiulară superioară. Această scriere conduce direct la factorizarea LU.

Eliminarea gaussiană

Considerăm o matrice $A \in \mathbb{R}^{n\times n}$. Scopul este de a construi o secvenţă $A^{(k)} = (a_{ij}^{(k)})$ de matrici prin efectuarea de transformări liniare, astfel încât să ajungem la o matrice triunghiulară superioară $U = (u_{ij})$ după un număr finit de paşi.

În ultima săptămână am văzut că dacă presupunem că pivoţii $a_{11} \neq 0$, $a_{kk}^{(k)} = (M_{k-1}...M_1\, A)_{kk} \neq 0$ la orice pas $k = 2, ..., n-1$, atunci

$$M_{n-1} M_{n-2} ... M_1\, A = U, \tag{3.7}$$

cu U o matrice triunghiulară superioară, unde

$$M_k = \begin{pmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & -m_{k+1,k} & 1 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & -m_{n,k} & 0 & \cdots & 1 \end{pmatrix} = I_n - m_k\, e_k^T, \tag{3.8}$$

şi

$$m_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ m_{k+1,k} \\ \vdots \\ m_{n,k} \end{pmatrix} \in \mathbb{R}^n, \qquad e_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^n, \quad m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, i = k+1, \cdots, n. \tag{3.9}$$

Mai mult, deoarece $M_k^{-1} = I_n + m_k e_k^T$, putem reconstrui matricea iniţială din matricea triunghiulară U. Într-adevăr, $A = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} U$. Pentru $i < j$ avem $e_i^T m_j = 0$, deoarece primele j componente ale lui m_j sunt nule. Ca urmare, toţi termenii încrucişaţi din produsul matricelor $I_n + m_i e_i^T$ dispar şi obţinem

$$A = \left(I_n + \sum_{i=1}^{n-1} m_i e_i^T \right) U = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ m_{21} & 1 & 0 & \cdots & \cdots & \vdots \\ m_{31} & m_{32} & 1 & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & 0 \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} & & 1 \end{pmatrix}}_{:=L} U \tag{3.10}$$

Deci dacă nu întâlnim pivoţi nuli ($a_{kk}^{(k)}$) atunci putem construi factorizarea LU a acelei matrice.

Dar cum ştim dacă vom întâlni pivoţi nuli fără a începe procesul de construcţie al factorizării?

Existenţa şi unicitatea factorizării LU

Teoremă 3.2.1. *Fie $A \in \mathbb{R}^{n\times n}$. Există factorizarea LU a matricei A cu $l_{ii} = 1$ pentru orice $i = 1, \cdots, n$ şi este unică dacă şi numai dacă submatricele principale $A_i = A(1:i, 1:i)$ ale lui de orice ordin $i = 1, \cdots, n-1$ sunt nesingulare.*

Demonstraţie. Vom demonstra mai întâi implicaţia "$\Longleftarrow$". Presupunem că submatricele principale $A_i = A(1:i, 1:i)$ sunt nesingulare pentru $i = 1, \ldots, n-1$. Construcţia factorilor se face prin inducţie şi, în acelaşi timp, arată unicitatea lor.

Pentru $i = 1$ avem $A_1 = (1)(a_{11})$. Deoarece A_1 este nesingulară atunci când $n > 1$, rezultă $a_{11} \neq 0$. Presupunem acum că $A_{i-1} = L_{i-1}U_{i-1}$, unde L_{i-1} este triunghiulară inferioară unitară, iar U_{i-1} este triunghiulară superioară. Scriem bloc matricea principală următoare sub forma

$$A_i = \begin{pmatrix} A_{i-1} & c \\ d^T & a_{ii} \end{pmatrix} = \begin{pmatrix} L_{i-1} & 0 \\ \ell^T & 1 \end{pmatrix} \begin{pmatrix} U_{i-1} & u \\ 0^T & u_{ii} \end{pmatrix}.$$

Prin identificarea blocurilor trebuie să avem $L_{i-1}u = c$, $U_{i-1}^T \ell = d$ şi $u_{ii} = a_{ii} - \ell^T u$. Deoarece A_{i-1} este nesingulară şi $\det L_{i-1} = 1$, rezultă $\det U_{i-1} = \det A_{i-1} \neq 0$. Aşadar cele două sisteme triunghiulare au soluţii unice, $u = L_{i-1}^{-1} c$ şi $\ell = U_{i-1}^{-T} d$, după care u_{ii} este determinat în mod unic. Pentru pasul final $i = n$ este suficientă nesingularitatea lui A_{n-1}; matricea A însăşi poate fi singulară, caz în care este posibil ca $u_{nn} = 0$.

Pentru implicaţia "$\Longrightarrow$", presupunem că $A = LU$ este o factorizare cu L inferior triunghiulară unitară şi că această factorizare este unică. Pentru fiecare i, blocul principal verifică $A_i = L_i U_i$, de unde $\det A_i = \prod_{j=1}^{i} u_{jj}$. Rămâne să arătăm că $u_{kk} \neq 0$ pentru orice $k < n$.

Presupunem, prin reducere la absurd, că $u_{kk} = 0$ pentru un anumit $k < n$. Pentru $t \in \mathbb{R}$ definim

$$S_t = I + te_{k+1}e_k^T, \qquad S_t^{-1} = I - te_{k+1}e_k^T,$$

unde am folosit $(e_{k+1}e_k^T)^2 = 0$. Atunci $A = (LS_t)(S_t^{-1}U)$. Matricea LS_t rămâne inferior triunghiulară unitară. În acelaşi timp, $S_t^{-1}U$ se obţine scăzând din linia $k+1$ a lui U de t ori linia k. Singura intrare care ar putea apărea sub diagonală în coloana k este $-tu_{kk} = 0$, deci $S_t^{-1}U$ rămâne superior triunghiulară. Pentru $t \neq 0$ obţinem în general factori diferiţi, ceea ce contrazice unicitatea. Prin urmare $u_{kk} \neq 0$ pentru $k = 1, \ldots, n-1$, iar $\det A_i = \prod_{j=1}^{i} u_{jj} \neq 0$ pentru $i = 1, \ldots, n-1$. □

De ce nu se cere ca A să fie nesingulară?

Condiţia priveşte numai minorii principali de ordine $1, \ldots, n-1$. Ultimul pivot u_{nn} poate fi zero fără a mai exista un rând inferior pe care să-l putem modifica pentru a produce o altă factorizare. De aceea o matrice singulară poate avea totuşi o factorizare LU unică; primul exemplu de mai jos ilustrează exact această situaţie.

Exemplu 3.2.2. Considerăm matricele

$$B = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}, \qquad C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad D = \begin{pmatrix} 0 & 1 \\ 0 & 2 \end{pmatrix}. \tag{3.11}$$

- B admite o unică factorizare LU.
- matricea neinversabilă C nu admite factorizare LU.
- matricea neinversabilă D admite o infinitate de factorizări de forma $D = L_\beta U_\beta$, cu

$$L_\beta = \begin{pmatrix} 1 & 0 \\ \beta & 1 \end{pmatrix}, \quad U_\beta = \begin{pmatrix} 0 & 1 \\ 0 & 2-\beta \end{pmatrix} \quad \forall\, \beta \in \mathbb{R}.$$

Există un alt rezultat:

Teoremă 3.2.3. *Dacă A este o matrice strict diagonal dominantă (pe linii sau pe coloane), atunci factorizarea LU fără pivotare există şi este unică. În particular, dacă A este diagonal dominantă pe coloane, atunci $|l_{ij}| \leq 1$ $\forall i, j$.*

Demonstraţie. Demonstrăm mai întâi că o matrice strict diagonal dominantă pe linii este nesingulară. Dacă $Ax = 0$ şi $x \neq 0$, alegem i astfel încât $|x_i| = \max_j |x_j| > 0$. Din ecuaţia i rezultă

$$|a_{ii}|\,|x_i| = \left|\sum_{j\neq i} a_{ij}x_j\right| \leq \sum_{j\neq i} |a_{ij}|\,|x_j| \leq \left(\sum_{j\neq i} |a_{ij}|\right)|x_i|,$$

contradicţie cu diagonal-dominanţa strictă. Deci matricea este nesingulară. Fiecare submatrice principală a unei matrice strict diagonal dominante pe linii este la rândul ei strict diagonal dominantă pe linii; toţi minorii principali conducători de ordine $1, \ldots, n-1$ sunt deci nenuli. Teorema de existenţă şi unicitate a factorizării LU se aplică.

Dacă A este strict diagonal dominantă pe coloane, atunci A^T este strict diagonal dominantă pe linii. Submatricele principale ale lui A^T au aceiaşi determinanţi ca cele ale lui A, deci concluzia privind existenţa şi unicitatea urmează din acelaşi criteriu.

Rămâne să justificăm estimarea multiplicatorilor. Vom arăta prin inducţie că submatricea activă după fiecare pas al eliminării păstrează dominanţa strictă pe coloane. Presupunem că, la pasul

k, matricea activă $A^{(k)} = (a_{ij}^{(k)})_{i,j=k}^{n}$ este strict diagonal dominantă pe coloane şi definim, pentru $i > k$, multiplicatorii $m_i = a_{ik}^{(k)}/a_{kk}^{(k)}$. Pentru $i, j > k$ avem

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_i a_{kj}^{(k)}.$$

Fixăm $j > k$. Din dominanţa coloanei j rezultă

$$|a_{jj}^{(k)}| > |a_{kj}^{(k)}| + \sum_{\substack{i>k \\ i\neq j}} |a_{ij}^{(k)}|,$$

iar din dominanţa coloanei k,

$$|a_{kk}^{(k)}| > |a_{jk}^{(k)}| + \sum_{\substack{i>k \\ i\neq j}} |a_{ik}^{(k)}|.$$

Prin urmare

$$\begin{aligned}
|a_{jj}^{(k+1)}| &= |a_{jj}^{(k)} - m_j a_{kj}^{(k)}| \\
&\geq |a_{jj}^{(k)}| - \frac{|a_{jk}^{(k)}|}{|a_{kk}^{(k)}|}|a_{kj}^{(k)}| \\
&> \sum_{\substack{i>k \\ i\neq j}} |a_{ij}^{(k)}| + \frac{|a_{kj}^{(k)}|}{|a_{kk}^{(k)}|} \sum_{\substack{i>k \\ i\neq j}} |a_{ik}^{(k)}| \\
&\geq \sum_{\substack{i>k \\ i\neq j}} \left| a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)} \right| = \sum_{\substack{i>k \\ i\neq j}} |a_{ij}^{(k+1)}|.
\end{aligned}$$

Aşadar complementul Schur este din nou strict diagonal dominant pe coloane. Inducţia porneşte de la $A^{(1)} = A$.

În particular, în coloana pivotului avem $|a_{ik}^{(k)}| < |a_{kk}^{(k)}|$ pentru orice $i > k$. Cum $l_{ik} = m_i$, rezultă

$$|l_{ik}| = \frac{|a_{ik}^{(k)}|}{|a_{kk}^{(k)}|} < 1,$$

şi deci, în forma mai slabă enunţată, $|l_{ij}| \leq 1$ pentru toate elementele subdiagonale ale lui L. □

Completare didactică: eliminarea gaussiană ca factorizare

Eliminarea gaussiană nu trebuie privită ca o succesiune de trucuri asupra liniilor. Fiecare pas este o multiplicare la stânga cu o matrice elementară, iar toţi multiplicatorii pot fi stocaţi într-o matrice triunghiulară inferior L. La final obţinem, în absenţa pivotării, factorizarea $A = LU$. Această interpretare separă două faze: o fază costisitoare, în care factorizăm A, şi o fază ieftină, în care rezolvăm $Ly = b$ şi apoi $Ux = y$.

Exemplu complet

Fie

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{pmatrix}.$$

Primul pivot este 2. Multiplicatorii sunt $m_{21} = 2$ şi $m_{31} = -1$, iar după eliminarea primei coloane rămân liniile $(0, -8, -2)$ şi $(0, 8, 3)$. Al doilea multiplicator este $m_{32} = -1$, ceea ce conduce la

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 0 & 1 \end{pmatrix}, \qquad A = LU.$$

Dacă alegem $x_\star = (1, 2, 3)^T$, atunci $b = Ax_\star = (7, -8, 18)^T$. Rezolvând succesiv $Ly = b$ şi $Ux = y$ recuperăm $x_\star$.

De ce păstrăm factorizarea?

Dacă avem aceeaşi matrice A şi mai mulţi termeni liberi $b^{(1)}, \ldots, b^{(p)}$, factorizarea se calculează o singură dată. Pentru fiecare termen liber nou efectuăm doar două substituţii de cost $O(n^2)$. Acest scenariu apare în simulări parametrice, metode Newton, probleme de control şi discretizări dependente de timp.

Costul dominant al eliminării gaussiene pentru o matrice densă este aproximativ $\frac{2}{3}n^3$ operaţii aritmetice de tip înmulţire/adunare, în timp ce cele două rezolvări triunghiulare au cost de ordin n^2. Diferenţa devine enormă pentru multe coloane de termeni liberi.

Laborator: o implementare LU elementară

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi eliminarea gaussiană fără pivotare şi salvaţi multiplicatorii sub diagonală, în aceeaşi matrice în care se formează U. Nu construiţi L la fiecare pas. La final extrageţi L şi U numai pentru verificare. Testaţi relaţia `norm(A-L*U)` şi rezolvarea unui sistem cu soluţie cunoscută.
Apoi modificaţi programul astfel încât să accepte o matrice B cu mai mulţi termeni liberi, rezolvând $AX = B$. Cronometraţi separat timpul de factorizare şi timpul rezolvărilor triunghiulare.

Exerciţii şi verificarea înţelegerii

1. Calculaţi manual factorizarea LU pentru $A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 4 & 6 \end{pmatrix}$, dacă aceasta poate fi obţinută fără pivotare.
2. Pornind de la matricile elementare de eliminare $E_1, \ldots, E_{n-1}$, explicaţi de ce $U = E_{n-1} \cdots E_1 A$ şi $L = (E_{n-1} \cdots E_1)^{-1}$.
3. Arătaţi că dacă toţi minorii principali de ordin $1, \ldots, n$ sunt nenule, eliminarea gaussiană poate fi efectuată fără schimbări de linii.
4. Pentru o matrice A deja factorizată, estimaţi costul rezolvării a p sisteme $Ax = b^{(j)}$. Comparaţi cu refactorizarea lui A pentru fiecare j.
5. Explicaţi de ce formarea explicită a lui A^{-1} prin LU şi apoi calculul $A^{-1}b$ efectuează mai multă muncă decât rezolvarea directă a celor două sisteme triunghiulare.

Studiu practic: discretizarea ecuaţiei Poisson

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Porniţi de la discretizarea unidimensională a problemei $-u'' = f$ pe $(0,1)$ cu condiţii Dirichlet. Obţineţi sistemul tridiagonal corespunzător pentru mai multe valori ale pasului h. Rezolvaţi sistemul mai întâi folosind o rutină LU densă scrisă de voi, apoi folosind structura tridiagonală. Comparaţi timpul, memoria şi eroarea faţă de o soluţie exactă aleasă în prealabil.

Rezolvare orientativă

Factorizare LU completă

Pentru $A = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 5 & 3 \\ 1 & 4 & 6 \end{pmatrix}$, primul pivot este 1. Eliminăm cu multiplicatorii $l_{21} = 2$ şi $l_{31} = 1$, obţinând liniile $(0, 1, 1)$ şi $(0, 2, 5)$. Al doilea pivot este 1, iar $l_{32} = 2$; ultima linie devine $(0, 0, 3)$. Astfel

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 2 & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 3 \end{pmatrix}.$$

Înmulţirea directă confirmă $LU = A$. Dacă termenul liber este $b = A(1,1,1)^T = (4, 10, 11)^T$, rezolvăm mai întâi $Ly = b$: $y_1 = 4$, $y_2 = 2$, $y_3 = 3$. Apoi $Ux = y$ dă $x_3 = 1$, $x_2 = 1$, $x_1 = 1$.

Situaţie practică: aceeaşi matrice, mai mulţi termeni liberi

Un motiv pentru a păstra o factorizare este rezolvarea mai multor sisteme $Ax = b^{(j)}$ cu aceeaşi matrice A şi termeni liberi diferiţi. De exemplu, într-un model mecanic liniar, necunoscutele pot fi deplasări, iar ecuaţiile de echilibru pot avea forma $Kx = f$; matricea K, care transformă deplasările în forţe interne, este numită în mecanică *matrice de rigiditate*. Nu vom folosi aici teoria mecanică, ci doar faptul că K rămâne aceeaşi când se schimbă forţa exterioară f. Astfel, matricea poate rămâne aceeaşi, iar vectorul termenilor liberi se schimbă. Dacă factorizarea costă aproximativ $O(n^3)$ şi fiecare rezolvare triunghiulară $O(n^2)$, pentru 1000 de termeni liberi costul total este $O(n^3 + 1000n^2)$, nu $1000O(n^3)$. Reutilizarea factorizării este deseori diferenţa dintre un calcul fezabil şi unul inutil de costisitor.

Exemplu complet: factorizare, două sisteme triunghiulare şi verificare

Fie

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 5 \\ -2 \\ 9 \end{pmatrix}.$$

Eliminarea fără pivotare produce multiplicatorii $m_{21} = 2$ şi $m_{31} = -1$. După primul pas, liniile a doua şi a treia devin $(0, -8, -2)$, respectiv $(0, 8, 3)$. La al doilea pas, $m_{32} = -1$, iar ultima linie devine $(0, 0, 1)$. Prin urmare

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 0 & 1 \end{pmatrix},$$

şi se verifică direct $A = LU$.

Pentru $Ax = b$ nu calculăm A^{-1}. Rezolvăm mai întâi $Ly = b$. Din prima ecuaţie $y_1 = 5$, din a doua $2y_1 + y_2 = -2$, deci $y_2 = -12$, iar din a treia $-y_1 - y_2 + y_3 = 9$, deci $y_3 = 2$. Apoi rezolvăm $Ux = y$: ultima ecuaţie dă $x_3 = 2$, a doua dă $-8x_2 - 4 = -12$, deci $x_2 = 1$, iar prima dă $2x_1 + 1 + 2 = 5$, deci $x_1 = 1$. Soluţia este $x = (1, 1, 2)^T$.

Acest calcul trebuie citit în trei niveluri: eliminarea construieşte factorii, factorii permit rezolvarea rapidă, iar rezolvarea pentru mai mulţi termeni liberi refoloseşte aceeaşi factorizare. Dacă avem $AX = B$ cu s coloane în B, costul mare $O(n^3)$ al factorizării este plătit o singură dată, iar fiecare coloană suplimentară necesită doar două substituţii de ordin $O(n^2)$.

Problematizare: de ce nu calculăm inversa?

Formula $x = A^{-1}b$ este matematic impecabilă, dar nu descrie algoritmul recomandat. Construirea explicită a inversei înseamnă, în esenţă, rezolvarea a n sisteme cu termeni liberi e_i, urmată de o înmulţire matrice–vector. Pentru un singur sau câţiva termeni liberi, această muncă este inutilă şi poate introduce erori suplimentare. În calcul numeric, „rezolvă sistemul” şi „calculează inversa” sunt operaţii distincte.

Sarcină de programare: propria factorizare LU

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi eliminarea gaussiană fără pivotare astfel încât funcţia să întoarcă explicit L şi U. Adăugaţi un contor pentru operaţiile aritmetice şi o verificare `norm(A-L*U)`. Apoi implementaţi o funcţie separată `luSolve(L,U,b)`. Testaţi pe matrici pentru care soluţia exactă este cunoscută. Într-un al doilea experiment, fixaţi A şi rezolvaţi sistemul pentru 100 de vectori b_j; comparaţi timpul dintre refolosirea factorilor şi refactorizarea lui A la fiecare termen liber.

Exerciţii suplimentare

1. Refaceţi exemplul de mai sus cu aritmetică exactă şi verificaţi fiecare relaţie $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$.
2. Arătaţi că $\det A = \prod_{i=1}^{n} u_{ii}$ atunci când L este unitară. Folosiţi formula pentru a calcula determinantul matricei din exemplu.
3. Explicaţi de ce factorizarea fără pivotare poate eşua pentru o matrice inversabilă. Construiţi un exemplu 2×2.
4. Presupuneţi că A este fixă şi $b = b(t)$ se schimbă la fiecare pas de timp. Descrieţi ce parte a calculului se face o singură dată şi ce parte se repetă.

5. Comparaţi experimental rezolvarea prin LU cu `inv(A)*b` pentru matrici de dimensiuni crescătoare. Folosiţi sisteme cu soluţie fabricată şi raportaţi timpul şi cea mai mare diferenţă absolută dintre componentele soluţiei calculate şi cele exacte.

3.3 Forma compactă a factorizării LU

Forma compactă a factorizării LU

O variantă a factorizării LU este factorizarea **Doolittle**. Aceasta este cunoscută şi ca formă compactă a eliminării gaussiene.

Această denumire se datorează faptului că aceste abordări necesită mai puţine rezultate intermediare decât metoda GEM standard pentru a genera factorizarea lui A.

Calcularea factorizării LU a lui A este echivalentă din punct de vedere formal cu rezolvarea următorului sistem neliniar de ecuaţii n^2

$$a_{ij} = \sum_{r=1}^{\min(i,j)} l_{ir}u_{rj}, i, j = 1, ..., n, \tag{3.12}$$

cele două matrice triunghiulare având împreună $n(n+1)$ intrări potenţial nenule. Deoarece $A = LU$ nu fixează separat scalarea fiecărei perechi coloană–rând, impunem n condiţii de normalizare, de exemplu $l_{ii} = 1$.

Dacă stabilim în mod arbitrar n coeficienţi (l_{ii}) la 1, ajungem la metoda Doolittle care oferă o cale eficientă sistemului neliniar.

De fapt, presupunând că primele $k-1$ coloane din L şi primele rânduri din U sunt disponibile şi stabilind $l_{kk} = 1$ (metoda Doolittle), se obţin următoarele ecuaţii din

$$a_{kj} = \sum_{r=1}^{k-1} l_{kr}u_{rj} + u_{kj}, \quad j = k, \cdots, n, \tag{3.13}$$

$$a_{ik} = \sum_{r=1}^{k-1} l_{ir}u_{rk} + l_{ik}u_{kk}, \quad i = k+1, \cdots, n. \tag{3.14}$$

Reţineţi că aceste ecuaţii pot fi rezolvate într-un mod secvenţial în ceea ce priveşte variabilele roşii u_{kj} şi l_{ik}.

Din metoda compactă Doolittle obţinem astfel mai întâi al k-lea rând al lui U şi apoi a k-a coloană a lui L, după cum urmează: pentru $k = 1, \cdots, n$

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr}u_{rj}, \quad j = k, \cdots, n, \tag{3.15}$$

$$l_{ik} = \frac{1}{u_{kk}} \left(a_{ik} - \sum_{r=1}^{k-1} l_{ir}u_{rk} \right), \quad i = k+1, \cdots, n. \tag{3.16}$$

Completare didactică: forma compactă şi calculul in-place

Într-o implementare matură nu este necesar să păstrăm separat toate matricile intermediare ale eliminării. Elementele lui U pot ocupa diagonala şi partea superioară a memoriei alocate lui

A, iar multiplicatorii lui L pot fi stocaţi sub diagonală. Diagonala lui L este 1 şi nu trebuie memorată. Această idee conduce la o factorizare *in-place*.

În varianta Doolittle, L are diagonala unitară. Presupunând cunoscute primele $k-1$ coloane relevante ale lui L şi primele $k-1$ linii ale lui U, elementele liniei k din U se obţin din $u_{kj} = a_{kj} - \sum_{s=1}^{k-1} l_{ks}u_{sj}$ pentru $j \geq k$. Apoi, pentru $i > k$, calculăm $l_{ik} = (a_{ik} - \sum_{s=1}^{k-1} l_{is}u_{sk})/u_{kk}$. Observaţi ordinea: mai întâi pivotul u_{kk} şi linia lui U, apoi coloana lui L.

Exemplu scurt

Pentru matricea din exemplul precedent, memoria finală poate conţine

$$\begin{pmatrix} 2 & 1 & 1 \\ 2 & -8 & -2 \\ -1 & -1 & 1 \end{pmatrix}.$$

Citită cu convenţia compactă, partea strict inferioară reprezintă multiplicatorii lui L, diagonala şi partea superioară reprezintă U, iar diagonala lui L este implicit egală cu 1.

Algoritm in-place

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o funcţie `LUc = lu_compact(A)` care suprascrie o copie a lui A. Interziceţi în cod construirea unei matrici elementare de eliminare. Scrieţi separat două funcţii care citesc factorii compactaţi şi rezolvă $Ly = b$, respectiv $Ux = y$, fără a extrage explicit L şi U.

Exerciţii şi verificarea înţelegerii

1. Derivaţi formulele Doolittle direct din egalitatea $a_{ij} = \sum_s l_{is}u_{sj}$.
2. Precizaţi care elemente trebuie deja cunoscute atunci când se calculează u_{kj} şi l_{ik}.
3. Ce problemă apare dacă $u_{kk} = 0$? Dar dacă $|u_{kk}|$ este foarte mic faţă de elementele coloanei? Legaţi răspunsul de secţiunea despre pivotare.
4. Estimaţi economia de memorie dintre păstrarea separată a lui A, L, U şi stocarea compactă.

Mini-proiect: profilarea accesului la memorie

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Comparaţi două implementări ale factorizării: una care actualizează matricea prin bucle scalar-cu-scalar şi una care foloseşte operaţii MATLAB pe blocuri/vectori. Pentru dimensiuni moderate, măsuraţi timpul şi discutaţi de ce numărul aritmetic de operaţii nu este

singurul factor care determină performanţa pe hardware modern.

Rezolvare orientativă

Citirea corectă a stocării compacte

Matricea compactă $C = \begin{pmatrix} 2 & 1 & 1 \\ 2 & -8 & -2 \\ -1 & -1 & 1 \end{pmatrix}$ nu reprezintă un nou operator. Convenţia este: U este partea triunghiulară superior a lui C, inclusiv diagonala, iar elementele strict sub diagonală sunt cele ale lui L; diagonala lui L este implicit 1. Rezultă exact factorii $L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -1 & 1 \end{pmatrix}$ şi $U = \begin{pmatrix} 2 & 1 & 1 \\ 0 & -8 & -2 \\ 0 & 0 & 1 \end{pmatrix}$.
Avantajul este că nu dublăm memoria. Dezavantajul este că programatorul trebuie să respecte riguros convenţia: aceeaşi poziţie de memorie are semnificaţii diferite în funcţie de poziţia faţă de diagonală.

Situaţie practică: rezolvare fără extragerea factorilor

La substituţia înainte, când avem nevoie de l_{ij} pentru $j < i$, citim direct C_{ij}; pentru diagonală folosim valoarea 1, nu C_{ii}. La substituţia înapoi, citim $u_{ij} = C_{ij}$ pentru $j \geq i$. Astfel, factorii nu sunt niciodată copiaţi. În aplicaţii mari, eliminarea copiilor temporare poate economisi memorie şi trafic de date, chiar dacă numărul de operaţii aritmetice rămâne acelaşi.

Exemplu şi observaţii despre stocarea in-place

În algoritmul LU cu L unitară, diagonala lui L nu trebuie stocată: ea este cunoscută a priori şi este formată numai din unităţi. În plus, multiplicatorii l_{ij} se află strict sub diagonală, în timp ce elementele lui U se află pe diagonală şi deasupra ei. Cele două mulţimi de poziţii nu se suprapun. De aceea aceeaşi matrice de n^2 numere poate conţine simultan informaţia ambilor factori.

Pentru matricea exemplului anterior, forma compactă este

$$\widehat{A} = \begin{pmatrix} 2 & 1 & 1 \\ 2 & -8 & -2 \\ -1 & -1 & 1 \end{pmatrix}.$$

Partea strict inferior triunghiulară reprezintă multiplicatorii din L, iar partea superior triunghiulară, inclusiv diagonala, este U. Citirea corectă este deci $L = I + \mathrm{tril}(\widehat{A}, -1)$ şi $U = \mathrm{triu}(\widehat{A})$.

De ce contează stocarea?

Pentru o matrice densă de ordin $n = 20000$, $n^2 = 4 \cdot 10^8$ numere. În double precision aceasta înseamnă aproximativ 3.2 GB numai pentru o singură matrice. Dublarea inutilă a memoriei poate transforma un algoritm corect într-unul imposibil de rulat. Complexitatea de memorie este o parte a analizei algoritmului, nu doar un detaliu de programare.

Sarcină practică: factorizare in-place

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Modificaţi implementarea LU astfel încât să nu creeze matrici separate L şi U în timpul eliminării. Suprascrieţi elementele eliminate cu multiplicatorii corespunzători. Scrieţi apoi două funcţii auxiliare care extrag factorii numai pentru verificare. Măsuraţi memoria folosită de versiunea cu trei matrici (A, L, U) şi de versiunea in-place pentru dimensiuni la care diferenţa devine vizibilă.

Întrebări şi exerciţii

1. Pentru forma compactă de mai sus, reconstruiţi L şi U fără a efectua din nou eliminarea.
2. Explicaţi de ce diagonala lui L nu poate fi folosită pentru a stoca altă informaţie dacă programul trebuie să permită şi factorizări cu o convenţie diferită de normalizare.
3. Cum se modifică interpretarea matricei compacte atunci când folosim pivotare parţială? Ce informaţie suplimentară trebuie stocată?
4. Estimaţi memoria, în MB, pentru o matrice 10000×10000 în precizie double. Comparaţi stocarea unei singure matrice cu stocarea separată a lui A, L, U.
5. Scrieţi pseudocodul unei funcţii care rezolvă $Ax = b$ folosind direct matricea compactă, fără a construi explicit L şi U.

3.4 Factorizarea LDM^T

Factorizarea LDM^T

Este posibil să se conceapă şi alte tipuri de factorizări ale lui A.

Mai exact, vom aborda unele variante în care factorizarea lui A este de forma

$$A = L\,D\,M^T, \tag{3.17}$$

unde L, M^T şi D sunt matrici inferior triunghiulare, superior triunghiulare şi, respectiv, diagonale.

După construirea acestei factorizări, rezolvarea sistemului se poate realiza rezolvând mai întâi sistemul inferior triunghiulară $Ly = b$, apoi cel diagonal $Dz = y$ şi în final sistemul superior triunghiulară $M^T x = z$, cu un cost de aproximativ $2n^2$ operaţii (mai precis, $2n^2 - n$ în convenţia în care diagonalele lui L şi M sunt unitare).

În cazul simetric, obţinem $M = L$, iar factorizarea LDL^T poate fi calculată cu jumătate din cost, după cum vom vedea când vom vorbi despre cazul matricelor simetrice. Factorizarea LDM^T se bucură de o proprietate analogă cu cea pentru factorizarea LU. În particular, se aplică următorul rezultat.

Teoremă 3.4.1. *Dacă toţi minorii principali ai unei matrice $A \in \mathbb{R}^{n\times n}$ sunt nenuli, atunci*

există o matrice diagonală unică D, o matrice triunghiulară inferioară unitară unică[1] *L şi o matrice triunghiulară superioară unitară unică M^T, astfel încât $A = LDM^T$.*

Demonstraţie. Ipoteza spune în particular că toţi minorii principali conducători de ordine $1, \dots, n$ sunt nenuli. Teorema anterioară furnizează factorizarea unică $A = LU$, cu L inferior triunghiulară unitară şi U superior triunghiulară. Deoarece $\det A \neq 0$, toate elementele diagonale u_{ii} sunt nenule. Definim

$$D = \operatorname{diag}(u_{11}, \dots, u_{nn}), \qquad M^T = D^{-1}U.$$

Atunci M^T este triunghiulară superioară unitară şi $A = LDM^T$, ceea ce dovedeşte existenţa.

Pentru unicitate, presupunem $A = L_1 D_1 M_1^T = L_2 D_2 M_2^T$. Produsele $D_i M_i^T$ sunt superior triunghiulare, deci am obţinut două factorizări LU cu factor inferior unitar. Unicitatea factorizării LU dă $L_1 = L_2$ şi $D_1 M_1^T = D_2 M_2^T$. Comparând diagonalele rezultă $D_1 = D_2$; cum diagonalele sunt nesingulare, urmează $M_1^T = M_2^T$. □

Completare didactică: separarea scalării de transformările triunghiulare

Factorizarea $A = LDM^T$, cu L şi M triunghiulare inferior cu diagonală unitară şi D diagonală, separă în mod explicit pivoturile de multiplicatori. Din punct de vedere conceptual, L şi M descriu eliminările, iar D păstrează scalările asociate pivoturilor. Pentru matrice simetrice, atunci când factorizarea se poate efectua fără pivotare, simetria conduce la $M = L$, deci la forma $A = LDL^T$.

Exemplu 2×2

Pentru $A = \begin{pmatrix} 2 & 4 \\ 6 & 13 \end{pmatrix}$ avem

$$L = \begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix}, \qquad D = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}, \qquad M^T = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}.$$

Înmulţind factorii obţinem $LDM^T = A$. În general, pentru $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ cu $a \neq 0$, primul pivot este $d_1 = a$, multiplicatorii sunt $l_{21} = c/a$ şi $m_{21} = b/a$, iar al doilea element diagonal este $d_2 = d - cb/a$.

Legătura cu complementul Schur

Cantitatea $d - cb/a$ este complementul Schur al blocului scalar a. Aceeaşi structură apare în eliminarea pe blocuri, în sisteme saddle-point, optimizare constrânsă şi metode de domeniu. Factorizările elementare studiate aici sunt astfel prototipuri pentru algoritmi mult mai mari.

[1] Noi numim matrice triunghiulară unitară o matrice triunghiulară care are intrările diagonale egale cu 1.

Laborator / algoritm

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi o factorizare LDL^T pentru matrice simetrice pentru care pivotarea nu este necesară. Verificaţi separat norma lui $A - LDL^T$ şi comparaţi rezultatul cu factorizarea Cholesky atunci când A este pozitiv definită. Observaţi că LDL^T nu necesită extragerea rădăcinilor pătrate.

Exerciţii şi verificarea înţelegerii

1. Derivaţi factorizarea LDL^T a matricei $\begin{pmatrix} 4 & 2 \\ 2 & 3 \end{pmatrix}$.
2. Arătaţi că dacă $A = LDL^T$ cu L inversabilă, inerţia formei pătratice este determinată de semnele elementelor lui D.
3. Explicaţi de ce pentru o matrice simetrică pozitiv definită elementele diagonale din D sunt pozitive.
4. Comparaţi paşii necesari pentru rezolvarea lui $Ax = b$ folosind LDL^T cu cei necesari pentru LU.

Mini-proiect: matrice simetrică, dar nu pozitiv definită

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi o familie de matrice simetrice cu valori proprii atât pozitive cât şi negative. Investigaţi când o implementare LDL^T fără pivotare eşuează şi comparaţi cu rutina MATLAB destinată matricelor generale. Scopul proiectului nu este reproducerea unei strategii sofisticate de pivotare simetrică, ci identificarea motivului pentru care simetria singură nu garantează siguranţa algoritmului.

Rezolvare orientativă

Factorizarea LDL^T a unei matrice SPD

Pentru $A = \begin{pmatrix} 4 & 2 \\ 2 & 3 \end{pmatrix}$ căutăm $L = \begin{pmatrix} 1 & 0 \\ \ell & 1 \end{pmatrix}$ şi $D = \operatorname{diag}(d_1, d_2)$. Din elementul $(1,1)$ obţinem $d_1 = 4$. Din elementul $(2,1)$ rezultă $\ell d_1 = 2$, deci $\ell = 1/2$. În final, elementul $(2,2)$ dă $\ell^2 d_1 + d_2 = 3$, adică $1 + d_2 = 3$, deci $d_2 = 2$. Prin urmare $A = \begin{pmatrix} 1 & 0 \\ 1/2 & 1 \end{pmatrix} \operatorname{diag}(4,2) \begin{pmatrix} 1 & 1/2 \\ 0 & 1 \end{pmatrix}$.

Situaţie practică: detectarea semnului formei pătratice

Dacă o factorizare simetrică produce D cu elemente de semne diferite, forma pătratică nu poate fi pozitiv definită. Într-o problemă de optimizare, aceasta poate semnala că Hessiana are direcţii de curbură negativă şi punctul curent nu este minim strict. Astfel, factorul diagonal nu este doar un obiect de calcul, ci şi un instrument de diagnostic structural.

Exemplu rezolvat: factorizarea LDM^T pentru o matrice nesimetrică

Considerăm

$$A = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix}.$$

Căutăm $L = \begin{pmatrix} 1 & 0 \\ \ell & 1 \end{pmatrix}$, $D = \operatorname{diag}(d_1, d_2)$ şi $M^T = \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix}$. Din produsul

$$LDM^T = \begin{pmatrix} d_1 & d_1 m \\ \ell d_1 & \ell d_1 m + d_2 \end{pmatrix}$$

identificăm succesiv $d_1 = 2$, $m = 1/2$, $\ell = 3/2$ şi $d_2 = 4 - (3/2)\cdot 2 \cdot (1/2) = 5/2$. Aşadar

$$A = \begin{pmatrix} 1 & 0 \\ 3/2 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 5/2 \end{pmatrix} \begin{pmatrix} 1 & 1/2 \\ 0 & 1 \end{pmatrix}.$$

Factorizarea separă transformările triunghiulare de scalările diagonale. În cazul simetric, factorii din stânga şi dreapta se leagă între ei şi ajungem la forma LDL^T.

Sarcină de implementare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pornind de la o implementare LU, scrieţi o rutină care normalizează diagonala factorului superior şi construieşte LDM^T. Verificaţi numeric identitatea $A = LDM^T$ comparând elementele celor două matrice şi raportând cea mai mare diferenţă în valoare absolută. Pentru matrici simetrice pentru care factorizarea există fără pivotare, verificaţi experimental că factorii triunghiulari unitari din stânga şi dreapta coincid până la erorile de rotunjire.

Exerciţii suplimentare

1. Calculaţi LDM^T pentru $A = \begin{pmatrix} 4 & 2 \\ 1 & 3 \end{pmatrix}$.
2. Arătaţi că $\det A = \prod_i d_i$ atunci când L şi M sunt unitare.
3. Pentru o matrice simetrică, deduceţi din unicitatea factorizării că $M = L$ în condiţiile corespunzătoare de existenţă.
4. Explicaţi ce problemă apare dacă un element d_i este zero înainte de ultimul pas.
5. Comparaţi conceptual LDL^T cu Cholesky: când este preferabil să nu extragem

rădăcini pătrate?

3.5 Pivotare

Pivotare

După cum s-a subliniat anterior, procesul GEM se întrerupe imediat ce se calculează o intrare pivotală zero. Într-un astfel de caz, trebuie să se apeleze la aşa-numita tehnică de pivotare, care constă în schimbarea rândurilor (sau a coloanelor) din sistem astfel încât să se obţină pivoţi nenuli.

Strategia de pivotare adoptată până în prezent poate fi generalizată prin căutarea, la fiecare pas k al procedurii de eliminare, a unei intrări pivotante care nu este nulă, căutând în interiorul intrărilor din subcoloana $A^{(k)}(k:n,k)$. Din acest motiv, se numeşte pivotare parţială (pe rânduri).

Se poate observa că o valoare mare a lui $m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, i = k+1, \cdots, n$ (generată, de exemplu, de o valoare mică a pivotului $a_{kk}^{(k)}$) poate amplifica erorile de rotunjire care afectează intrările $a_{kj}^{(k)}$. Prin urmare, pentru a asigura o mai bună stabilitate, la pasul k alegem $r \in \{k, \ldots, n\}$ astfel încât

$$|a_{rk}^{(k)}| = \max_{j=k,\ldots,n} |a_{jk}^{(k)}|,$$

iar $a_{rk}^{(k)}$ devine pivotul după schimbarea liniilor k şi r. În general, pivotarea parţială se efectuează la fiecare etapă, chiar dacă pivotul curent este deja nenul.

Alternativ, procesul de căutare ar fi putut fi extins la întreaga submatrice $A^{(k)}(k:n,k:n)$, finalizându-se cu o pivotare completă.

Observaţi, totuşi, că pivotarea parţială necesită un cost suplimentar de ordin n^2 comparaţii, în timp ce pivotarea completă necesită aproximativ $n^3/3$ comparaţii. Acest cost este de acelaşi ordin cu cel aritmetic al eliminării, motiv pentru care în practică se preferă aproape întotdeauna pivotarea parţială.

Matrice de permutare

Schimbul dintre a i-a şi j-a linie a unei matrice; acest lucru se poate face prin înmulţirea prealabilă a lui A cu matricea $P^{(i,j)}$ de elemente

$$p_{rs}^{(i,j)} = \begin{cases} 1 & \text{dacă } r = s = 1, \cdots, i-1, i+1, \cdots j-1, j+1, \cdots n, \\ 1 & \text{dacă } r = j, s = i \text{ or } r = i, s = j, \\ 0, & \text{în caz contrar.} \end{cases} \tag{3.18}$$

Matricele de tipul $P^{(i,j)}$ se numesc matrice de permutare elementară.

Produsul matricelor de permutare elementară se numeşte matrice de permutare şi efectuează schimburile de rânduri asociate fiecărei matrice de permutare elementară.

În practică, o matrice de permutare este o reordonare pe rânduri a matricei identitate.

Să analizăm modul în care pivotarea parţială afectează factorizarea LU indusă de GEM.

La prima etapă a GEM cu pivotare parţială, după ce se află intrarea a_{r1} de modul maxim din prima coloană, se construieşte matricea elementară de permutare P_1 care schimbă prima linie cu a r-a linie (dacă $r = 1$, P_1 este matricea identitate).

În continuare, se generează prima matrice de transformare gaussiană M_1 şi se stabileşte

$$A^{(2)} = M_1 P_1 A^{(1)}. \tag{3.19}$$

O abordare similară se face acum pentru $A^{(2)}$, căutând o nouă matrice de permutare P_2 şi o nouă matrice M_2 astfel încât

$$A^{(3)} = M_2 P_2 A^{(2)} = M_2 P_2 M_1 P_1 A^{(1)}. \tag{3.20}$$

Executând toate etapele de eliminare, matricea superior triunghiulară U rezultată este acum dată de

$$U = A(n) = \underbrace{M_{n-1} P_{n-1} \cdots M_2 P_2 M_1 P_1}_{:=M} A^{(1)}. \tag{3.21}$$

Reţineţi că $m_{ik}^{(k)}$ utilizat în construcţia lui M_k este acum $m_{ik}^{(k)} = \frac{b_{ik}^{(k)}}{b_{kk}^{(k)}}$, unde $b_{ik}^{(k)}$ sunt intrările matricei $P_k A^{(k)}$.

Obţinem că $U = M\,A$ şi, astfel, $U = (MP^{-1})PA$, unde $P = P_{n-1} \cdots P_1$. Afirmăm că $L = PM^{-1}$ este triunghiulară inferioară unitară.

Nu trebuie să fim îngrijoraţi de prezenţa inversului lui M, deoarece $M^{-1} = P_1^{-1} M_1^{-1} \cdots P_{n-1}^{-1} M_{n-1}^{-1}$ şi $P_i^{-1} = P_i^T$, în timp ce $M_i^{-1} = 2I_n - M_i = I_n + m_i\, e_i^T$.

Prin urmare, avem

$$\begin{aligned} L &= P_{n-1} \cdots P_2 P_1 P_1^{-1} (I_n + m_1\, e_1^T) P_2^{-1} (I_n + m_2\, e_2^T) \cdots P_{n-1}^{-1} (I_n + m_{n-1}\, e_{n-1}^T) \\ &= P_{n-1} \cdots P_2 (I_n + m_1\, e_1^T) P_2^{-1} (I_n + m_2\, e_2^T) \cdots P_{n-1}^{-1} (I_n + m_{n-1}\, e_{n-1}^T). \end{aligned}$$

Să discutăm acum

$$\begin{aligned} &P_{n-1} \cdots P_2 (I_n + m_1\, e_1^T) \\ &= (P_{n-1} \cdots P_2 + P_{n-1} \cdots P_2 m_1\, e_1^T P_2^T \cdots P_{n-1}^T P_{n-1} \cdots P_2) \\ &= (P_{n-1} \cdots P_2 + P_{n-1} \cdots P_2 m_1\, e_1^T (P_{n-1} \cdots P_2)^T P_{n-1} \cdots P_2) \\ &= (P_{n-1} \cdots P_2 + P_{n-1} \cdots P_2 m_1 (P_{n-1} \cdots P_2 e_1)^T P_{n-1} \cdots P_2) \\ &= [I_n + P_{n-1} \cdots P_2 m_1 (P_{n-1} \cdots P_2 e_1)^T] P_{n-1} \cdots P_2. \end{aligned} \tag{3.22}$$

Vom avea

$$\begin{aligned} &P_{n-1} \cdots P_2 (I_n + m_1\, e_1^T) \\ &= [I_n + \underbrace{P_{n-1} \cdots P_2 m_1}_{:=\widetilde{m}_1} (P_{n-1} \cdots P_2 e_1)^T] P_{n-1} \cdots P_2. \end{aligned} \tag{3.23}$$

Dar permutarea $P_{n-1} \cdots P_2$ permută doar intrările de la 2 la n dintr-un vector; intrările 1 rămân neatinse. Aceasta înseamnă că prima intrare a lui $\widetilde{m}_1$ este încă zero şi e_1 rămâne neschimbat sub această permutare, adică $P_{n-1} \cdots P_2 e_1 = e_1$.

Astfel, avem de fapt

$$\begin{aligned} &P_{n-1} \cdots P_2 (I_n + m_1\, e_1^T) \\ &= \underbrace{[I_n + \widetilde{m}_1 e_1^T]}_{\text{inferior triunghiulară}} \; P_{n-1} \cdots P_2 \end{aligned} \tag{3.24}$$

şi

$$L = \underbrace{[I_n + \widetilde{m}_1 e_1^T]}_{\text{inferior triunghiulară}} \; P_{n-1} \cdots P_2 P_2^{-1}(I_n + m_2\, e_2^T) \cdots P_{n-1}^{-1}(I_n + m_{n-1}\, e_{n-1}^T).$$

Repetând argumentul avem că L este triunghiulară inferioară.

Deoarece $L = PM^{-1}$ este triunghiulară inferioară unitară, factorizarea LU se citeşte

$$PA = LU. \tag{3.25}$$

Odată ce L, U şi P sunt disponibile, rezolvarea sistemului liniar iniţial se rezumă la rezolvarea sistemelor triunghiulare $Ly = Pb$ şi $Ux = y$.

Teoremă 3.5.1. *Fie $A \in \mathbb{R}^{n\times n}$ o matrice nesingulară. Atunci există o matrice de permutare P astfel încât $P\,A = L\,U$, unde L şi U sunt, respectiv, o matrice triunghiulară inferioară şi una triunghiulară superioară, obţinute prin eliminarea gaussiană.*

Demonstraţie. Demonstraţia este deja făcută, cu excepţia faptului că la orice pas avem

$$\max \operatorname{abs}(A^{(k)}(k:n,k)) \neq 0. \tag{3.26}$$

Dacă ar fi posibil să avem

$$\max \operatorname{abs}(A^{(k)}(k:n,k)) = 0, \tag{3.27}$$

atunci $\det A = 0$ ceea ce este evitat de ipoteză. □

Dacă se realizează pivotarea completă, la primul pas al procesului, odată găsit elementul a_{qr} al celui mai mare modul din submatricea $A(1:n, 1:n)$, trebuie să schimbăm prima linie şi prima coloană cu a q-a linie şi a r-a coloană. Se generează astfel matricea $P_1 A^{(1)} Q_1$, unde P_1 şi Q_1 sunt matrici de permutare pe rânduri şi, respectiv, pe coloane.

În consecinţă, acţiunea matricei M_1 este acum astfel încât $A^{(2)} = M_1 P_1 A^{(1)} Q_1$. Repetând procesul, la ultima etapă, obţinem

$$U = A^{(n)} = M_{n-1} P_{n-1} \cdots M_1 P_1 A^{(1)} Q_1 \cdots Q_{n-1}. \tag{3.28}$$

În cazul pivotării complete, factorizarea LU devine

$$PAQ = LU, \tag{3.29}$$

unde $Q = Q_1 \cdots Q_{n-1}$ este o matrice de permutare care ţine cont de toate permutările care au fost operate. Prin construcţie, matricea L este tot inferior triunghiulară, cu intrări de modul mai mici sau egale cu 1.

Calculul inversei unei matrice

Calculul explicit al inversei unei matrice poate fi efectuat folosind factorizarea LU după cum urmează.

Notând cu X inversa unei matrice nesingulare în $\mathbb{R}^{n\times n}$, vectorii coloană ai lui X sunt soluţiile sistemelor liniare $A\,x_i = e_i$, pentru $i = 1, \cdots, n$.

Presupunând că $PA = LU$, unde P este matricea de permutare cu pivotare parţială, trebuie să rezolvăm $2n$ sisteme triunghiulare de forma $Ly_i = Pe_i$, $U\,x_i = y_i$, $i = 1, \cdots, n$, adică o succesiune de sisteme liniare care au aceeaşi matrice de coeficienţi, dar părţi drepte diferite.

Aplicarea factorizării LU într-o problemă de deformare elastică 1D

Se consideră o bară elastică de lungime $L = 1\,\text{m}$, cu capetele menţinute fixe

$$u(0) = u(1) = 0.$$

Ecuaţia care descrie deformarea barei este de tip Poisson:

$$-k\,u''(x) = q(x), \qquad 0 < x < 1,$$

unde u este deplasarea, k este un coeficient de elasticitate, iar $q(x)$ este forţa care acţionează asupra barei.

Aproximarea numerică a derivatei a doua

Pentru fiecare nod interior x_i, a doua derivată $u''(x_i)$ se poate aproxima prin diferenţe finite centrale.

Pornim de la seriile Taylor în jurul nodului x_i:

$$u(x_i + h) = u(x_i) + hu'(x_i) + \frac{h^2}{2}u''(x_i) + \frac{h^3}{6}u'''(x_i) + O(h^4),$$

$$u(x_i - h) = u(x_i) - hu'(x_i) + \frac{h^2}{2}u''(x_i) - \frac{h^3}{6}u'''(x_i) + O(h^4).$$

Adunând cele două ecuaţii, derivata întâi se elimină:

$$u(x_i + h) + u(x_i - h) = 2u(x_i) + h^2u''(x_i) + O(h^4),$$

de unde rezultă formula de diferenţe finite centrale:

$$\boxed{u''(x_i) \approx \frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2}}.$$

Această formulă stă la baza discretizării ecuaţiei Poisson pentru fiecare nod interior.

Discretizare numerică a ecuaţiei Poisson

Alegem $N = 4$ noduri interioare, deci $N + 1 = 5$ subintervale egale de lungime $h = L/(N + 1) = 0.2\,\text{m}$, şi notăm deplasările în nodurile $x_i = ih$ prin u_i, $i = 1, \ldots, 4$. Nodurile de frontieră sunt $x_0 = 0$ şi $x_5 = L$, cu $u_0 = u_5 = 0$.

Înlocuind aproximaţia derivatei a doua în ecuaţia Poisson:

$$-\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = \frac{q(x_i)}{k}, \quad i = 1, \ldots, 4,$$

sau echivalent:

$$2u_i - u_{i-1} - u_{i+1} = h^2\frac{q(x_i)}{k}.$$

Astfel se generează un sistem liniar tridiagonal:

$$A \cdot \mathbf{u} = \mathbf{b},$$

unde

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} h^2 q(x_1)/k \\ h^2 q(x_2)/k \\ h^2 q(x_3)/k \\ h^2 q(x_4)/k \end{bmatrix}.$$

Factorizarea LU

Observaţie: valoarea deplasării în fiecare nod interior depinde doar de nodurile vecine, ceea ce face sistemul tridiagonal şi foarte potrivit pentru factorizarea LU. Pentru rezolvarea eficientă, factorăm matricea A ca:

$$A = L \cdot U,$$

unde L este triunghiulară inferioară, iar U este triunghiulară superioară:

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & 0 \\ 0 & -\frac{2}{3} & 1 & 0 \\ 0 & 0 & -\frac{3}{4} & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} 2 & -1 & 0 & 0 \\ 0 & \frac{3}{2} & -1 & 0 \\ 0 & 0 & \frac{4}{3} & -1 \\ 0 & 0 & 0 & \frac{5}{4} \end{bmatrix}.$$

Matrice cu structură de bandă

Metodele de discretizare pentru problemele cu valori la frontieră conduc adesea la rezolvarea sistemelor liniare cu matrici care au forme de bandă, bloc sau rare. Exploatarea structurii matricei permite o reducere drastică a costurilor de calcul ale factorizării şi ale algoritmilor de substituţie.

Vom aborda variante speciale ale factorizării GEM sau LU care sunt concepute în mod corespunzător pentru a trata matrici de acest tip.

Completare didactică: pivotarea ca mecanism de stabilizare

Existenţa unei factorizări algebrice nu este suficientă pentru un algoritm numeric robust. Dacă pivotul folosit la un pas este foarte mic, multiplicatorii de eliminare pot deveni foarte mari, iar erorile de rotunjire se pot amplifica. Pivotarea parţială alege, în coloana curentă, un element de modul maxim şi schimbă liniile astfel încât acel element să devină pivot.

Exemplul clasic al unui pivot periculos

Considerăm

$$A = \begin{pmatrix} 10^{-16} & 1 \\ 1 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 + 10^{-16} \\ 2 \end{pmatrix},$$

care are soluţia exactă $x = (1, 1)^T$. Fără pivotare, primul multiplicator este 10^{16}. În aritmetică finită, operaţia de eliminare combină numere de scări foarte diferite şi poate pierde cifre semnificative. Dacă schimbăm cele două linii, primul pivot devine 1, multiplicatorul are modul 10^{-16}, iar calculul este mult mai benign.

Cu pivotare parţială, factorizarea se scrie $PA = LU$, unde P este o matrice de permutare. Pentru rezolvarea sistemului trebuie deci folosit termenul liber permutat: mai întâi rezolvăm $Ly = Pb$, apoi $Ux = y$.

De ce pivotăm?

În aritmetica de calculator, împărţirea la un pivot foarte mic poate produce multiplicatori foarte mari şi poate amplifica efectul rotunjirilor. Pivotarea încearcă să evite asemenea alegeri nefavorabile. Analiza cantitativă a sensibilităţii problemei şi a erorilor algoritmului va fi făcută abia în capitolul următor.

Laborator: cu şi fără pivotare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi două versiuni ale eliminării gaussiene: una fără pivotare şi una cu pivotare parţială. Testaţi familia $A_\varepsilon = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix}$ pentru $\varepsilon = 10^{-2}, 10^{-4}, \ldots, 10^{-16}$. Folosiţi o soluţie exactă prestabilită şi reprezentaţi eroarea relativă în funcţie de ε. Raportaţi şi mărimea maximă a multiplicatorilor.

Exerciţii şi verificarea înţelegerii

1. Scrieţi matricea de permutare care schimbă liniile 1 şi 3 ale unei matrice 4×4 şi verificaţi că $P^{-1} = P^T$.
2. Efectuaţi un pas de pivotare parţială pentru matricea $\begin{pmatrix} 0.01 & 2 & 1 \\ 3 & 1 & -1 \\ 2 & 4 & 1 \end{pmatrix}$.
3. Explicaţi de ce pivotarea parţială garantează multiplicatori de modul cel mult 1 la pasul curent.
4. Daţi un exemplu în care pivotarea este necesară algebric deoarece pivotul curent este exact zero, deşi matricea este inversabilă.
5. De ce o matrice de permutare nu modifică norma euclidiană a unui vector?

Mini-proiect: factor de creştere

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Pentru matrice aleatoare şi pentru câteva matrice construite deliberat, calculaţi raportul dintre cel mai mare element în modul apărut în factorul U şi cel mai mare element în modul al matricei iniţiale. Comparaţi eliminarea fără pivotare cu pivotarea parţială şi urmăriţi cum se modifică multiplicatorii şi elementele intermediare.

Rezolvare orientativă

Permutarea unui pivot nul

Pentru $A = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$ eliminarea fără pivotare nu poate începe, deşi $\det A = -2 \neq 0$. Schimbăm liniile cu $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ şi obţinem $PA = \begin{pmatrix} 2 & 3 \\ 0 & 1 \end{pmatrix}$. În acest caz $L = I$ şi $U = PA$. Pentru sistemul $Ax = b$, termenul liber trebuie schimbat în aceeaşi ordine: $PAx = Pb$. Acest exemplu arată motivul algebric minimal al pivotării. Exemplul cu 10^{-16} arată motivul numeric: uneori pivotul nu este zero, dar este atât de mic încât folosirea lui este riscantă.

Situaţie practică: alegerea pivotului şi unităţile de măsură

Dacă liniile provin din ecuaţii exprimate în unităţi foarte diferite, elementul cu modul maxim poate reflecta mai ales scala ecuaţiei. O scalare preliminară poate face comparaţia pivoturilor mai semnificativă. Totuşi, după scalare trebuie urmărită cu atenţie problema transformată; efectul scalării asupra sensibilităţii va fi analizat sistematic în capitolul următor.

Exemplu rezolvat: pivotarea şi efectul scării ecuaţiilor

Considerăm matricea

$$A = \begin{pmatrix} 10^{3} & 10^{6} \\ 1 & 1 \end{pmatrix}.$$

Pivotarea parţială simplă compară $|10^3|$ şi $|1|$ în prima coloană şi păstrează prima linie. Totuşi, raportat la cel mai mare element al fiecărei linii, primul candidat are mărimea relativă $10^3/10^6 = 10^{-3}$, iar al doilea are $1/1 = 1$. Acest exemplu motivează pivotarea parţială scalată: înainte de alegerea pivotului, comparăm $|a_{ik}|/s_i$, unde $s_i = \max_j |a_{ij}|$.

Nu trebuie însă confundate două idei. Pivotarea scalată poate evita o alegere nefavorabilă produsă numai de unităţile de măsură. Dacă două coloane sunt aproape dependente liniar, apar în continuare dificultăţi numerice; acestea vor fi cuantificate în capitolul despre analiza erorii.

Ce trebuie stocat în practică?

Într-o implementare eficientă nu construim de obicei matricea de permutare P. Stocăm un vector de indici p care descrie schimburile de linii. Aplicarea lui P asupra unui vector b înseamnă doar reordonarea componentelor lui b conform lui p. Aceasta economiseşte memorie şi evită operaţii cu o matrice care conţine în principal zerouri.

Sarcină de implementare: vector de pivotare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Extindeţi factorizarea in-place astfel încât, la pasul k, să găsească indicele $p = \operatorname{argmax}_{i \geq k} |a_{ik}|$, să schimbe liniile şi să actualizeze un vector de permutare. Funcţia trebuie să întoarcă matricea compactă şi vectorul p. Scrieţi apoi `luPivotSolve`, care

aplică permutarea lui b şi efectuează cele două substituţii. Verificaţi relaţia $PA \approx LU$ fără a construi neapărat P: comparaţi $A(p,:)$ cu $L * U$.

Exerciţii suplimentare

1. Pentru $A_\varepsilon = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix}$ şi $\varepsilon = 10^{-12}$, efectuaţi primul pas cu şi fără pivotare. Comparaţi multiplicatorii.
2. Arătaţi că o schimbare de linii este o transformare ortogonală: $P^T P = I$. Ce consecinţă are pentru norma euclidiană?
3. Explicaţi diferenţa dintre pivotare parţială, pivotare totală şi pivotare parţială scalată.
4. De ce pivotarea totală necesită şi memorarea permutărilor de coloane? Cum se recuperează ordinea necunoscutelor?
5. Construiţi un experiment în care rescalarea unei singure ecuaţii schimbă pivotul ales de pivotarea parţială, dar nu schimbă soluţia exactă a sistemului.

3.6 Factorizarea Cholesky

Matrice simetrice pozitiv definite: Factorizarea Cholesky

În continuare folosim abrevierea **SPD** (din engleză, *symmetric positive definite*) pentru o matrice **simetrică pozitiv definită**. Pentru factorizarea Cholesky vom folosi în text convenţia $A = H^T H$, cu H superior triunghiular. Forma echivalentă $A = LL^T$, cu $L = H^T$ inferior triunghiular, este folosită în unele exemple şi exerciţii; cele două convenţii descriu aceeaşi factorizare.

După cum s-a arătat deja, factorizarea LDM^T se simplifică considerabil atunci când A este simetrică, deoarece într-un astfel de caz $M = L$, obţinându-se aşa-numita factorizare LDL^T. Costul de calcul se înjumătăţeşte, faţă de factorizarea LU, la aproximativ $(n^3/3)$ flop-uri.

Factorizarea Cholesky

Teoremă 3.6.1 (Factorizarea Cholesky)**.** *Fie $A \in \mathbb{R}^{n\times n}$ simetrică şi pozitiv definită. Atunci există o unică matrice triunghiulară superioară H, cu elemente diagonale strict pozitive, astfel încât*

$$A = H^T H.$$

Demonstraţie. Demonstrăm existenţa prin inducţie după n. Pentru $n = 1$, avem $A = (a_{11})$ cu $a_{11} > 0$, iar $H = (\sqrt{a_{11}})$.

Presupunem rezultatul adevărat pentru matrici de ordin $n-1$ şi scriem

$$A = \begin{pmatrix} B & c \\ c^T & a \end{pmatrix},$$

unde $B \in \mathbb{R}^{(n-1)\times(n-1)}$. Orice submatrice principală a unei matrice SPD este SPD, deci B este SPD. Prin ipoteza de inducţie există o unică matrice triunghiulară superioară R, cu diagonală pozitivă, pentru care $B = R^T R$.

Căutăm H sub forma

$$H = \begin{pmatrix} R & h \\ 0^T & \beta \end{pmatrix}, \qquad \beta > 0.$$

Atunci

$$H^T H = \begin{pmatrix} R^T R & R^T h \\ h^T R & h^T h + \beta^2 \end{pmatrix}.$$

Identificarea cu A impune $R^T h = c$, deci $h = R^{-T} c$, şi $\beta^2 = a - h^T h$. Trebuie să arătăm că această ultimă cantitate este strict pozitivă. Deoarece $B = R^T R$, avem $h^T h = c^T B^{-1} c$. Pentru vectorul nenul $z = (-B^{-1}c, 1)^T$ rezultă

$$0 < z^T A z = a - c^T B^{-1} c = a - h^T h.$$

Aşadar $\beta = \sqrt{a - h^T h}$ este real şi strict pozitiv, iar existenţa este demonstrată.

Pentru unicitate, presupunem $A = H_1^T H_1 = H_2^T H_2$, unde H_1, H_2 sunt superior triunghiulare cu diagonală pozitivă. Atunci

$$Q := H_1 H_2^{-1}$$

este triunghiulară superioară şi verifică $Q^T Q = I$, deci este ortogonală. O matrice ortogonală triunghiulară superioară este diagonală, cu elemente diagonale egale cu ± 1. Deoarece diagonalele lui H_1 şi H_2 sunt pozitive, diagonala lui Q este pozitivă; prin urmare $Q = I$ şi $H_1 = H_2$. □

De ce apar radicali reali în algoritm?

La fiecare pas, expresia de sub radical este un complement Schur al unei submatrice principale SPD. Pozitiva definire nu este doar o ipoteză abstractă: ea garantează exact că pivotul Cholesky este strict pozitiv şi că algoritmul poate continua fără pivotare.

Calculul elementelor factorului Cholesky

Teoremă 3.6.2 (Formulele recursive Cholesky)**.** *Fie $A = (a_{ij})$ SPD şi $A = H^T H$, unde $H = (h_{ij})$ este triunghiulară superioară cu $h_{jj} > 0$. Atunci, pentru $j = 1, \ldots, n$,*

$$h_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} h_{kj}^2},$$

iar, după calculul lui h_{jj}, pentru $i = j+1, \ldots, n$,

$$h_{ji} = \frac{1}{h_{jj}} \left(a_{ji} - \sum_{k=1}^{j-1} h_{kj} h_{ki} \right).$$

În particular, $h_{11} = \sqrt{a_{11}}$.

Demonstraţie. Din $A = H^T H$, pentru $j \le i$ avem

$$a_{ji} = (H^T H)_{ji} = \sum_{k=1}^{j} h_{kj} h_{ki},$$

deoarece $h_{kj} = 0$ pentru $k > j$. Dacă $i = j$, obţinem

$$a_{jj} = \sum_{k=1}^{j-1} h_{kj}^2 + h_{jj}^2.$$

Cum $h_{jj} > 0$, rezultă formula pentru h_{jj}. Dacă $i > j$, separăm ultimul termen al sumei,

$$a_{ji} = \sum_{k=1}^{j-1} h_{kj} h_{ki} + h_{jj} h_{ji},$$

şi, deoarece $h_{jj} > 0$, izolarea lui h_{ji} dă formula a doua. Ordinea de calcul este astfel determinată: coloanele (sau, echivalent, liniile factorului superior) se obţin succesiv, iar fiecare formulă foloseşte numai elemente deja calculate. □

Un exemplu Cholesky făcut cu pixul pe hârtie

Pentru

$$A = \begin{pmatrix} 4 & 2 & 2 \\ 2 & 5 & 1 \\ 2 & 1 & 3 \end{pmatrix}$$

avem $h_{11} = 2$, $h_{12} = 1$, $h_{13} = 1$. Apoi $h_{22} = \sqrt{5-1} = 2$ şi $h_{23} = (1 - 1 \cdot 1)/2 = 0$. În final, $h_{33} = \sqrt{3 - 1^2 - 0^2} = \sqrt{2}$. Prin urmare

$$H = \begin{pmatrix} 2 & 1 & 1 \\ 0 & 2 & 0 \\ 0 & 0 & \sqrt{2} \end{pmatrix}, \qquad H^T H = A.$$

Studentul trebuie să poată explica, pentru fiecare dintre cele şase numere calculate, din ce egalitate $a_{ij} = (H^T H)_{ij}$ provine formula folosită.

Când folosim Cholesky?

Factorizarea Cholesky este construită pentru matrici simetrice pozitiv definite. Dacă matricea unui sistem are deja această structură, metoda o exploatează direct şi necesită mai puţine operaţii şi mai puţină memorie decât o factorizare generală.

Principiu de alegere a metodei

Nu transformăm artificial orice sistem într-unul potrivit pentru Cholesky. În acest capitol alegem metoda după structura deja disponibilă a matricei: LU cu pivotare pentru cazul general, Cholesky pentru cazul SPD şi algoritmi specializaţi pentru matrici de bandă. Motivele numerice mai fine pentru această alegere vor putea fi formulate după introducerea normelor şi a analizei erorii.

Factorizarea Cholesky va fi folosită şi ulterior, în probleme care conduc la matrici SPD.

Dicţionar între convenţiile Cholesky

În corpul capitolului am folosit convenţia $A = H^T H$, cu H superior triunghiular. În multe implementări, inclusiv în exerciţiile de laborator, este mai comodă convenţia $A = LL^T$, cu $L = H^T$ inferior triunghiular. Formulele sunt echivalente prin $\ell_{ij} = h_{ji}$ pentru $i \geq j$:

$$\ell_{jj} = \sqrt{a_{jj} - \sum_{k<j} \ell_{jk}^2}, \qquad \ell_{ij} = \frac{1}{\ell_{jj}} \left(a_{ij} - \sum_{k<j} \ell_{ik} \ell_{jk} \right), \quad i > j.$$

În laborator vom folosi consecvent forma inferioară $A = LL^T$.

Completare didactică: de ce Cholesky este metoda naturală pentru SPD

Dacă A este simetrică pozitiv definită, informaţia structurală trebuie folosită. Factorizarea Cholesky scrie $A = LL^T$, cu L triunghiulară inferior şi diagonală strict pozitivă. Faţă de o factorizare LU generală, se stochează practic un singur factor şi costul aritmetic se reduce aproximativ la jumătate: termenul dominant este de ordin $\frac{1}{3}n^3$.

Pozitiva definire garantează că expresiile aflate sub radical în calculul elementelor diagonale sunt pozitive. Astfel, proprietatea matematică a matricei este exact ceea ce permite existenţa algoritmului fără pivotare.

Exemplu construit din factor

Considerăm

$$L = \begin{pmatrix} 2 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

Atunci

$$A = LL^T = \begin{pmatrix} 4 & 2 & 2 \\ 2 & 5 & 1 \\ 2 & 1 & 2 \end{pmatrix}$$

este automat simetrică pozitiv definită. Dacă $x_\star = (1, 2, 3)^T$, atunci $b = Ax_\star = (14, 15, 10)^T$. Pentru rezolvare efectuăm $Ly = b$ şi apoi $L^T x = y$. Acest exemplu este util şi pentru testarea unui cod: fiindcă A a fost construită dintr-un factor cunoscut, avem o verificare independentă.

Implementarea factorizării Cholesky

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi formulele element-cu-element, fără a apela `chol`. Testaţi mai întâi pe matrice de forma $A = RR^T + \alpha I$, cu $\alpha > 0$, unde R este aleatoare. Comparaţi `norm(A-L*L')` cu precizia maşinii. Apoi modificaţi A astfel încât să devină indefinită şi observaţi locul în care algoritmul încearcă să extragă rădăcina pătrată dintr-o valoare nepozitivă.

Aplicaţie: ecuaţii normale

Dacă o matrice are forma $A^T A$ şi A are rang coloană complet, atunci $A^T A$ este simetrică pozitiv definită şi poate fi factorizată Cholesky. Aplicaţiile acestei observaţii la probleme dreptunghiulare vor fi discutate mai târziu, după introducerea instrumentelor necesare pentru analiza erorii.

Exerciţii şi verificarea înţelegerii

1. Calculaţi factorizarea Cholesky a matricei $\begin{pmatrix} 4 & 2 \\ 2 & 5 \end{pmatrix}$.

2. Demonstraţi direct că dacă $A = LL^T$ şi L este inversabilă, atunci A este pozitiv definită.
3. Arătaţi că o matrice SPD este inversabilă şi că elementele sale diagonale sunt pozitive.
4. Comparaţi numărul de elemente care trebuie stocate pentru o matrice simetrică densă cu cel necesar pentru o matrice generală.
5. Explicaţi de ce pivotarea parţială standard ar distruge, în general, simetria dacă ar fi aplicată naiv într-un algoritm Cholesky.

Mini-proiect: matrice Gram

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Generaţi n vectori $v_1, \dots, v_n \in \mathbb{R}^m$ liniar independenţi. Matricea $G = (v_i^T v_j)_{ij} = V^T V$, ale cărei elemente sunt produsele scalare ale vectorilor, se numeşte matrice Gram. Formaţi această matrice. Verificaţi pozitiva definire şi calculaţi Cholesky. Apoi apropiaţi două coloane una de alta şi urmăriţi determinantul lui G şi elementele diagonale ale factorului Cholesky.

Rezolvare orientativă

Cholesky 2×2

Pentru $A = \begin{pmatrix} 4 & 2 \\ 2 & 5 \end{pmatrix}$, scriem $L = \begin{pmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{pmatrix}$. Din $LL^T = A$ rezultă $\ell_{11}^2 = 4$, deci alegem diagonala pozitivă $\ell_{11} = 2$. Apoi $\ell_{21}\ell_{11} = 2$, deci $\ell_{21} = 1$. Ultima relaţie este $\ell_{21}^2 + \ell_{22}^2 = 5$, de unde $\ell_{22} = 2$. Astfel $L = \begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix}$.
Dacă $b = (6, 7)^T$, rezolvarea $Ly = b$ dă $y = (3, 2)^T$, iar $L^T x = y$ dă $x = (1, 1)^T$. Nu apare niciun calcul al inversei.

Situaţie practică: matrice Gram aproape singulară

Dacă două coloane ale unei matrici V sunt aproape egale, matricea Gram $G = V^T V$ rămâne teoretic SPD cât timp coloanele sunt independente, dar cea mai mică valoare proprie devine foarte mică. Cholesky poate calcula în continuare factorii, însă problema este sensibilă. Acesta este un exemplu bun de separare între existenţa factorizării şi calitatea numerică a soluţiei.

Exemplu complet: Cholesky şi rezolvarea unui sistem SPD

Fie

$$A = \begin{pmatrix} 4 & 2 \\ 2 & 3 \end{pmatrix}, \qquad b = \begin{pmatrix} 6 \\ 5 \end{pmatrix}.$$

Matricea este simetrică, iar minorii principali sunt $4 > 0$ şi $\det A = 8 > 0$, deci este pozitiv definită. Căutăm $A = LL^T$ cu $L = \begin{pmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{pmatrix}$. Din identificarea elementelor rezultă $\ell_{11} = 2$, $\ell_{21} = 1$ şi $\ell_{22} = \sqrt{3-1} = \sqrt{2}$. Prin urmare

$$L = \begin{pmatrix} 2 & 0 \\ 1 & \sqrt{2} \end{pmatrix}.$$

Rezolvăm $Ly = b$: $y_1 = 3$, iar $y_2 = (5-3)/\sqrt{2} = \sqrt{2}$. Apoi $L^T x = y$: din a doua ecuaţie $\sqrt{2}x_2 = \sqrt{2}$, deci $x_2 = 1$, iar din prima $2x_1 + x_2 = 3$, deci $x_1 = 1$. Soluţia este $x = (1, 1)^T$.

De ce este Cholesky aproximativ de două ori mai ieftină decât LU?

Pentru o matrice simetrică nu avem nevoie să calculăm şi să stocăm separat informaţia de deasupra şi de sub diagonală: una este transpusul celeilalte. Numărul dominant de operaţii pentru Cholesky este aproximativ $n^3/3$, faţă de aproximativ $2n^3/3$ pentru eliminarea gaussiană densă. Avantajul este simultan algebric, numeric şi de memorie.

Sarcină practică: detectarea pierderii pozitivităţii

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi Cholesky folosind formula diagonală $\ell_{jj} = \sqrt{a_{jj} - \sum_{k<j} \ell_{jk}^2}$. Dacă expresia de sub radical devine nepozitivă, opriţi algoritmul şi raportaţi indicele. Testaţi pe matrici $A = B^T B + \alpha I$ pentru $\alpha > 0$, apoi micşoraţi α. Comparaţi cu o matrice simetrică indefinită. Explicaţi de ce simpla simetrie nu este suficientă.

Exerciţii suplimentare

1. Calculaţi factorul Cholesky pentru $A = \begin{pmatrix} 9 & 3 \\ 3 & 5 \end{pmatrix}$ şi verificaţi produsul.
2. Demonstraţi că dacă $A = LL^T$ cu L nesingulară, atunci $x^T A x = (L^T x)^T (L^T x) > 0$ pentru orice $x \neq 0$.
3. Arătaţi că matricea Gram $A^T A$ este pozitiv semidefinită şi precizaţi când este pozitiv definită.
4. Pentru A cu rang coloană complet, demonstraţi direct că $A^T A$ este SPD şi explicaţi de ce Cholesky este aplicabilă acestei matrice.
5. Formulaţi o verificare element cu element a identităţii $A = LL^T$ pentru o factorizare calculată.

3.7 Matrice cu structură de bandă

Matrice cu structură de bandă

Definiţie 3.7.1. *Spunem că o matrice $A \in \mathbb{R}^{m \times n}$ are bandă inferioară p dacă $a_{ij} = 0$ când $i > j + p$ şi banda superioară q dacă $a_{ij} = 0$ când $j > i + q$.*

Matricele diagonale sunt matrici cu benzi pentru care $p = q = 0$, în timp ce matricele bidiagonale au $p = 1, q = 0$ sau $p = 0, q = 1$ iar dacă $p = 1$ şi $q = 1$ atunci spunem că avem o matrice tridiagonală.

Rezultatul principal pentru matrici cu benzi este următorul.

Teoremă 3.7.2. *Fie $A \in \mathbb{R}^{n\times n}$. Să presupunem că există o factorizare LU a lui A. Dacă A are lăţimea de bandă superioară q şi lăţimea de bandă inferioară p, atunci L are lăţimea de bandă inferioară p şi U are lăţimea de bandă superioară q.*

Demonstraţie. Folosim formulele lui Doolittle şi inducţia după pasul k. Presupunem că primele $k-1$ rânduri din U şi primele $k-1$ coloane din L respectă benzile. Dacă $j > k+q$, atunci $a_{kj} = 0$, iar

$$u_{kj} = a_{kj} - \sum_{r=1}^{k-1} l_{kr}u_{rj} = 0,$$

pentru că $j > k+q > r+q$ implică $u_{rj} = 0$ pentru orice $r < k$. Analog, dacă $i > k+p$, atunci $a_{ik} = 0$ şi

$$l_{ik} = \frac{1}{u_{kk}}\left(a_{ik} - \sum_{r=1}^{k-1} l_{ir}u_{rk}\right) = 0,$$

deoarece $i > k+p > r+p$ implică $l_{ir} = 0$. Inducţia porneşte la $k = 1$ şi arată că eliminarea nu creează elemente nenule în afara benzilor prescrise. □

În special, observaţi că aceeaşi zonă de memorie utilizată pentru A este suficientă pentru a stoca şi factorizarea sa LU.

Să considerăm, într-adevăr, că o matrice A având lăţimea de bandă superioară q şi lăţimea de bandă inferioară p este de obicei stocată într-o matrice $(p+q+1)\times n$, pe care o vom nota cu B, presupunând că

$$b_{i-j+q+1,j} = a_{ij} \tag{3.30}$$

pentru toţi indicii i, j care se încadrează în banda matricei, în rest fiind zero.

De exemplu, în cazul matricei tridiagonale

$$A = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix} \tag{3.31}$$

stocarea compactă se citeşte

$$B = \begin{pmatrix} 0 & -1 & -1 & -1 & -1 \\ 2 & 2 & 2 & 2 & 2 \\ -1 & -1 & -1 & -1 & 0 \end{pmatrix} \tag{3.32}$$

Acelaşi format poate fi utilizat pentru stocarea factorizării LU a lui A.

Este clar că acest format de stocare poate fi destul de incomod în cazul în care doar câteva benzi ale matricei sunt mari.

La limită, dacă doar o coloană şi un rând ar fi pline, am avea $p = q = n$ şi astfel B ar fi o matrice plină cu multe intrări zero.

În cele din urmă, observăm că inversa unei matrice cu benzi este în general plină (aşa cum se întâmplă pentru matricea A considerată mai sus).

Matrice tridiagonale

Considerăm cazul particular al unui sistem liniar cu matrice tridiagonală nesingulară A dată de

$$\begin{pmatrix} a_1 & c_1 & & 0 \\ b_2 & a_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ 0 & & b_n & a_n \end{pmatrix}. \tag{3.33}$$

În acest caz, matricile L şi U din factorizarea LU a lui A sunt matrici bidiagonale de forma

$$\begin{pmatrix} 1 & 0 & & 0 \\ \beta_2 & 1 & \ddots & \\ & \ddots & \ddots & 0 \\ 0 & & \beta_n & 1 \end{pmatrix}, \qquad \begin{pmatrix} \alpha_1 & c_1 & & \\ 0 & \alpha_2 & \ddots & 0 \\ & \ddots & \ddots & c_{n-1} \\ 0 & & 0 & \alpha_n \end{pmatrix} \tag{3.34}$$

Algoritmul Thomas

Coeficienţii α_i, β_i pot fi calculaţi cu uşurinţă prin următoarele relaţii

$$\alpha_1 = a_1, \quad \beta_i = \frac{b_i}{\alpha_{i-1}}, \quad \alpha_i = a_i - \beta_i c_{i-1}, \qquad i = 2, \cdots, n. \tag{3.35}$$

Algoritmul Thomas poate fi extins şi pentru a rezolva întregul sistem tridiagonal $Ax = f$. Acest lucru înseamnă rezolvarea a două sisteme bidiagonale $Ly = f$ şi $Ux = y$, pentru care se aplică următoarele formule:

$$(Ly = f): \quad y_1 = f_1, y_i = f_i - \beta_i y_{i-1}, \qquad i = 2, \cdots, n, \tag{3.36}$$

$$(Ux = y): \quad x_n = \frac{y_n}{\alpha_n}, \quad x_i = \frac{y_i - c_i x_{i+1}}{\alpha_i}, \qquad i = n-1, \cdots, 1. \tag{3.37}$$

Sisteme pe blocuri

În această secţiune ne ocupăm de factorizarea LU a matricelor partiţionate în blocuri, în care fiecare bloc poate avea o dimensiune diferită.

Obiectivul nostru este dublu: optimizarea ocupării spaţiului de stocare prin exploatarea adecvată a structurii matricei şi reducerea costului de calcul al soluţiei sistemului.

Factorizarea *LU*

Fie $A = \mathbb{R}^{n\times n}$ următoarea matrice partiţionată în blocuri

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \tag{3.38}$$

unde $A_{11} \in \mathbb{R}^{r\times r}$ este o matrice nesingulară a cărei factorizare $L_{11}D_1R_{11}$ este cunoscută, în timp ce $A_{22} \in \mathbb{R}^{(n-r)\times(n-r)}$.

În acest caz este posibilă factorizarea A folosind doar factorizarea LU a blocului A_{11}. Într-adevăr, este adevărat că

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & I_{n-r} \end{pmatrix} \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & I_{n-r} \end{pmatrix}, \tag{3.39}$$

unde

$$\begin{aligned} L_{21} &= A_{21} R_{11}^{-1} D_1^{-1}, \\ R_{12} &= D_1^{-1} L_{11}^{-1} A_{12}, \\ D_2 &= A_{22} - L_{21} D_1 R_{12}. \end{aligned} \tag{3.40}$$

Dacă este necesar, procedura de reducere poate fi repetată pe matricea D_2, obţinându-se astfel o versiune în bloc a factorizării LU.

Dacă A_{11} ar fi un scalar, abordarea de mai sus ar reduce cu unu dimensiunea factorizării unei matrice date.

Prin aplicarea iterativă a acestei metode se obţine un mod alternativ de efectuare a eliminării Gauss.

Completare didactică: exploatarea structurii de bandă

O matrice de bandă are majoritatea elementelor exact nule, iar aceste zerouri nu trebuie nici stocate, nici procesate. Dacă semi-lăţimile benzii sunt fixe când n creşte, costul unei factorizări scade de la ordin cubic la ordin liniar în n. Acest exemplu ilustrează principiul general: *algoritmul trebuie ales în funcţie de structură, nu doar în funcţie de dimensiune.*

Matricele tridiagonale apar la discretizarea derivatelor de ordinul al doilea. Pentru problema $-u''(x) = f(x)$, formula diferenţelor finite centrate produce, după înmulţirea cu h^2, o matrice cu 2 pe diagonală şi -1 pe diagonalele imediat vecine.

Exemplu: sistem tridiagonal

Pentru

$$T = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

se verifică imediat că $x = (1, 1, 1, 1)^T$ este soluţie. Algoritmul Thomas este eliminarea gaussiană specializată pentru această structură; el modifică doar cele trei diagonale şi are cost $O(n)$.

Dacă notăm cu a_i subdiagonala, d_i diagonala şi c_i supradiagonala, la pasul i putem elimina a_i folosind multiplicatorul $m_i = a_i / \widetilde{d}_{i-1}$, apoi actualizăm numai $\widetilde{d}_i = d_i - m_i c_{i-1}$ şi termenul liber $\widetilde{b}_i = b_i - m_i \widetilde{b}_{i-1}$. Substituţia înapoi foloseşte aceeaşi structură locală.

Laborator: Thomas versus matrice densă

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al

cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi un algoritm Thomas care primeşte doar cei trei vectori ai diagonalelor şi vectorul b. Nu construiţi matricea $n \times n$. Comparaţi timpul şi memoria cu o versiune densă pentru $n = 10^3, 10^4, 10^5$ atunci când este posibil. Comparaţi apoi cu reprezentarea rară MATLAB, de tip `sparse`.

Exerciţii şi verificarea înţelegerii

1. Aplicaţi manual algoritmul Thomas sistemului de ordin 4 de mai sus.
2. Determinaţi numărul de valori scalare necesare pentru stocarea unei matrice tridiagonale de ordin n.
3. Arătaţi că matricea tridiagonală cu 2 pe diagonală şi -1 pe diagonalele vecine este SPD.
4. Ce condiţie trebuie verificată pentru ca pasul $m_i = a_i/\widetilde{d}_{i-1}$ să fie definit? Discutaţi ce garantează pozitiva definire în exemplul Poisson.
5. Generalizaţi ideea de stocare la o matrice cu p subdiagonale şi q supradiagonale nenule.

Studiu practic: temperatură într-o bară

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Consideraţi o bară unidimensională cu temperaturi prescrise la capete şi cu o sursă termică $f(x)$. Discretizaţi problema staţionară pentru mai multe valori ale lui n, rezolvaţi sistemul prin Thomas şi comparaţi cu o soluţie analitică pentru un f simplu. Studiaţi separat eroarea de discretizare şi eroarea algoritmului liniar.

Rezolvare orientativă

Thomas pentru sistemul de ordin 4

Pentru diagonalele $a_i = -1$, $d_i = 2$, $c_i = -1$ şi $b = (1, 0, 0, 1)^T$, eliminăm succesiv. Primul multiplicator este $m_2 = -1/2$, deci noul pivot este $\widetilde{d}_2 = 2 - (-1/2)(-1) = 3/2$, iar $\widetilde{b}_2 = 1/2$. Apoi $m_3 = -2/3$, $\widetilde{d}_3 = 4/3$, $\widetilde{b}_3 = 1/3$. În final $m_4 = -3/4$, $\widetilde{d}_4 = 5/4$, $\widetilde{b}_4 = 5/4$. Substituţia înapoi dă $x_4 = 1$, apoi $x_3 = 1$, $x_2 = 1$, $x_1 = 1$.
Am folosit numai diagonalele, nu matricea completă. Pentru n mare, aceasta este esenţa economiei.

Situaţie practică: discretizare versus rezolvare

Dacă soluţia numerică a unei EDP are eroare de ordin h^2, nu are sens să rezolvăm sistemul liniar până la o toleranţă de ordin 10^{-14} pentru un h relativ mare. Toleranţa algoritmului liniar trebuie pusă în contextul erorii de modelare/discretizare. Într-un calcul ştiinţific complet, “precizia” nu este determinată de un singur pas al lanţului de calcul.

Exemplu rezolvat: un sistem tridiagonal provenit din diferenţe finite

Considerăm matricea

$$T = \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Vectorul $x = (1, 1, 1, 1)^T$ este soluţie, deoarece pe liniile interioare apare combinaţia $-1+2-1 = 0$, iar pe liniile de margine obţinem 1. O astfel de matrice apare la discretizarea operatorului $-u''$ cu condiţii la frontieră.

Dacă ignorăm structura şi folosim o factorizare densă, alocăm $O(n^2)$ memorie şi efectuăm $O(n^3)$ operaţii. Algoritmul Thomas pentru o matrice tridiagonală păstrează numai cele trei diagonale şi efectuează un număr de operaţii proporţional cu n. Pentru probleme obţinute din discretizări cu milioane de necunoscute, diferenţa dintre cele două perspective este decisivă.

Sarcină de implementare: algoritmul Thomas

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Reprezentaţi matricea numai prin trei vectori: subdiagonala a, diagonala d şi supradiagonala c. Implementaţi eliminarea înainte direct pe aceşti vectori şi apoi substituţia înapoi. Testaţi familia tridiagonală cu diagonala 2 şi diagonalele vecine -1 pentru $n = 10^2, 10^3, 10^4, 10^5$. Comparaţi timpul şi memoria cu o reprezentare densă numai pentru dimensiunile la care aceasta este încă fezabilă.

Exerciţii suplimentare

1. Executaţi manual algoritmul Thomas pentru sistemul 4×4 de mai sus.
2. Arătaţi că matricea tridiagonală cu 2 pe diagonală şi -1 pe diagonalele vecine este SPD.
3. Dacă lăţimea totală de bandă este fixă şi independentă de n, estimaţi costul factorizării în funcţie de n.
4. Explicaţi de ce stocarea zerourilor din afara benzii este o risipă şi propuneţi o structură de date.
5. Investigaţi numeric ce se întâmplă când unul dintre pivoţii algoritmului Thomas devine foarte mic. Când ar fi necesară pivotarea?

Fişă de seminar şi laborator: metode directe în ansamblu

Problema rezolvată 1: alegerea algoritmului după structură

Presupunem că trebuie rezolvate patru sisteme de aceeaşi dimensiune: (a) o matrice densă generală; (b) o matrice densă SPD; (c) o matrice tridiagonală SPD; (d) aceeaşi matrice densă generală pentru 500 de termeni liberi. În cazul (a), alegerea standard este LU cu pivotare.

În (b), Cholesky exploatează simetria şi pozitiva definire şi reduce costul şi stocarea. În (c), folosirea unei factorizări dense ar ignora aproape toate zerourile; un algoritm tridiagonal are cost liniar. În (d), cheia este refolosirea factorilor: factorizăm o dată şi efectuăm câte două substituţii pentru fiecare termen liber.

Această problemă nu cere un calcul complicat; urmăreşte recunoaşterea structurii înainte de aplicarea unui algoritm generic.

Problema rezolvată 2: determinantul fără dezvoltare Laplace

Dacă $PA = LU$, atunci $\det(P)\det(A) = \det(L)\det(U) = \prod_i u_{ii}$, deoarece L este unitară. Cum $\det(P) = \pm 1$, determinantul lui A se obţine aproape gratuit după factorizare. Numărul de schimburi de linii determină semnul lui $\det(P)$. Astfel, o cantitate care prin dezvoltare Laplace ar fi prohibitivă poate fi calculată cu aceeaşi factorizare folosită pentru rezolvarea sistemului.

Laborator integrator 3: o rutină de rezolvare adaptivă

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o funcţie de nivel înalt care inspectează matricea şi alege o strategie: dacă este triunghiulară, substituţie; dacă este simetrică şi Cholesky reuşeşte, Cholesky; dacă este tridiagonală, Thomas; altfel, LU cu pivotare. Raportaţi metoda aleasă. Nu urmăriţi să reproduceţi toate optimizările unei biblioteci profesionale, ci să demonstraţi logica deciziei şi să testaţi fiecare ramură pe exemple controlate.

Set de probleme

1. Pentru o matrice densă $n \times n$, comparaţi ordinele de cost ale factorizării şi substituţiei. Pentru câţi termeni liberi începe să domine costul substituţiilor?
2. Arătaţi că dacă A este SPD, pivoţii Cholesky sunt reali şi pozitivi.
3. Construiţi o matrice pentru care LU fără pivotare eşuează la primul pas, dar $PA = LU$ există.
4. Daţi un exemplu singular cu factorizare LU unică şi explicaţi de ce ultimul pivot poate fi zero.
5. Demonstraţi formula $S_t^{-1} = I - t e_{k+1} e_k^T$ folosită în demonstraţia unicităţii.
6. Pentru o matrice pentadiagonală, estimaţi stocarea necesară prin vectori de diagonale.
7. Comparaţi determinantul obţinut din LU cu `det(A)` pentru matrici apropiate de singularitate şi observaţi cât de repede poate deveni determinantul foarte mic. Nu folosiţi această observaţie ca măsură cantitativă a sensibilităţii; aceasta va fi definită în capitolul următor.
8. Într-o aplicaţie cu matrice fixă şi termeni liberi care sosesc în timp real, proiectaţi o structură care păstrează factorii şi rezolvă noile sisteme.

Proiect de capitol: bibliotecă minimală de metode directe

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Realizaţi un pachet cu funcţii pentru substituţie înainte/înapoi, LU cu pivotare, Cholesky şi sistem tridiagonal. Fiecare funcţie trebuie să aibă teste unitare pe soluţii fabricate şi mesaje clare în caz de eşec. Comparaţi componentele soluţiei calculate cu soluţia fabricată şi cu rezultatul unei rutine MATLAB standard. Adăugaţi un benchmark pentru dimensiuni crescătoare şi discutaţi timpul, memoria şi ipotezele fiecărei metode.

Sarcini de aprofundare şi sinteză a metodelor directe

Probleme de aprofundare: factorizare, pivotare şi structură

1. Pentru $A(c) = \begin{pmatrix} 1 & c & 0 \\ 2 & 4 & 1 \\ 3 & 5 & 1 \end{pmatrix}$ determinaţi valorile lui c pentru care eliminarea gaussiană fără pivotare produce un pivot nul la pasul al doilea sau al treilea. Decideţi separat dacă matricea este singulară sau numai eliminarea fără schimbări de linii eşuează.
2. Demonstraţi legătura dintre existenţa factorizării $A = LU$ cu L unitară şi nenulitatea minorilor principali conducători care apar în procesul de eliminare.
3. Construiţi o matrice inversabilă 3×3 pentru care eliminarea fără pivotare eşuează la primul pas, dar o singură permutare de linii permite continuarea. Scrieţi explicit matricea de permutare.
4. Pentru $A = LU$, cu L unitară, demonstraţi $\det A = \prod_i u_{ii}$. Cum se modifică formula pentru $PA = LU$?
5. Arătaţi că dacă A este simetrică pozitiv definită, atunci factorizarea Cholesky poate fi construită pas cu pas şi elementele diagonale ale factorului sunt strict pozitive.
6. Pentru o matrice tridiagonală deduceţi recurenţele algoritmului Thomas din eliminarea gaussiană şi număraţi operaţiile necesare la fiecare pas. Explicaţi de ce numărul total creşte liniar cu n.
7. Presupuneţi că aceeaşi matrice A apare în sistemele $Ax = b_j$, $j = 1, \dots, m$. Comparaţi numărul de operaţii necesare dacă factorizaţi A la fiecare rezolvare cu situaţia în care factorii sunt calculaţi o singură dată.
8. Pentru o matrice de bandă cu semilăţime p, număraţi aproximativ elementele care trebuie stocate şi comparaţi cu cele n^2 elemente ale unei matrice dense.
9. Explicaţi diferenţa algoritmică dintre un pivot exact nul şi un pivot nenul, dar foarte mic. Construiţi un exemplu 2×2 pentru fiecare situaţie şi urmăriţi eliminarea numerică.
10. Comparaţi LU şi Cholesky pentru o matrice simetrică pozitiv definită: ce informaţie suplimentară foloseşte Cholesky şi ce operaţii sunt evitate?

Laborator integrator: un solver care exploatează structura

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Implementaţi metodele din capitol şi comparaţi-le fără a folosi încă instrumentele de analiză introduse în capitolul următor.

1. Scrieţi funcţii separate pentru substituţie înainte, substituţie înapoi, LU cu pivotare parţială, Cholesky şi Thomas. Nu folosiţi `inv`.
2. Testaţi-le pe: o matrice densă generală, una simetrică pozitiv definită şi una tridiagonală.
3. Pentru fiecare familie alegeţi $x_\star = \mathbf{1}$, formaţi $b = Ax_\star$ şi comparaţi componentele soluţiei calculate cu cele ale lui $x_\star$.
4. Pentru dimensiuni crescătoare măsuraţi timpul de factorizare şi timpul necesar rezolvării după ce factorii sunt deja disponibili.
5. Pentru problema cu mai mulţi termeni liberi comparaţi refactorizarea repetată cu reutilizarea factorilor.
6. Pentru matricea tridiagonală comparaţi memoria necesară stocării dense cu memoria necesară celor trei diagonale.
7. Comparaţi rezultatele cu `A\b` numai ca verificare externă şi raportaţi cea mai mare diferenţă absolută dintre componentele celor două soluţii.

Proiect de capitol: discretizarea unei probleme la limită 1D

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Consideraţi problema $-u''(x) = f(x)$ pe $(0,1)$, cu valorile soluţiei prescrise la capete, $u(0) = u(1) = 0$. Acest tip de condiţie la frontieră se numeşte *condiţie Dirichlet*. Formula diferenţelor finite centrale şi construcţia sistemului tridiagonal au fost prezentate în exemplul de discretizare din acest capitol.

1. **Modelare.** Folosiţi diferenţe finite centrale şi deduceţi sistemul tridiagonal pentru necunoscutele interioare.
2. **Calcul manual.** Pentru trei necunoscute interioare scrieţi complet sistemul şi efectuaţi eliminarea de tip Thomas.
3. **Structură.** Arătaţi că matricea este simetrică şi pozitiv definită.
4. **Algoritm.** Implementaţi rezolvarea prin Cholesky şi Thomas.
5. **Experiment.** Folosiţi o soluţie exactă simplă pentru a genera termenul drept şi comparaţi valorile calculate în nodurile grilei cu valorile exacte.
6. **Complexitate.** Măriţi numărul de puncte şi comparaţi timpul celor două implementări.

7. **Interpretare.** Explicaţi de ce structura tridiagonală trebuie folosită în alegerea algoritmului şi nu tratată doar ca un detaliu de programare.

Test de capitol

Test – Metode directe pentru sisteme liniare

1. Calcul explicit şi verificare critică. Pentru matricea

$$A = \begin{pmatrix} 4 & 2 & 0 \\ 2 & 3 & 1 \\ 0 & 1 & 2 \end{pmatrix},$$

un instrument AI afirmă că o factorizare Cholesky este

$$L = \begin{pmatrix} 2 & 0 & 0 \\ 1 & \sqrt{2} & 0 \\ 0 & 1/\sqrt{2} & 1 \end{pmatrix}.$$

Este afirmaţia corectă? Verificaţi prin calculul lui LL^T. Dacă este necesar, construiţi factorizarea corectă pas cu pas. Înainte de calcul, explicaţi de ce factorizarea Cholesky este sau nu este permisă pentru această matrice.

2. Cod MATLAB. Se doreşte rezolvarea unui sistem $Ax = b$ după o factorizare $PA = LU$ deja calculată:

```
y = L\(P*b);
x = U\y;
```

Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi:

A) prima instrucţiune reprezintă o substituţie înainte;
B) a doua instrucţiune reprezintă o substituţie înapoi;
C) vectorul b nu trebuie permutat niciodată;
D) aceeaşi factorizare poate fi reutilizată pentru mai mulţi termeni liberi.

Capitolul 4

Norme, condiţionare şi analiza erorii

Aritmetica în virgulă mobilă

Calculatorul lucrează cu un număr finit de valori reprezentabile. Din acest motiv, rezultatul unei operaţii aritmetice trebuie, în general, rotunjit. Aceste rotunjiri explică de ce două formule echivalente algebric pot conduce la rezultate numerice diferite.

Definiţie 4.0.1 (Modelul elementar de rotunjire)**.** *Notăm cu* $\mathrm{fl}(x)$ *valoarea lui* x *după rotunjire. Pentru o operaţie elementară* $\circ \in \{+, -, \times, /\}$*, dacă nu apare overflow sau underflow şi rezultatul este normalizat, se foloseşte modelul*

$$\mathrm{fl}(a \circ b) = (a \circ b)(1 + \delta), \qquad |\delta| \leq u,$$

unde u *este* unit roundoff.

În aritmetica IEEE 754, pentru precizia simplă binară avem $u = 2^{-24} \simeq 5.96 \cdot 10^{-8}$, iar pentru precizia dublă $u = 2^{-53} \simeq 1.11 \cdot 10^{-16}$. Aceste valori descriu mărimea erorii relative produse de o singură rotunjire; ele nu reprezintă o estimare directă a erorii unui calcul cu multe operaţii.

De ce 0.1 nu este reprezentabil exact în binar

Numărul zecimal 0.1 are o dezvoltare binară infinită, aşa cum 1/3 are o dezvoltare zecimală infinită. În calculator se păstrează numai un număr finit de cifre binare. De aceea, pentru rezultate obţinute numeric, testul `a==b` trebuie folosit cu atenţie. De regulă se verifică dacă $|a - b|$ este mai mic decât o toleranţă aleasă în funcţie de mărimea datelor.

Anularea la scădere

La scăderea a două numere foarte apropiate se pot pierde multe cifre semnificative. De exemplu,

$$\sqrt{x+1} - \sqrt{x}, \qquad x \gg 1,$$

se calculează mai bine folosind forma echivalentă

$$\sqrt{x+1} - \sqrt{x} = \frac{1}{\sqrt{x+1} + \sqrt{x}}.$$

A doua formulă evită scăderea a două numere apropiate. Aceeaşi problemă apare la construirea reflectorilor Householder şi la calculul rădăcinilor unei ecuaţii de gradul al doilea.

De ce apare 2^{-27} într-un exemplu ulterior

Dacă o coloană conţine o perturbaţie de ordin 2^{-27}, atunci în A^TA apar termeni de ordin 2^{-54}. În precizie dublă, $2^{-54} < u = 2^{-53}$, iar aceşti termeni se pot pierde când sunt adunaţi la numere de ordinul unu. Exemplul din capitolul despre cele mai mici pătrate arată, printr-un caz concret, una dintre dificultăţile formării ecuaţiilor normale.

Eroare înainte şi eroare înapoi

Pentru sistemul $Ax = b$, *eroarea înainte* compară soluţia calculată $\widehat{x}$ cu soluţia exactă x. *Eroarea înapoi* măsoară cât trebuie modificate datele pentru ca $\widehat{x}$ să fie soluţia exactă a problemei modificate. Dacă

$$(A + \delta A)\widehat{x} = b + \delta b,$$

atunci, pentru perturbaţii mici, se obţine estimarea

$$\frac{\|x - \widehat{x}\|}{\|x\|} \lesssim \kappa(A) \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right).$$

Prin urmare, eroarea rezultatului depinde atât de algoritm, cât şi de condiţionarea problemei.

Eliminarea Gauss şi erorile de rotunjire

Pentru eliminarea Gauss cu pivotare parţială, analiza erorii conduce la estimări de forma

$$\|\delta A\|_\infty \leq c_n \, \rho_n \, u \, \|A\|_\infty,$$

unde ρ_n este factorul de creştere din procesul de eliminare, iar c_n depinde de dimensiune şi de modul în care sunt grupate operaţiile. Factorul ρ_n explică de ce, în unele exemple, erorile de rotunjire pot fi amplificate chiar dacă se foloseşte pivotare parţială.

Exerciţii

1. Pentru $x = 10^{10}$, calculaţi $\sqrt{x+1} - \sqrt{x}$ direct şi prin formula raţionalizată. Comparaţi rezultatele în precizie dublă.
2. Explicaţi de ce pragul de rang numeric $\tau \approx \max(m, n) u \sigma_1$ depinde atât de precizia maşinii, cât şi de mărimea valorilor singulare.
3. Daţi un exemplu în care $\|b - A\widehat{x}\|$ este mic, dar $\|x - \widehat{x}\|$ este mare. Ce rol are numărul de condiţionare?

Descompunerea spectrală

Unul dintre cele mai utile rezultate legate de valorile proprii este teorema descompunerii spectrale, care afirmă că orice matrice simetrică A are o bază ortonormală de vectori proprii.

Reamintire: teorema spectrală

Fie A o matrice simetrică în $\mathbb{R}^{n \times n}$. Atunci există o matrice ortogonală $U \in \mathbb{R}^{n \times n}$ ($U^TU = U\,U^T = I$) şi o matrice diagonală $D = \operatorname{diag}(d_1, d_2, ..., d_n)$ pentru care

$$U^T \, A \, U = D.$$

Coloanele matricei U din factorizare constituie o bază ortonormată formată din vectorii

proprii ai lui A, iar elementele diagonale ale lui D sunt valorile proprii corespunzătoare.

Din descompunerea spectrală $A = UDU^T$ rezultă imediat formulele pentru urmă şi determinant. Folosind invarianta urmei la transformări de similitudine şi multiplicativitatea determinantului,

$$\operatorname{tr}(A) = \operatorname{tr}(D) = \sum_{i=1}^{n} \lambda_i(A), \qquad \det A = \det D = \prod_{i=1}^{n} \lambda_i(A).$$

Aceste identităţi rămân adevărate pentru orice matrice pătratică, dacă valorile proprii sunt numărate cu multiplicitate algebrică; demonstraţia generală se obţine din polinomul caracteristic sau din forma Schur.

Norme matriceale

Ansamblul valorilor proprii ale lui A se numeşte spectrul lui A, notat prin $\sigma(A)$.

Se pot demonstra următoarele proprietăţi

$$\det(A) = \prod_{i=1}^{n} \lambda_i, \qquad \operatorname{tr}(A) = \sum_{i=1}^{n} \lambda_i \tag{4.1}$$

şi se concluzionează că $\sigma(A) = \sigma(A^T)$, şi $\sigma(A^H) = \sigma(\overline{A})$, unde $A^H = \overline{A}^T$.

Modulul maxim al valorilor proprii ale lui A se numeşte raza spectrală a lui A şi se notează cu

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|. \tag{4.2}$$

Norme în $\mathbb{R}^n$

Un exemplu de spaţiu normat este $\mathbb{R}^n$, echipat, de exemplu, cu norma p (sau norma Hölder); aceasta din urmă se defineşte pentru un vector x cu componente x_i ca fiind

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}, \qquad \text{pentru} \quad 1 \leq p < \infty. \tag{4.3}$$

Observaţi că limita pe măsură ce p merge la infinit a lui $\|x\|_p$ există, este finită şi este egală cu modulul maxim al componentelor lui x. O astfel de limită defineşte, la rândul său, o normă, numită norma infinit (sau norma maxim), dată de

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|. \tag{4.4}$$

Când $p = 2$, regăsim definiţia standard a normei euclidiene

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}} = (x^T x)^{\frac{1}{2}}. \tag{4.5}$$

Norme matriceale

Definiţie 4.0.2. *O normă matricială este o funcţie* $\|\cdot\| : \mathbb{R}^{m \times n} \to \mathbb{R}$ *astfel încât*

1. $\|A\| \geq 0 \; \forall A \in \mathbb{R}^{m \times n}$ *şi* $\|A\| = 0$ *dacă şi numai dacă* $A = 0$*;*
2. $\|\alpha A\| = |\alpha| \, \|A\| \; \forall \alpha \in \mathbb{R}, \forall A \in \mathbb{R}^{m \times n}$*;*

3. $\|A+B\| \leq \|A\| + \|B\| \ \forall\, A, B \in \mathbb{R}^{m\times n}$.

Definiţie 4.0.3. *Spunem că o normă matricială $\|\cdot\|_{\mathbb{R}^{m\times n}}$ este compatibilă sau consistentă cu normele vectorială $\|\cdot\|_{\mathbb{R}^m}$ şi $\|\cdot\|_{\mathbb{R}^n}$ dacă*

$$\|A\,x\|_{\mathbb{R}^m} \leq \|A\|_{\mathbb{R}^{m\times n}}\, \|x\|_{\mathbb{R}^n} \quad \forall\, x \in \mathbb{R}^n. \tag{4.6}$$

Definiţie 4.0.4. *Spunem că o normă matricială $\|\cdot\|$ este submultiplicativă dacă $\forall\, A \in \mathbb{R}^{n\times m}$, $\forall\, B \in \mathbb{R}^{m\times q}$.*

$$\|A\,B\| \leq \|A\|\,\|B\|. \tag{4.7}$$

În multe lucrări, definiţia unei norme matriciale include şi submultiplicitatea.

Această proprietate nu este satisfăcută de toate normele matriciale. De exemplu, norma $\|A\|_\Delta = \max_{i=1,\dots,n, j=1,\dots,m} |a_{ij}|$ nu îndeplineşte condiţia submultiplicativă, de exemplu, această condiţie nu este îndeplinită pentru pentru $A = B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$. Prin urmare, este o normă, dar nu este o normă submultiplicativă.

Observaţi că, dată fiind o anumită normă submultiplicativă $\|\cdot\|_\alpha$, există întotdeauna o normă vectorială compatibilă. De exemplu, dat fiind orice vector fix $y \neq 0$ în $\mathbb{R}^n$, este suficient să se definească norma vectorială consistentă sub forma

$$\|x\| = \|x\,y^T\|_\alpha \quad \forall\, x \in \mathbb{R}^n. \tag{4.8}$$

Un exemplu de normă matricială este norma Frobenius (sau norma euclidiană în $\mathbb{R}^{n^2}$)

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{n} |a_{ij}|^2} = \sqrt{\mathrm{tr}(A\,A^T)} \tag{4.9}$$

şi este compatibilă cu norma vectorială euclidiană $\|\cdot\|_2$. Într-adevăr,

$$\|A\,x\|_2^2 = \sum_{i=1}^{n} |\sum_{j=1}^{n} a_{ij}x_j|^2 \leq \sum_{i=1}^{n} \left(\sum_{j=1}^{n} |a_{ij}|^2 \sum_{j=1}^{n} |x_j|^2 \right) = \|A\|_F^2 \|x\|_2^2. \tag{4.10}$$

Observaţi că pentru o astfel de normă $\|I_n\|_F = \sqrt{n}$. Pentru o normă oarecare, care ar putea fi o aşteptare rezonabilă pentru $\|I_n\|$?

Norme matriceale induse

Teoremă 4.0.5. *Fie $\|\cdot\|_{\mathbb{R}^m}$ şi $\|\cdot\|_{\mathbb{R}^n}$ norme vectoriale. Funcţia*

$$\|A\| = \sup_{x\neq 0} \frac{\|Ax\|_{\mathbb{R}^m}}{\|x\|_{\mathbb{R}^n}}$$

este o normă matricială, numită normă indusă sau normă matricială naturală.

Demonstraţie. Mai întâi, supremul este finit. Într-adevăr, raportul este omogen de grad zero în x, deci este suficient să luăm supremul pe sfera unitate $\{x : \|x\|_{\mathbb{R}^n} = 1\}$. În dimensiune finită această sferă este compactă, iar aplicaţia $x \mapsto \|Ax\|_{\mathbb{R}^m}$ este continuă.

Avem evident $\|A\| \geq 0$. Dacă $A = 0$, atunci $\|A\| = 0$. Reciproc, dacă $\|A\| = 0$, atunci $\|Ax\| = 0$ pentru orice x, deci $Ax = 0$ pentru orice x şi prin urmare $A = 0$.

Pentru orice scalar $\alpha \in \mathbb{R}$, $\|\alpha A\| = \sup_{x\neq 0} |\alpha|\,\|Ax\|/\|x\| = |\alpha|\,\|A\|$. În sfârşit, din inegalitatea triunghiului pentru norma vectorială,

$$\frac{\|(A+B)x\|}{\|x\|} \leq \frac{\|Ax\|}{\|x\|} + \frac{\|Bx\|}{\|x\|} \leq \|A\| + \|B\|.$$

Luând supremul după $x \neq 0$, rezultă $\|A+B\| \leq \|A\| + \|B\|$. □

Cazuri relevante de norme matriceale induse sunt aşa-numitele norme p definite astfel

$$\|A\|_p = \sup_{x\neq 0} \frac{\|A\,x\|_p}{\|x\|_p}. \tag{4.11}$$

Norma 1 şi norma infinit sunt uşor de calculat, deoarece

$$\|A\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}|, \qquad \|A\|_\infty = \max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}| \tag{4.12}$$

şi se numesc norma sumei coloanelor şi, respectiv, norma sumei rândurilor.

Mai mult, avem $\|A\|_1 = \|A^T\|_\infty$ şi dacă A este simetrică $\|A\|_1 = \|A\|_\infty$.

Teoremă 4.0.6. *Fie $\|\cdot\|_{\mathbb{R}^{m\times n}}$ o normă matriceală indusă de normele vectoriale $\|\cdot\|_{\mathbb{R}^m}$ şi $\|\cdot\|_{\mathbb{R}^n}$. Atunci:*

1. *$\|Ax\|_{\mathbb{R}^m} \leq \|A\|_{\mathbb{R}^{m\times n}}\,\|x\|_{\mathbb{R}^n}$;*
2. *pentru o normă indusă de aceeaşi normă vectorială pe domeniu şi codomeniu, $\|I_n\| = 1$;*
3. *dacă dimensiunile sunt compatibile şi fiecare normă matricială este indusă de normele vectoriale corespunzătoare, atunci $\|AB\| \leq \|A\|\,\|B\|$.*

Demonstraţie. Din definiţia normei induse, $\|A\| = \sup_{x\neq 0} \|Ax\|/\|x\|$. Prin urmare, pentru orice $x \neq 0$ avem $\|Ax\|/\|x\| \leq \|A\|$, adică $\|Ax\| \leq \|A\|\,\|x\|$; pentru $x = 0$ relaţia este evidentă.

Pentru matricea identitate,

$$\|I_n\| = \sup_{x\neq 0} \frac{\|I_n x\|}{\|x\|} = \sup_{x\neq 0} \frac{\|x\|}{\|x\|} = 1.$$

În sfârşit, pentru orice $x \neq 0$, folosind de două ori compatibilitatea obţinem $\|ABx\| \leq \|A\|\,\|Bx\| \leq \|A\|\,\|B\|\,\|x\|$. Împărţind prin $\|x\|$ şi luând supremul după $x \neq 0$ rezultă $\|AB\| \leq \|A\|\,\|B\|$. □

Observaţi că normele p sunt submultiplicative. Mai mult, observăm că proprietatea de submultiplicativitate ar permite doar să concluzionăm că $\|I_n\| \geq 1$. Într-adevăr, $\|I_n\| = \|I_n\,I_n\| \leq \|I_n\|^2$.

Norma $\|A\|_\Delta = \max_{i=1,\ldots,n,j=1,\ldots,m} |a_{ij}|$ care nu este submultiplicativă, de asemenea, nu este o normă matriceală indusă. O normă care nu este indusă poate fi sau nu submultiplicativă. De exemplu, $\|\cdot\|_\Delta$ nu este submultiplicativă, dar norma Frobenius $\|A\|_F = (\sum_{i,j} |a_{ij}|^2)^{1/2}$ este submultiplicativă, deşi nu este o normă indusă pentru $n > 1$. Într-adevăr, dacă notăm cu b_j coloanele lui B, atunci

$$\|AB\|_F^2 = \sum_j \|Ab_j\|_2^2 \leq \|A\|_2^2 \sum_j \|b_j\|_2^2 = \|A\|_2^2\|B\|_F^2 \leq \|A\|_F^2\|B\|_F^2,$$

unde am folosit $\|A\|_2 \leq \|A\|_F$. Pe de altă parte, orice normă indusă satisface $\|I_n\| = 1$, în timp ce $\|I_n\|_F = \sqrt{n}$; de aceea, pentru $n > 1$, norma Frobenius nu poate fi indusă.

Norma spectrală pentru matrice simetrice

Teoremă 4.0.7. *Fie $A \in \mathbb{R}^{n\times n}$ simetrică. Atunci $\|A\|_2 = \rho(A)$.*

Demonstraţie. Prin teorema spectrală există o matrice ortogonală U şi o matrice diagonală $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ astfel încât $A = UDU^T$. Pentru $x \neq 0$ punem $y = U^T x$. Deoarece U este ortogonală, $\|y\|_2 = \|x\|_2$, iar

$$\frac{\|Ax\|_2^2}{\|x\|_2^2} = \frac{\|UDy\|_2^2}{\|y\|_2^2} = \frac{\sum_{i=1}^n \lambda_i^2 y_i^2}{\sum_{i=1}^n y_i^2} \leq \max_i |\lambda_i|^2.$$

Prin urmare $\|A\|_2 \leq \max_i |\lambda_i|$. Egalitatea se obţine alegând $y = e_j$, unde $|\lambda_j| = \max_i |\lambda_i|$, adică alegând $x = Ue_j$, un vector propriu corespunzător. Rezultă $\|A\|_2 = \max_i |\lambda_i| = \rho(A)$. □

De ce nu apare aici problema vectorilor proprii complecşi?

Pentru o matrice reală simetrică toate valorile proprii sunt reale şi există o bază ortonormală de vectori proprii reali. Problema vectorilor proprii complecşi apare pentru o matrice reală generală şi va fi tratată explicit mai jos.

Pentru matricea identitate rezultă imediat $\|I_n\|_2 = 1$.

Matrice nesimetrice

Definiţie 4.0.8. *Fie $A \in \mathbb{R}^{m\times n}$. Se numesc valori singulare ale matricei A, numerele reale $\sigma_i(A)$ definite prin*

$$\sigma_i(A) = \sqrt{\lambda_i(A^T A)}. \tag{4.13}$$

Dacă A este simetrică, valorile singulare sunt modulele valorilor proprii, reordonate descrescător. Mai precis, există o permutare π astfel încât

$$\sigma_i(A) = |\lambda_{\pi(i)}(A)|, \qquad \sigma_1(A) \geq \cdots \geq \sigma_n(A) \geq 0. \tag{4.14}$$

În particular, $\sigma_1(A) = \max_i |\lambda_i(A)| = \rho(A)$. Nu este corect, în general, să identificăm $\sigma_i(A)$ cu $|\lambda_i(A)|$ păstrând aceeaşi ordine a indicilor; de exemplu, pentru $A = \operatorname{diag}(1, -5)$ avem $(\lambda_1, \lambda_2) = (1, -5)$, dar $(\sigma_1, \sigma_2) = (5, 1)$.

Norma spectrală pentru matrice generale

Teoremă 4.0.9. *Fie $A \in \mathbb{R}^{m\times n}$ şi fie $\sigma_1(A)$ cea mai mare valoare singulară a sa. Atunci*

$$\|A\|_2 = \sqrt{\rho(A^T A)} = \sigma_1(A).$$

Demonstraţie. Pentru orice $x \neq 0$,

$$\frac{\|Ax\|_2^2}{\|x\|_2^2} = \frac{x^T A^T A x}{x^T x}.$$

Matricea $A^T A$ este simetrică şi pozitiv semidefinită. Prin teorema spectrală, maximul câtului Rayleigh din membrul drept este cea mai mare valoare proprie a lui $A^T A$, adică $\rho(A^T A)$. Luând supremul după $x \neq 0$ şi apoi rădăcina pătrată, obţinem $\|A\|_2 = \sqrt{\rho(A^T A)} = \sigma_1(A)$. □

Este clar că calculul lui $\|A\|_2$ este mult mai costisitor decât calculul lui $\|A\|_1$ sau al lui $\|A\|_\infty$. Cu toate acestea, dacă este necesară doar o estimare a lui $\|A\|_2$, următoarele relaţii pot fi utilizate în mod profitabil în cazul matricelor pătrate

$$\begin{aligned} &\max_{i,j} |a_{ij}| \le \|A\|_2 \le n \max |a_{ij}|, \\ &\frac{1}{\sqrt{n}} \|A\|_\infty \le \|A\|_2 \le \sqrt{n}\, \|A\|_\infty, \\ &\frac{1}{\sqrt{n}} \|A\|_1 \le \|A\|_2 \le \sqrt{n}\, \|A\|_1, \qquad \|A\|_2 \le \sqrt{\|A\|_1 \|A\|_\infty}. \end{aligned} \tag{4.15}$$

Relaţii dintre norme şi raza spectrală

Teoremă 4.0.10. *Fie $A \in \mathbb{R}^{n\times n}$ şi fie $\|\cdot\|_M$ o normă matricială consistentă cu o normă vectorială $\|\cdot\|$ pe $\mathbb{R}^n$. Atunci*

$$\rho(A) \le \|A\|_M.$$

Demonstraţie. Dacă toate valorile proprii sunt reale, demonstraţia este imediată: pentru $Av = \lambda v$, $v \neq 0$, $|\lambda|\,\|v\| = \|Av\| \le \|A\|_M\,\|v\|$.

Pentru o matrice reală generală, însă, o valoare proprie şi vectorul propriu asociat pot fi complecşi. Pentru a nu ascunde acest punct, extindem norma vectorială de pe $\mathbb{R}^n$ la $\mathbb{C}^n$ prin

$$\|z\|_{\mathbb{C}} := \max_{\theta\in[0,2\pi]} \|\operatorname{Re}(e^{i\theta} z)\|, \qquad z \in \mathbb{C}^n.$$

Aceasta este o normă pe $\mathbb{C}^n$: omogenitatea pentru scalari complecşi rezultă printr-o rotaţie a parametrului θ, iar inegalitatea triunghiului se obţine luând maximul după θ. În plus, deoarece A este reală,

$$\|Az\|_{\mathbb{C}} = \max_\theta \|A \operatorname{Re}(e^{i\theta} z)\| \le \|A\|_M \max_\theta \|\operatorname{Re}(e^{i\theta} z)\| = \|A\|_M \|z\|_{\mathbb{C}}.$$

Dacă acum $Az = \lambda z$ cu $z \in \mathbb{C}^n \setminus \{0\}$, atunci $|\lambda|\,\|z\|_{\mathbb{C}} = \|Az\|_{\mathbb{C}} \le \|A\|_M \|z\|_{\mathbb{C}}$. Deci $|\lambda| \le \|A\|_M$ pentru orice $\lambda \in \sigma(A)$, iar prin maximizare rezultă $\rho(A) \le \|A\|_M$. □

O alternativă pentru norma spectrală

Dacă folosim în particular norma euclidiană şi norma matricială indusă $\|\cdot\|_2$, complexificarea este imediată: pentru $z \in \mathbb{C}^n$ folosim $\|z\|_2 = (\sum_i |z_i|^2)^{1/2}$. Aceeaşi formulă de normă de operator rămâne valabilă pentru matricea reală A privită pe $\mathbb{C}^n$. Astfel, din $Az = \lambda z$ rezultă direct $|\lambda|\,\|z\|_2 = \|Az\|_2 \le \|A\|_2 \|z\|_2$.

În restul prelegerilor noastre, dacă nu specificăm altceva, folosim de regulă norma spectrală şi o notăm cu $\|\cdot\|$.

O normă indusă arbitrar de apropiată de raza spectrală

Teoremă 4.0.11. *Fie $A \in \mathbb{R}^{n\times n}$ şi $\varepsilon > 0$. Există o normă vectorială pe $\mathbb{R}^n$ şi norma matricială indusă asociată, notată $\|\cdot\|_{A,\varepsilon}$, astfel încât*

$$\rho(A) \le \|A\|_{A,\varepsilon} \le \rho(A) + \varepsilon.$$

Demonstraţie. Inegalitatea din stânga este consecinţa teoremei precedente. Demonstrăm existenţa normei pentru inegalitatea din dreapta. Considerăm forma Jordan a lui A peste $\mathbb{C}$: există o matrice inversabilă $S \in \mathbb{C}^{n\times n}$ astfel încât $J = S^{-1}AS$ este bloc-diagonală, fiecare bloc având forma $J_r(\lambda) = \lambda I + N$, unde N are elemente 1 pe supradiagonală şi zero în rest.

Alegem $0 < \delta < \varepsilon$ şi, pentru un bloc de dimensiune r, punem $D_r = \operatorname{diag}(1, \delta, \delta^2, \ldots, \delta^{r-1})$. Atunci $D_r^{-1} J_r(\lambda) D_r$ are λ pe diagonală şi δ pe supradiagonală. Dacă D este bloc diagonala formată din matricile D_r, rezultă

$$\|D^{-1}JD\|_\infty \le \rho(A) + \delta < \rho(A) + \varepsilon,$$

deoarece suma absolută a elementelor de pe orice rând este cel mult $|\lambda| + \delta$.

Definim acum, pentru $x \in \mathbb{R}^n$,

$$\|x\|_{A,\varepsilon} := \|D^{-1}S^{-1}x\|_\infty,$$

unde norma din dreapta este norma maxim pe $\mathbb{C}^n$. Aceasta este o normă pe spaţiul vectorial real $\mathbb{R}^n$. Mai precis, notând $y = D^{-1}S^{-1}x$, avem $D^{-1}S^{-1}Ax = D^{-1}JDy$, deci

$$\|Ax\|_{A,\varepsilon} \le \|D^{-1}JD\|_\infty \, \|x\|_{A,\varepsilon} < (\rho(A) + \varepsilon)\|x\|_{A,\varepsilon}.$$

Prin definiţia normei induse rezultă $\|A\|_{A,\varepsilon} \le \rho(A) + \varepsilon$. □

Interpretare

Raza spectrală nu este, în general, o normă matricială. Teorema afirmă însă că, pentru o matrice fixată A, putem adapta geometria spaţiului vectorial astfel încât norma de operator a lui A să fie oricât de aproape dorim de $\rho(A)$. Această observaţie este folosită în demonstrarea criteriului $A^k \to 0 \iff \rho(A) < 1$.

Analiza sensibilităţii soluţiei

La fiecare pas al GEM în urma rotunjirilor numerelor se rezolvă un sistem perturbat

$$(A + \delta A)(x + \delta\, x) = b + \delta b, \tag{4.16}$$

soluţia acestui sistem perturbat fiind perturbată faţă de soluţia sistemului de start

$$A\, x = b. \tag{4.17}$$

Ne dorim să caracterizăm perturbarea δx în funcţie de perturbările δA şi δb.

Vom folosi în continuare *numărul de condiţionare*.

Numărul de condiţionare

Numărul de condiţionare al unei matrice $A \in \mathbb{R}^{n\times n}$ este definit prin

$$K(A) = \|A\| \, \|A^{-1}\|, \tag{4.18}$$

unde $\|\cdot\|$ este o normă indusă.

Se poate observa că numărul de condiţionare depinde de norma aleasă.

Se observă însă că indiferent de norma aleasă $K(A) \ge 1$ deoarece

$$1 = \|A\, A^{-1}\| \le \|A\| \, \|A^{-1}\| = K(A).$$

Mai mult, $K(A) = K(A^{-1})$ şi $K(\alpha\, A) = K(A)$, $\forall \alpha \neq 0$.
Pentru norma $\|\cdot\|_2$ pe $\mathbb{R}^{n\times n}$, $K_2(A) = \|A\|_2\, \|A^{-1}\|_2$ este dat de

$$K_2(A) = \frac{\sigma_1(A)}{\sigma_n(A)}, \tag{4.19}$$

iar în cazul matricelor pozitiv definite

$$K_2(A) = \frac{\lambda_1(A)}{\lambda_n(A)} = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}. \tag{4.20}$$

$K_2(A)$ se numeşte numărul de condiţionare spectral.

Analiza a priori a erorii

Teoremă 4.0.12. *Fie $A \in \mathbb{R}^{n\times n}$ o matrice inversabilă şi $\delta A \in \mathbb{R}^{n\times n}$ astfel încât, într-o normă indusă,*

$$\|A^{-1}\|\, \|\delta A\| < 1. \tag{4.21}$$

Atunci, dacă $x \in \mathbb{R}^n$ este soluţie a sistemului $A\, x = b$ cu $b \in \mathbb{R}^n$ $(b \neq 0)$ şi $\delta x \in \mathbb{R}^n$ verifică

$$(A + \delta A)(x + \delta x) = b + \delta b, \tag{4.22}$$

atunci

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{K(A)}{1 - K(A)\|\delta A\|/\|A\|} \left(\frac{\|\delta b\|}{\|b\|} + \frac{\|\delta A\|}{\|A\|} \right) \tag{4.23}$$

Câteva observaţii

Condiţia $\|A^{-1}\|\, \|\delta A\| < 1$ asigură faptul că $(A + \delta A)$ rămâne inversabilă.
Dacă $\|A^{-1}\|\, \|\delta A\| < 1$, atunci $\rho(A^{-1}\delta A) < 1$.

Lemă 4.0.13. *Fie $A \in \mathbb{R}^{n\times n}$. Atunci*

$$A^k \longrightarrow 0 \quad (k \to \infty) \qquad \Longleftrightarrow \qquad \rho(A) < 1.$$

În plus, seria geometrică $\sum_{k=0}^{\infty} A^k$ converge dacă şi numai dacă $\rho(A) < 1$. În acest caz,

$$\sum_{k=0}^{\infty} A^k = (I - A)^{-1}.$$

Dacă $\|\cdot\|$ este o normă matricială indusă pentru care $\|A\| < 1$, atunci

$$\frac{1}{1 + \|A\|} \leq \|(I - A)^{-1}\| \leq \frac{1}{1 - \|A\|}.$$

Demonstraţia criteriului $A^k \to 0$

Demonstraţie. Presupunem mai întâi că $\rho(A) < 1$. Alegem $\varepsilon > 0$ astfel încât $\rho(A) + \varepsilon < 1$. Din teorema precedentă există o normă matricială indusă, pe care o notăm simplu cu $\|\cdot\|$, pentru care $\|A\| \le \rho(A) + \varepsilon =: q < 1$. Prin submultiplicativitate, $\|A^k\| \le \|A\|^k \le q^k \to 0$; deci $A^k \to 0$.

Reciproc, presupunem $A^k \to 0$. Fie $\lambda \in \sigma(A)$ şi fie $z \in \mathbb{C}^n \setminus \{0\}$ un vector propriu asociat. Aici trebuie ţinut cont de faptul că, deşi A este reală, λ şi z pot fi complecşi. Scriem $z = x + iy$, cu $x, y \in \mathbb{R}^n$. Din $A^k \to 0$ rezultă $A^k x \to 0$ şi $A^k y \to 0$, deci $A^k z \to 0$ în $\mathbb{C}^n$. Dar $A^k z = \lambda^k z$. Cum $z \neq 0$, rezultă $|\lambda|^k \to 0$, deci $|\lambda| < 1$. Aceasta este adevărat pentru orice valoare proprie, prin urmare $\rho(A) < 1$. □

Seria geometrică de matrice

Demonstraţie. Dacă $\rho(A) < 1$, alegem ca mai sus o normă indusă pentru care $\|A\| < 1$. Atunci

$$\sum_{k=0}^{\infty} \|A^k\| \le \sum_{k=0}^{\infty} \|A\|^k = \frac{1}{1 - \|A\|} < \infty,$$

deci seria $\sum_{k=0}^{\infty} A^k$ converge absolut şi, în particular, converge. Pentru sumele parţiale $S_n = I + A + \cdots + A^n$ avem identitatea $(I - A)S_n = I - A^{n+1}$. Cum $A^{n+1} \to 0$, trecând la limită obţinem $(I - A)S = I$, unde $S = \sum_{k=0}^{\infty} A^k$. În dimensiune finită aceasta implică faptul că $I - A$ este inversabilă şi $S = (I - A)^{-1}$.

Reciproc, dacă seria $\sum_{k=0}^{\infty} A^k$ converge, atunci termenul ei general trebuie să tindă la zero, adică $A^k \to 0$. Din prima parte a lemei rezultă $\rho(A) < 1$. □

Estimările pentru $(I - A)^{-1}$

Pentru limita superioară folosim direct seria geometrică:

$$\|(I - A)^{-1}\| = \left\| \sum_{k=0}^{\infty} A^k \right\| \le \sum_{k=0}^{\infty} \|A^k\| \le \sum_{k=0}^{\infty} \|A\|^k = \frac{1}{1 - \|A\|}.$$

Pentru limita inferioară, din $I = (I - A)(I - A)^{-1}$, submultiplicativitatea şi $\|I\| = 1$ obţinem

$$1 \le \|I - A\| \, \|(I - A)^{-1}\| \le (1 + \|A\|) \, \|(I - A)^{-1}\|,$$

de unde $\|(I - A)^{-1}\| \ge 1/(1 + \|A\|)$.

Cum va fi folosită această lemă?

Ea transformă o condiţie spectrală, $\rho(A) < 1$, într-o proprietate de convergenţă a puterilor şi a seriei geometrice. Exact acelaşi mecanism va apărea la metodele iterative: dacă matricea de iteraţie este B, convergenţa pentru orice aproximaţie iniţială este echivalentă cu $\rho(B) < 1$.

Revenim la teorema de demonstrat: analiza a priori a erorii

Rezultatul enunţat mai sus se demonstrează acum folosind seria geometrică de matrice.

Demonstraţia teoremei de analiză a priori

Demonstraţie. Deoarece $\|A^{-1}\delta A\| < 1$, avem că $I+A^{-1}\delta A$ este inversabilă şi din lema precedentă rezultă că

$$\|(I + A^{-1}\delta\, A)^{-1}\| \leq \frac{1}{1 - \|A^{-1}\delta A\|} \leq \frac{1}{1 - \|A^{-1}\|\,\|\delta A\|}. \tag{4.24}$$

Pe de altă parte, din

$$(A + \delta A)(x + \delta x) = b + \delta b, \tag{4.25}$$

şi $A\,x = b$ găsim

$$\delta x = (I + A^{-1}\delta A)^{-1}A^{-1}(\delta b - \delta A\, x), \tag{4.26}$$

iar trecând la normă deducem

$$\|\delta x\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\|\,\|\delta A\|}(\|\delta b\| + \|\delta A\|\ \|x\|). \tag{4.27}$$

În final, împărţind prin $\|x\|$ (care este nenul pentru că $b \neq 0$ şi A este inversabilă), apoi folosind faptul că $\|x\| \geq \frac{\|b\|}{\|A\|}$ se deduce inegalitatea dorită. □

Extensie didactică: norme, perturbaţii şi numărul de condiţionare

Ideea principală

O eroare numerică nu este un obiect absolut: trebuie măsurată. Normele oferă limbajul prin care comparăm vectori, matrice, reziduuri şi perturbaţii. Alegerea normei schimbă valorile numerice ale estimărilor, dar în dimensiune finită toate normele sunt echivalente. Pentru analiza algoritmilor vom folosi frecvent o normă matricială indusă de o normă vectorială, deoarece satisface automat $\|Ax\| \leq \|A\|\,\|x\|$.

Pentru $x \in \mathbb{R}^n$, normele uzuale sunt $\|x\|_1 = \sum_i |x_i|$, $\|x\|_2 = (\sum_i |x_i|^2)^{1/2}$ şi $\|x\|_\infty = \max_i |x_i|$. Normele matriciale induse asociate normelor 1 şi ∞ au formule uşor de calculat: $\|A\|_1$ este suma maximă pe coloană a modulelor elementelor, iar $\|A\|_\infty$ este suma maximă pe linie. Norma spectrală este $\|A\|_2 = \sqrt{\lambda_{\max}(A^TA)} = \sigma_{\max}(A)$.

Pentru o matrice simetrică, valorile singulare sunt modulele valorilor proprii, deci $\|A\|_2 = \max_i |\lambda_i(A)|$. Dacă A este SPD, această expresie devine simplu $\lambda_{\max}(A)$.

De la reziduu la eroare

Fie $Ax = b$ şi o aproximare $\widehat{x}$. Cu $r = b - A\widehat{x}$ şi $e = x - \widehat{x}$ avem $Ae = r$, deci $e = A^{-1}r$. Prin urmare,

$$\frac{\|e\|}{\|x\|} \leq \|A^{-1}\|\,\frac{\|r\|}{\|x\|}.$$

Cum $\|b\| = \|Ax\| \leq \|A\|\,\|x\|$, rezultă estimarea practică

$$\frac{\|x - \widehat{x}\|}{\|x\|} \leq \kappa(A)\,\frac{\|b - A\widehat{x}\|}{\|b\|}, \qquad \kappa(A) = \|A\|\,\|A^{-1}\|.$$

Aşadar un reziduu relativ de ordin 10^{-12} este excelent pentru o matrice cu $\kappa(A) \approx 10^2$, dar poate fi insuficient pentru una cu $\kappa(A) \approx 10^{12}$.

Exemplu: o problemă care devine sensibilă

Pentru $A_\varepsilon = \operatorname{diag}(1, \varepsilon)$ avem $\kappa_2(A_\varepsilon) = 1/\varepsilon$ dacă $0 < \varepsilon \le 1$. Sistemul cu $b = (1, \varepsilon)^T$ are soluţia $x = (1, 1)^T$. Dacă perturbăm numai a doua componentă a termenului liber la $\varepsilon + \delta$, noua soluţie are componenta a doua $1 + \delta/\varepsilon$. O perturbare absolută foarte mică poate produce deci o modificare mare atunci când ε este mic.

Acest exemplu arată şi de ce numărul de condiţionare este adimensional atunci când se folosesc erori relative. El măsoară, în prima aproximaţie, câte cifre semnificative pot fi pierdute prin sensibilitatea problemei.

Perturbarea matricei

Dacă matricea însăşi este perturbată, $(A + \Delta A)(x + \Delta x) = b + \Delta b$, analiza exactă devine mai complexă. Totuşi, atunci când $\|A^{-1}\Delta A\| < 1$, matricea perturbată rămâne inversabilă şi pot fi obţinute estimări în care apare din nou $\kappa(A)$. O matrice apropiată de una singulară are, în consecinţă, un număr de condiţionare mare.

Laborator: reziduu mic, eroare mare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi matrici cu numere de condiţionare aproximativ $10^2, 10^6, 10^{10}$. O modalitate controlată este să luaţi matrici diagonale sau să construiţi $A = Q\Sigma V^T$ cu Q, V ortogonale şi valori singulare prescrise. Pentru o soluţie $x_\star$ aleasă, calculaţi $b = Ax_\star$, perturbaţi b cu zgomot relativ mic şi măsuraţi modificarea soluţiei. Comparaţi raportul amplificării cu `cond(A)`.

Exerciţii şi verificarea înţelegerii

1. Calculaţi $\|A\|_1$ şi $\|A\|_\infty$ pentru $A = \begin{pmatrix} 1 & -2 \\ 3 & 4 \end{pmatrix}$.
2. Demonstraţi proprietatea de submultiplicativitate pentru o normă matricială indusă: $\|AB\| \le \|A\|\,\|B\|$.
3. Pentru o matrice ortogonală Q, arătaţi că $\|Q\|_2 = 1$ şi $\kappa_2(Q) = 1$.
4. Dacă A este SPD, demonstraţi că $\kappa_2(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$.
5. Construiţi două aproximări $\widehat{x}_1, \widehat{x}_2$ pentru acelaşi sistem astfel încât prima să aibă reziduu mai mic, dar eroare într-o normă aleasă să nu fie neapărat mai mică. Discutaţi rolul direcţiei reziduului.
6. Explicaţi de ce normalizarea sistemului prin înmulţirea unei ecuaţii cu un factor mare poate modifica unele norme şi comportamentul algoritmului, chiar dacă mulţimea soluţiilor exacte rămâne aceeaşi.

Mini-proiect: matricea Hilbert

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Pentru $n = 2, 3, \ldots, 14$, construiţi matricea Hilbert H_n, rezolvaţi $H_n x = b$ cu o soluţie exactă $x_\star = \mathbf{1}$, apoi calculaţi reziduul relativ, eroarea relativă şi `cond(H)`. Reprezentaţi cele trei mărimi pe scară logaritmică. Explicaţi de ce un algoritm poate produce un reziduu foarte mic şi totuşi o soluţie cu puţine cifre corecte.

Rezolvare orientativă

Calculul normelor şi interpretarea lor

Pentru $A = \begin{pmatrix} 1 & -2 \\ 3 & 4 \end{pmatrix}$, suma modulelor pe prima coloană este $1 + 3 = 4$, iar pe a doua $2 + 4 = 6$, deci $\|A\|_1 = 6$. Pe linii obţinem $1 + 2 = 3$ şi $3 + 4 = 7$, deci $\|A\|_\infty = 7$. Aceste norme sunt ieftin de calculat şi au interpretări directe.
Pentru $A = \operatorname{diag}(2, 0.01)$, norma spectrală este 2, norma lui A^{-1} este 100, deci $\kappa_2(A) = 200$. Dacă datele conţin o perturbare relativă de ordin 10^{-5} orientată pe direcţia celei mai mici valori singulare, eroarea relativă a soluţiei poate ajunge la ordin $2 \cdot 10^{-3}$.

Situaţie practică: prag de încredere

Să presupunem că senzorii furnizează date cu aproximativ şase cifre semnificative, iar modelul liniar are $\kappa(A) \approx 10^8$. Chiar cu un algoritm stabil, nu este realist să aşteptăm opt cifre corecte în soluţie. Înainte de a "îmbunătăţi codul", trebuie analizată condiţionarea şi, eventual, reformulat modelul sau introdusă regularizare.

Exemplu rezolvat: reziduu minuscul, eroare vizibilă

Fie

$$A_\delta = \begin{pmatrix} 1 & 1 \\ 1 & 1+\delta \end{pmatrix}, \qquad b = A_\delta \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2+\delta \end{pmatrix}.$$

Perturbăm numai a doua componentă a termenului liber cu ε şi rezolvăm $(A_\delta)(x+e) = b + (0, \varepsilon)^T$. Eroarea satisface $A_\delta e = (0, \varepsilon)^T$. Din prima ecuaţie $e_1 + e_2 = 0$, iar scăzând prima ecuaţie din a doua obţinem $\delta e_2 = \varepsilon$. Prin urmare

$$e_2 = \frac{\varepsilon}{\delta}, \qquad e_1 = -\frac{\varepsilon}{\delta}.$$

Dacă $\delta = 10^{-6}$ şi $\varepsilon = 10^{-8}$, o perturbare de numai 10^{-8} produce componente ale erorii de ordin 10^{-2}. Mecanismul nu ţine de algoritm: matricea este aproape singulară, iar cele două ecuaţii sunt aproape dependente.

Întrebarea corectă despre un rezultat numeric

Nu este suficient să întrebăm „cât de mic este reziduul?”. Trebuie să întrebăm şi „cât de sensibilă este problema?”. O soluţie calculată poate avea reziduu aproape de precizia maşinii şi totuşi să nu aibă multe cifre corecte dacă $\kappa(A)$ este mare. De aceea rapoartele numerice serioase includ reziduul, o estimare a condiţionării şi, când este posibil, o eroare înainte faţă de o soluţie de referinţă.

Demonstraţie detaliată: estimarea erorii din reziduu

Dacă $Ax = b$ şi $\widehat{x}$ este aproximaţia calculată, definim $r = b - A\widehat{x}$ şi $e = x - \widehat{x}$. Atunci $Ae = r$, deci $e = A^{-1}r$. Pentru orice normă matricială subordonată, $\|e\| \le \|A^{-1}\|\,\|r\|$. Pe de altă parte, din $b = Ax$ avem $\|b\| \le \|A\|\,\|x\|$, deci $1/\|x\| \le \|A\|/\|b\|$. Înmulţind cele două inegalităţi obţinem

$$\frac{\|e\|}{\|x\|} \le \|A\|\|A^{-1}\|\frac{\|r\|}{\|b\|} = \kappa(A)\frac{\|r\|}{\|b\|}.$$

Această derivare arată exact unde intră numărul de condiţionare şi de ce el amplifică efectul unui reziduu relativ.

Sarcină practică: o hartă eroare–reziduu–condiţionare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi matrici cu valori singulare prescrise $1, 10^{-1}, \dots, 10^{-p}$, alegând factori ortogonali aleatori. Pentru fiecare matrice generaţi $x_\star$, formaţi $b = Ax_\star$ şi rezolvaţi sistemul. Raportaţi $\kappa_2(A)$, eroarea relativă şi reziduul relativ. Reprezentaţi eroarea în funcţie de $\kappa_2(A)$ şi comparaţi cu ordinul de mărime sugerat de $\kappa(A)\,u_{mach}$.

Exerciţii suplimentare

1. Calculaţi κ_2 pentru $A = \mathrm{diag}(1, 10^{-4}, 10^{-7})$.
2. Pentru exemplul A_δ, calculaţi determinantul şi explicaţi de ce tinde la zero când $\delta \to 0$.
3. Arătaţi că $\kappa_2(Q) = 1$ pentru orice matrice ortogonală Q.
4. Dacă înmulţim matricea cu un scalar nenul, se schimbă $\kappa(A)$? Demonstraţi.
5. Construiţi două aproximaţii cu acelaşi reziduu relativ, dar pentru două matrici cu condiţionări foarte diferite. Comparaţi erorile înainte.

4.1 Îmbunătăţirea acurateţei eliminării gaussiene

Îmbunătăţirea acurateţei eliminării gaussiene

După cum s-a menţionat anterior, dacă matricea sistemului este prost condiţionată, soluţia generată de GEM ar putea fi inexactă, chiar dacă reziduul său la pasul i, adică $r^{(i)} = b - Ax^{(i)}$,

este mic. În această secţiune, menţionăm două tehnici de îmbunătăţire a acurateţei soluţiei calculate de GEM.

Scalarea problemei

În cazul în care intrările din A variază foarte mult ca mărime, este probabil ca în timpul procesului de eliminare intrările mari să fie însumate cu intrările mici, având drept consecinţă apariţia erorilor de rotunjire. Un remediu constă în efectuarea unei redimensionări a matricei A înainte de a se efectua eliminarea.

Scalarea pe rând a lui A constă în găsirea unei matrice diagonale nesingulare D_1 astfel încât intrările diagonale ale lui D_1A să aibă acelaşi ordin de mărime (aceeaşi dimensiune). Sistemul liniar $Ax = b$ se transformă în

$$D_1 A\, x = D_1 b. \tag{4.28}$$

Atunci când atât liniile, cât şi coloanele lui A trebuie să fie scalate, versiunea scalată a sistemului devine

$$(D_1 A\, D_2) y = D_1 b \qquad \text{cu} \qquad y = D_2^{-1} x, \tag{4.29}$$

presupunând, de asemenea, că D_2 este inversabil. Matricea D_1 redimensionează ecuaţiile, în timp ce D_2 redimensionează necunoscutele.

Scalarea problemei

Observaţi că, pentru a preveni erorile de rotunjire, matricile de scalare sunt alese sub forma

$$D_1 = \text{diag}(\beta^{r_1}, ..., \beta^{r_n}), D_2 = \text{diag}(\beta^{c_1}, ..., \beta^{c_n}) \tag{4.30}$$

unde β este baza aritmeticii în virgulă mobilă utilizată, iar exponenţii $r_1, \cdots, r_n, c_1, \cdots, c_n$ trebuie determinaţi.

Rafinare iterativă

Rafinarea iterativă este o tehnică de îmbunătăţire a acurateţei unei soluţii obţinute printr-o metodă directă. Să presupunem că sistemul liniar $Ax = b$ a fost rezolvat cu ajutorul factorizării LU (cu pivotare parţială sau completă) şi să notăm cu $x(0)$ soluţia calculată. După ce s-a fixat o toleranţă de eroare, tol, rafinarea iterativă se desfăşoară astfel: pentru $i = 0, 1, \cdots$, până la convergenţă:

1. se calculează reziduul $r^{(i)} = b - Ax^{(i)}$, de preferinţă într-o precizie mai mare decât cea folosită pentru factorii LU;
2. se rezolvă sistemul liniar $Az^{(i)} = r^{(i)}$ folosind factorii L, U (şi permutarea, dacă este cazul) calculaţi o singură dată pentru A;
3. actualizaţi soluţia stabilind $x^{(i+1)} = x^{(i)} + z^{(i)}$;
4. dacă $\|z^{(i)}\|/\|x^{(i+1)}\| < tol$, atunci încheiem procesul returnând soluţia $x^{(i+1)}$. În caz contrar, algoritmul reîncepe de la pasul 1.

În absenţa erorilor de rotunjire, procesul s-ar opri la primul pas, producând soluţia exactă.

Completare didactică: scalare şi rafinare iterativă

Două tehnici simple pot îmbunătăţi substanţial comportamentul unei rezolvări directe: scalarea problemei înainte de factorizare şi rafinarea iterativă după obţinerea unei prime soluţii. Ele nu schimbă metoda de bază, ci încearcă să o folosească într-un regim numeric mai favorabil.

Scalarea

Dacă o linie a lui A conţine valori de ordin 10^{-8} şi alta valori de ordin 10^{8}, comparaţia directă a elementelor şi alegerea pivoturilor pot fi influenţate de unităţile de măsură. O scalare pe linii înlocuieşte $Ax = b$ cu $D_r A x = D_r b$, unde D_r este diagonală şi inversabilă. Soluţia exactă nu se schimbă. O scalare pe coloane, $A D_c y = b$ cu $x = D_c y$, schimbă variabilele şi trebuie interpretată corespunzător.

Scopul nu este să facem toate elementele egale, ci să evităm diferenţe artificiale de scară. Scalarea nu garantează un număr de condiţionare mic şi nu repară o problemă intrinsec aproape singulară.

Rafinarea iterativă

Presupunem că am calculat o factorizare stabilă a lui A şi o primă aproximaţie $x^{(0)}$. La pasul k calculăm reziduul $r^{(k)} = b - A x^{(k)}$, rezolvăm sistemul de corecţie $A d^{(k)} = r^{(k)}$ folosind aceeaşi factorizare şi actualizăm $x^{(k+1)} = x^{(k)} + d^{(k)}$. În aritmetică exactă, dacă sistemul de corecţie este rezolvat exact, un singur pas ar fi suficient. În aritmetică finită, repetarea poate recupera cifre pierdute.

De ce poate funcţiona?

Nu refactorizăm matricea. Folosim factorizarea existentă pentru a rezolva o problemă a cărei soluţie este tocmai eroarea curentă. Dacă reziduul este calculat cu acurateţe mai bună decât soluţia iniţială, corecţia poate recupera informaţie semnificativă. În aritmetică cu precizie mixtă, avantajul poate fi şi mai mare: factorizarea poate fi ieftină, iar reziduul calculat într-o precizie mai mare.

Laborator: rafinare iterativă

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi:

```
[L,U,P] = lu(A);
x = U \ (L \ (P*b));
for k = 1:maxit
    r = b - A*x;
    d = U \ (L \ (P*r));
    xnew = x + d;
    if norm(d) <= tol*(1+norm(xnew)), break; end
    x = xnew;
end
```

Testaţi pe o matrice moderat rău condiţionată. Urmăriţi separat norma reziduului şi eroarea faţă de o soluţie exactă construită. Încercaţi şi calcularea reziduului într-o precizie mai mare, dacă mediul folosit permite.

Exerciţii şi verificarea înţelegerii

1. Demonstraţi că în aritmetică exactă, dacă d rezolvă exact $Ad = b - Ax$, atunci $x + d$ este soluţia exactă.
2. Propuneţi două criterii de oprire: unul bazat pe corecţie şi unul bazat pe reziduul relativ. Discutaţi avantajele fiecăruia.
3. Pentru o scalare pe coloane $x = D_c y$, scrieţi sistemul satisfăcut de y şi formula de recuperare a lui x.
4. Daţi un exemplu în care scalarea pe linii modifică foarte mult mărimea elementelor, dar nu schimbă deloc soluţia exactă.
5. De ce este important pentru costul rafinării iterative să refolosim factorii LU?

Studiu practic: precizie mixtă

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Dacă mediul permite tipuri de precizie diferită, calculaţi factorizarea într-o precizie mai redusă şi reziduul într-o precizie mai mare. Investigaţi pentru ce niveluri de condiţionare rafinarea recuperează o soluţie apropiată de cea obţinută direct în precizie înaltă. Dacă nu aveţi acces la precizie mixtă, simulaţi o precizie redusă prin rotunjirea deliberată a factorilor.

Rezolvare orientativă

De ce o corecţie exactă ar rezolva problema într-un pas

Fie $x^{(0)}$ orice aproximaţie şi $r = b - Ax^{(0)}$. Dacă rezolvăm exact $Ad = r$ şi punem $x^{(1)} = x^{(0)} + d$, atunci $Ax^{(1)} = Ax^{(0)} + Ad = Ax^{(0)} + b - Ax^{(0)} = b$. Prin urmare $x^{(1)}$ este soluţia exactă. Faptul că în practică sunt necesare mai multe iteraţii provine exclusiv din aritmetica finită şi din faptul că sistemul de corecţie este rezolvat aproximativ.

Situaţie practică: oprirea rafinării

Dacă norma corecţiei scade până la nivelul $\|d^{(k)}\| \approx u\|x^{(k)}\|$, unde u este precizia maşinii, adăugarea corecţiei poate să nu mai schimbe reprezentarea lui x. Continuarea iteraţiilor nu mai aduce informaţie. Un criteriu bun trebuie să urmărească atât reziduul, cât şi mărimea corecţiei, şi să impună un număr maxim de paşi.

Exemplu rezolvat: o iteraţie de rafinare

Considerăm

$$A = \begin{pmatrix} 10 & 7 \\ 3 & 2 \end{pmatrix}, \qquad b = \begin{pmatrix} 17 \\ 5 \end{pmatrix},$$

cu soluţia exactă $x_\star = (1,1)^T$. Să presupunem că o primă rezolvare produce $\widehat{x}^{(0)} = (0.99, 1.02)^T$. Reziduul este

$$r^{(0)} = b - A\widehat{x}^{(0)} = \begin{pmatrix} -0.04 \\ -0.01 \end{pmatrix}.$$

Rezolvăm sistemul de corecţie $A\delta x^{(0)} = r^{(0)}$ folosind factorii deja calculaţi ai lui A şi obţinem $\delta x^{(0)} = (0.01, -0.02)^T$. Actualizarea $\widehat{x}^{(1)} = \widehat{x}^{(0)} + \delta x^{(0)}$ recuperează aici exact $(1,1)^T$.

În aritmetică reală exactă, dacă sistemul de corecţie ar fi rezolvat exact, o singură corecţie ar recupera soluţia. Utilitatea rafinării apare tocmai deoarece factorii şi rezolvările sunt afectate de erori de rotunjire. Un avantaj este că reziduul poate fi calculat într-o precizie mai mare decât cea folosită la factorizare, ceea ce stă la baza variantelor moderne de mixed precision iterative refinement.

Sarcină practică: rafinare iterativă

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Factorizaţi o dată A în precizie standard. Calculaţi o aproximaţie iniţială şi apoi repetaţi: $r_k = b - Ax_k$, rezolvaţi $A\delta_k = r_k$ cu factorii existenţi şi puneţi $x_{k+1} = x_k + \delta_k$. Opriţi când reziduul nu mai scade. Testaţi matrici Hilbert şi matrici bine condiţionate. Dacă mediul permite, calculaţi reziduul în precizie extinsă şi comparaţi rezultatele.

Exerciţii suplimentare

1. Refaceţi calculul exemplului de mai sus şi verificaţi exact vectorul de corecţie.
2. Explicaţi de ce nu refactorizăm A la fiecare pas de rafinare.
3. Ce se întâmplă dacă matricea este atât de rău condiţionată încât rezolvarea sistemului de corecţie nu mai furnizează o direcţie utilă?
4. De ce calculul reziduului în precizie mai mare poate îmbunătăţi rafinarea?
5. Propuneţi un criteriu de oprire bazat simultan pe corecţie şi reziduu, nu numai pe numărul de iteraţii.

Fişă de seminar: eroare, condiţionare şi stabilitate

Problema rezolvată 1: perturbarea termenului liber

Dacă $Ax = b$ şi $A(x + \Delta x) = b + \Delta b$, atunci $A\Delta x = \Delta b$. Pentru A nesingulară, $\Delta x = A^{-1}\Delta b$. Rezultă $\|\Delta x\|/\|x\| \le \kappa(A)\, \|\Delta b\|/\|b\|$. Această estimare este de tip worst-case: spune cât de mare poate fi amplificarea pentru o perturbare nefavorabil orientată. Pentru o perturbare aleatoare, eroarea observată poate fi mult mai mică. Nu trebuie confundat un majorant cu o predicţie exactă a fiecărui experiment.

Problema rezolvată 2: scalarea nu schimbă condiţionarea printr-un scalar

Pentru orice $\alpha \neq 0$, avem $(\alpha A)^{-1} = \alpha^{-1} A^{-1}$. Într-o normă absolut omogenă, $\kappa(\alpha A) = \|\alpha A\|\|\alpha^{-1} A^{-1}\| = \kappa(A)$. Aşadar simpla înmulţire a întregului sistem cu 10^6 nu schimbă condiţionarea. În schimb, scalarea diferită a liniilor sau coloanelor poate schimba condiţionarea numerică, pentru că nu este o multiplicare uniformă.

Backward error

Un algoritm este backward stable dacă rezultatul calculat poate fi interpretat ca soluţia exactă a unei probleme foarte apropiate de cea iniţială. Aceasta este o noţiune de proprietate a algoritmului. Pentru a transforma backward error mic într-o eroare înainte mică avem nevoie şi de o problemă bine condiţionată. Cele două concepte sunt complementare.

Laborator integrator 4: matrice Hilbert

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru $n = 2, \dots, 14$, construiţi matricea Hilbert $H_n = (1/(i+j-1))$. Alegeţi $x_\star = \mathbf{1}$ şi $b = H_n x_\star$. Rezolvaţi cu solverul standard, raportaţi $\kappa_2(H_n)$, reziduul relativ şi eroarea înainte. Reprezentaţi toate trei pe scară logaritmică. Aplicaţi apoi rafinarea iterativă şi observaţi până la ce dimensiune aduce o îmbunătăţire.

Set de probleme

1. Demonstraţi majorantul perturbării termenului liber.
2. Derivaţi o estimare de prim ordin pentru perturbarea simultană a lui A şi b.
3. Arătaţi că pentru o matrice diagonală pozitivă, $\kappa_2(A) = \max a_{ii} / \min a_{ii}$.
4. De ce determinantul foarte mic nu este echivalent cu număr de condiţionare mare dacă matricea este rescalată?
5. Daţi un exemplu de matrice cu determinant 1 şi condiţionare arbitrar de mare.
6. Ce informaţie oferă reziduul normalizat prin $\|A\|\|x\| + \|b\|$ faţă de simplul $\|r\|/\|b\|$?
7. Explicaţi de ce rafinarea iterativă poate folosi aceeaşi factorizare.
8. Formulaţi un protocol minim pentru raportarea calităţii unui sistem liniar rezolvat numeric.

Proiect de capitol: raport de sensibilitate

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Alegeţi trei familii de matrici cu condiţionări diferite. Pentru fiecare produceţi 100 de perturbaţii ale termenului liber de aceeaşi normă relativă şi reprezentaţi distribuţia erorilor soluţiei. Comparaţi cel mai rău caz observat cu majorantul dat de $\kappa(A)$. Discutaţi diferenţa dintre direcţia singulară cea mai sensibilă şi o perturbare aleatoare.

Sarcini de aprofundare: condiţionare, stabilitate şi rafinare

Probleme de aprofundare: sensibilitatea problemei versus comportamentul algoritmului

1. Pentru $A = \operatorname{diag}(1, 10^{-4}, 10^{-8})$, calculaţi $\kappa_2(A)$ şi construiţi doi vectori de perturbare ai lui b cu aceeaşi normă, dar care produc amplificări foarte diferite în soluţie. Explicaţi rolul direcţiei perturbării.
2. Demonstraţi că $\kappa_2(Q) = 1$ pentru o matrice ortogonală şi că $\kappa_2(Q_1 A Q_2) = \kappa_2(A)$ dacă Q_1, Q_2 sunt ortogonale.
3. Pentru $A_\varepsilon = \begin{pmatrix} 1 & 1 \\ 1 & 1+\varepsilon \end{pmatrix}$, analizaţi determinantul, valorile proprii ale lui $A_\varepsilon^T A_\varepsilon$ şi numărul de condiţionare $\kappa_2(A_\varepsilon)$ când $\varepsilon \to 0$. Comparaţi concluziile folosind definiţia normei spectrale din capitol.
4. Deduceţi o estimare a erorii înainte pornind de la reziduul $r = b - A\widehat{x}$: $\|x - \widehat{x}\|/\|x\| \lesssim \kappa(A)\|r\|/\|b\|$. Precizaţi ipotezele şi locurile în care apar inegalităţi.
5. Daţi un exemplu în care reziduul relativ este de ordinul preciziei maşinii, dar eroarea relativă este mare. Explicaţi de ce aceasta nu demonstrează instabilitatea algoritmului.
6. Definiţi conceptual eroarea înainte şi eroarea înapoi. Pentru o aproximaţie $\widehat{x}$, construiţi o perturbare δb astfel încât $A\widehat{x} = b + \delta b$ şi exprimaţi δb prin reziduu.
7. Arătaţi că înmulţirea matricei cu un scalar nenul nu modifică $\kappa(A)$ într-o normă indusă, dar scalarea diferită a liniilor sau coloanelor o poate modifica.
8. Pentru matricea Hilbert H_n, explicaţi de ce alegerea $b = H_n \mathbf{1}$ este mai bună pentru un experiment de condiţionare decât alegerea unui b arbitrar dacă dorim să cunoaştem exact eroarea înainte.
9. Analizaţi rafinarea iterativă în forma $r_k = b - Ax_k$, $Ad_k = r_k$, $x_{k+1} = x_k + d_k$. În aritmetică exactă, ce s-ar întâmpla după un singur pas dacă sistemul pentru d_k ar fi rezolvat exact?
10. Explicaţi de ce calcularea reziduului într-o precizie mai mare decât cea folosită la factorizare poate face rafinarea iterativă mult mai eficientă.
11. Construiţi un exemplu în care schimbarea unităţilor unei variabile produce o matrice mai prost scalată. Propuneţi o rescalare diagonală şi măsuraţi efectul asupra condiţionării.
12. Distingeţi prin câte o propoziţie: problemă rău condiţionată, algoritm instabil, implementare defectuoasă şi criteriu de oprire neadecvat.

Laborator integrator: hartă condiţionare–eroare–precizie

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă

este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Separaţi experimental sensibilitatea problemei de efectul preciziei numerice.

1. Construiţi matrici simetrice pozitiv definite $A = QDQ^T$, unde Q este ortogonală şi $D = \mathrm{diag}(1, 10^{-p/(n-1)}, \ldots, 10^{-p})$ pentru $p = 2, 4, 6, 8, 10, 12$. Această construcţie foloseşte numai teorema spectrală deja introdusă.
2. Alegeţi $x_\star$ aleator, formaţi $b = Ax_\star$ şi rezolvaţi în dublă precizie. Raportaţi $\kappa_2(A)$, reziduul relativ şi eroarea relativă.
3. Repetaţi calculul după conversia datelor la `single`. Comparaţi ordinul erorii cu $\kappa(A)u_{mach}$ pentru cele două precizii.
4. Pentru fiecare caz perturbaţi b cu un vector de normă relativă 10^{-8}, o dată într-o direcţie aleatoare şi o dată pe direcţia vectorului propriu asociat celei mai mici valori proprii a lui A. Comparaţi amplificarea.
5. Implementaţi rafinarea iterativă folosind factorii calculaţi o singură dată. Salvaţi istoria reziduului şi a erorii.
6. Pentru o matrice Hilbert, comparaţi rezolvarea directă, rafinarea iterativă şi o versiune cu scalare diagonală simplă.
7. Reprezentaţi grafic eroarea în funcţie de $\kappa_2(A)$ pe scară logaritmică şi marcaţi regimul în care se pierd progresiv cifre semnificative.
8. Formulaţi o concluzie distinctă pentru: condiţionarea problemei, stabilitatea solverului şi utilitatea rafinării.

Mini-proiect: interpolare polinomială sensibilă

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Studiaţi sisteme Vandermonde pătratice construite pe noduri aflate într-un interval nefavorabil, de exemplu $[100, 101]$, înainte de introducerea formală a problemei celor mai mici pătrate.

1. **Modelare.** Pentru gradele $3, 5, 8$, alegeţi câte $n + 1$ noduri şi construiţi sistemul pătratic care determină polinomul interpolator în baza monomială.
2. **Diagnostic.** Calculaţi condiţionarea matricei Vandermonde înainte şi după transformarea variabilei la intervalul $[-1, 1]$.
3. **Calcul manual.** Pentru trei puncte şi gradul 2, scrieţi sistemul complet şi efectuaţi manual rescalarea variabilei.
4. **Algoritm.** Rezolvaţi sistemele printr-o metodă directă adecvată şi comparaţi coeficienţii în cele două parametrizări.
5. **Perturbare.** Modificaţi valorile nodale cu perturbaţii relative controlate şi măsuraţi variaţia coeficienţilor şi variaţia valorilor polinomului pe interval.
6. **Validare.** Evaluaţi interpolatoarele pe o grilă fină şi comparaţi două mărimi diferite:

eroarea în coeficienţi şi eroarea funcţiei.

7. **Interpretare.** Discutaţi de ce un vector de coeficienţi foarte sensibil nu implică automat o eroare de aceeaşi mărime în valorile funcţiei pe interval.
8. **Raport.** Includeţi condiţionare, reziduuri, erori şi o concluzie despre rolul scalării înaintea algoritmului propriu-zis.

Test de capitol

Test – Norme, condiţionare şi analiza erorii

1. Calcul explicit. Se consideră

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 1/10 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 1/10 \end{pmatrix}.$$

a) Calculaţi $\|A\|_1$, $\|A\|_\infty$ şi, folosind rezultatele capitolului, $\|A\|_2$.

b) Calculaţi numărul de condiţionare $\kappa_2(A)$.

c) Rezolvaţi $Ax = b$, apoi înlocuiţi b cu $\widetilde{b} = (2,\, 11/100)^T$. Comparaţi variaţia relativă a datelor cu variaţia relativă a soluţiei şi interpretaţi rezultatul.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi.

A) numărul de condiţionare descrie sensibilitatea problemei la perturbaţii;

B) un număr de condiţionare mare dovedeşte automat că programul conţine o eroare;

C) reziduul mic şi eroarea mică sunt noţiuni diferite;

D) pentru orice matrice inversabilă $\kappa(A) \geq 1$ într-o normă indusă.

Capitolul 5

Metode iterative pentru sisteme liniare

Despre convergenţa metodelor iterative

Ideea de bază a metodelor iterative este de a construi o secvenţă de vectori $x^{(k)}$ care se bucură de proprietatea de convergenţă

$$x = \lim_{k \to \infty} x^{(k)}, \tag{5.1}$$

unde x este soluţia pentru

$$A\,x = b. \tag{5.2}$$

În practică, procesul iterativ se opreşte la valoarea minimă a lui n astfel încât $\|x^{(n)} - x\| < \varepsilon$, unde ε este o toleranţă fixă şi $\|\cdot\|$ este orice normă vectorială convenabilă.

Cu toate acestea, deoarece soluţia exactă nu este, evident, disponibilă, este necesar să se introducă criterii de oprire adecvate pentru a monitoriza convergenţa iteraţiei.

Clasificarea metodelor iterative

Iteraţiile introduse mai sus sunt cazuri speciale de metodelor iterative de forma

$$x^{(0)} = f_0(A, b), \tag{5.3}$$

$$x^{(n+1)} = f_{n+1}(x^{(n)}, x^{(n-1)}, \cdots, x^{(n-m)}, A, b), \qquad \text{pentru} \quad n \geq m, \tag{5.4}$$

unde f_i şi $x^{(m)}, \cdots, x^{(1)}$ sunt funcţii şi, respectiv, vectori daţi.

Numărul de paşi de care depinde iteraţia curentă se numeşte **ordinul metodei**.

Dacă funcţiile f_i sunt independente de indicele iteraţiei i, metoda se numeşte **staţionară**, în caz contrar este **nestaţionară**.

În sfârşit, dacă f_i depinde liniar de $x^{(0)}, \cdots, x^{(m)}$, metoda se numeşte **liniară**, în caz contrar este **neliniară**.

În lumina acestor definiţii, metodele pe care le vom considera sunt metode iterative liniare staţionare de ordinul întâi.

Luăm în considerare metodele iterative de forma

$$\text{dată} \quad x^{(0)}, \qquad x^{(k+1)} = Bx^{(k)} + f, \qquad k \geq 0, \tag{5.5}$$

unde $B \in \mathbb{R}^{n \times n}$ este matricea de iteraţie, iar $f \in \mathbb{R}^n$ este un vector determinat de partea dreaptă b.

Definiţie 5.0.1. *Se spune că o metodă iterativă de forma (5.5) este conformă cu $Ax = b$ dacă f şi B sunt astfel încât $x = Bx + f$. În mod echivalent,*

$$f = (I - B)A^{-1}b. \tag{5.6}$$

După ce am notat cu

$$e^{(k)} = x^{(k)} - x \tag{5.7}$$

eroarea la al k-lea pas al iteraţiei, condiţia de convergenţă se rezumă la a cere ca $\lim_{k\to\infty} e^{(k)} = 0$ pentru orice alegere a datelor iniţiale. $x^{(0)}$ (adesea numită presupunere iniţială).

Consistenţa nu este suficientă pentru a asigura convergenţa iteraţiei.

Teoremă 5.0.2. *Fie (5.5) o metodă consistentă. Atunci, secvenţa de vectori $x^{(k)}$ converge către soluţia lui $Ax = b$ pentru orice alegere a lui $x^{(0)}$ dacă şi numai dacă $\rho(B) < 1$.*

Demonstraţie. Din consistenţă avem $x = Bx + f$. Scăzând această relaţie din iteraţie obţinem

$$e^{(k+1)} = Be^{(k)}, \qquad e^{(k)} := x^{(k)} - x,$$

şi prin inducţie $e^{(k)} = B^k e^{(0)}$.

Dacă $\rho(B) < 1$, lema $B^k \to 0 \iff \rho(B) < 1$ arată că $B^k \to 0$, deci $e^{(k)} \to 0$ pentru orice eroare iniţială.

Reciproc, presupunem că iteraţia converge la x pentru orice $x^{(0)}$. Atunci $B^k e^{(0)} \to 0$ pentru orice $e^{(0)} \in \mathbb{R}^n$, ceea ce implică $B^k \to 0$ (de exemplu aplicând relaţia vectorilor bazei canonice). Lema citată dă $\rho(B) < 1$. Acest argument acoperă şi cazul valorilor proprii complexe şi cazul limită $\rho(B) = 1$, care nu pot fi excluse printr-un argument bazat doar pe o valoare proprie reală. □

Metode iterative liniare

O tehnică generală de concepere a unor metode iterative liniare consistente se bazează pe o *scriere aditivă* a matricei A de forma $A = P - N$, unde P şi N sunt două matrici adecvate şi P este nesingulară.

Din motive care vor fi clarificate în secţiunile ulterioare, P se numeşte *matrice de precondiţionare*. În acest capitol litera P are acest sens local şi nu trebuie confundată cu matricea de permutare din capitolul 3 sau cu matricea stocastică PageRank din capitolul 14.

Mai exact, dat fiind $x^{(0)}$, se poate calcula $x^{(k)}$ pentru $k \geq 1$, rezolvând sistemele

$$Px^{(k+1)} = Nx^{(k)} + b \quad \Leftrightarrow \quad x^{(k+1)} = P^{-1}Nx^{(k)} + P^{-1}b, \quad k \geq 0. \tag{5.8}$$

Alternativ, iteraţia poate fi scrisă sub forma

$$x^{(k+1)} = x^{(k)} + P^{-1}r^{(k)} \quad k \geq 0. \tag{5.9}$$

unde

$$r^{(k)} = b - Ax^{(k)} \tag{5.10}$$

reprezintă **vectorul rezidual** la pasul k.

$$x^{(k+1)} = x^{(k)} + P^{-1}r^{(k)} \quad k \geq 0. \tag{5.11}$$

unde

$$r^{(k)} = b - Ax^{(k)} \tag{5.12}$$

reprezintă **vectorul rezidual** la pasul k.

Relaţia de mai sus evidenţiază faptul că un sistem liniar, cu matricea coeficienţilor P, trebuie rezolvat pentru a actualiza soluţia la pasul $k+1$.

Prin urmare, P, pe lângă faptul că nu este singulară, trebuie să fie uşor de inversat, pentru a menţine costul total de calcul scăzut. (Observaţi că, dacă P ar fi egal cu A şi $N = 0$, metoda ar converge într-o singură iteraţie, dar la acelaşi cost ca o metodă directă).

Două rezultate generale permit analiza unitară a mai multor metode staţionare. Pentru o matrice simetrică pozitiv definită M folosim norma vectorială $\|x\|_M = (x^T Mx)^{1/2}$ şi norma de operator indusă

$$\|B\|_M = \sup_{x \neq 0} \frac{\|Bx\|_M}{\|x\|_M} = \left\|M^{1/2} B M^{-1/2}\right\|_2.$$

Teoremă 5.0.3. *Fie $A = P - N$, cu A şi P simetrice şi pozitiv definite, şi fie $B = P^{-1}N = I - P^{-1}A$. Dacă $2P - A$ este pozitiv definită, atunci iteraţia este convergentă pentru orice $x^{(0)}$ şi*

$$\rho(B) = \|B\|_A = \|B\|_P < 1.$$

În plus, pentru orice eroare nenulă $e^{(k)}$ avem $\|e^{(k+1)}\|_A < \|e^{(k)}\|_A$ şi $\|e^{(k+1)}\|_P < \|e^{(k)}\|_P$.

Demonstraţie. Deoarece A şi P sunt SPD, matricea $P^{-1}A$ este similară cu matricea simetrică pozitiv definită $P^{-1/2}AP^{-1/2}$. Prin urmare, valorile sale proprii μ_i sunt reale şi pozitive. Ipoteza $2P - A \succ 0$ este echivalentă cu

$$2I - P^{-1/2}AP^{-1/2} \succ 0,$$

deci $0 < \mu_i < 2$. Valorile proprii ale lui $B = I - P^{-1}A$ sunt $1 - \mu_i$, astfel încât $\rho(B) < 1$.

Mai mult, $PB = P - A$ este simetrică, deci B este autoadjunctă pentru produsul scalar $\langle x, y\rangle_P = x^T Py$. În consecinţă, după transformarea $P^{1/2}BP^{-1/2}$ obţinem o matrice simetrică, iar $\|B\|_P = \rho(B)$. Analog, $AB = A - AP^{-1}A$ este simetrică, deci B este autoadjunctă şi în produsul scalar asociat lui A, de unde $\|B\|_A = \rho(B)$. Cum $e^{(k+1)} = Be^{(k)}$, rezultă pentru $e^{(k)} \neq 0$

$$\|e^{(k+1)}\|_M \leq \|B\|_M \|e^{(k)}\|_M < \|e^{(k)}\|_M, \qquad M \in \{A, P\}.$$

□

Teoremă 5.0.4. *Fie $A = P - N$, unde A este simetrică şi pozitiv definită. Dacă $P + P^T - A$ este pozitiv definită, atunci P este inversabilă, iar iteraţia cu $B = I - P^{-1}A$ este convergentă monoton în raport cu norma $\|\cdot\|_A$ şi*

$$\rho(B) \leq \|B\|_A < 1.$$

Demonstraţie. Mai întâi, P este inversabilă. Într-adevăr, dacă ar exista $x \neq 0$ cu $Px = 0$, atunci

$$x^T(P + P^T - A)x = -x^T Ax < 0,$$

în contradicţie cu ipoteza $P + P^T - A \succ 0$.

Pentru $B = I - P^{-1}A$, un calcul direct dă identitatea

$$A - B^T A B = A P^{-T}(P + P^T - A)P^{-1}A.$$

Membrul drept este pozitiv definit: pentru $x \neq 0$, vectorul $P^{-1}Ax$ este nenul şi

$$x^T(A - B^T A B)x = (P^{-1}Ax)^T(P + P^T - A)(P^{-1}Ax) > 0.$$

Prin urmare,

$$\|Bx\|_A^2 = x^T B^T A B x < x^T A x = \|x\|_A^2 \quad \text{pentru orice } x \neq 0.$$

Pe sfera compactă $\{x : \|x\|_A = 1\}$, funcţia continuă $x \mapsto \|Bx\|_A$ îşi atinge maximul, iar inegalitatea strictă arată că acest maxim este mai mic decât 1. Astfel $\|B\|_A < 1$, deci $\rho(B) \leq \|B\|_A < 1$ şi iteraţia converge. Monotonia erorii rezultă imediat din $e^{(k+1)} = Be^{(k)}$. □

Extensie didactică: metode iterative ca probleme de punct fix

O metodă iterativă nu încearcă să producă soluţia după un număr predeterminat de operaţii. Ea construieşte un şir $x^{(0)}, x^{(1)}, \ldots$ şi urmăreşte apropierea de soluţia exactă. Avantajul apare mai ales pentru sisteme foarte mari şi rare, unde o factorizare directă ar necesita prea multă memorie sau ar produce factori mult mai denşi decât matricea iniţială.

Presupunem o descompunere $A = P - N$, unde P este uşor de inversat. Ecuaţia $Ax = b$ devine $Px = Nx + b$, deci

$$x = P^{-1}Nx + P^{-1}b.$$

Definind $B = P^{-1}N$ şi $c = P^{-1}b$, obţinem iteraţia staţionară $x^{(k+1)} = Bx^{(k)} + c$. Soluţia exactă este un punct fix al aplicaţiei $x \mapsto Bx + c$.

Ecuaţia erorii

Dacă x este soluţia exactă şi $e^{(k)} = x - x^{(k)}$, atunci $e^{(k+1)} = Be^{(k)}$, deci $e^{(k)} = B^k e^{(0)}$. Convergenţa pentru orice aproximaţie iniţială este echivalentă, în dimensiune finită, cu $B^k \to 0$, iar criteriul spectral fundamental este $\rho(B) < 1$. Astfel, analiza unui algoritm iterativ liniar se reduce la studiul matricei de iteraţie.

Este important că $\|B\| < 1$ într-o normă compatibilă este o condiţie suficientă comodă, dar nu necesară în orice normă dată. Criteriul exact este formulat prin raza spectrală.

Reziduu şi criteriu de oprire

În practică nu cunoaştem eroarea $e^{(k)}$, deoarece nu cunoaştem x. Putem calcula însă reziduul $r^{(k)} = b - Ax^{(k)}$. Un criteriu uzual este $\|r^{(k)}\| \leq \texttt{tol}(\|A\|\,\|x^{(k)}\| + \|b\|)$ sau o variantă relativă adaptată problemei. Legătura cu analiza de condiţionare din capitolul precedent este directă: dacă A este inversabilă şi $b \neq 0$, atunci, pentru o normă indusă compatibilă,

$$\frac{\|x - x^{(k)}\|}{\|x\|} \leq \kappa(A)\,\frac{\|r^{(k)}\|}{\|b\|}.$$

Prin urmare, un reziduu relativ mic garantează o eroare înainte mică numai atunci când problema nu este puternic rău condiţionată. Oprirea doar când $\|x^{(k+1)} - x^{(k)}\|$ este mică poate fi înşelătoare dacă metoda stagnează.

Laborator: schemă generică de iteraţie

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o funcţie care primeşte o funcţie pentru aplicarea lui B şi vectorul c, nu neapărat matricea B explicită. La fiecare iteraţie salvaţi norma reziduului. Reprezentaţi $\log_{10} \|r^{(k)}\|$ în funcţie de k. Pentru o convergenţă liniară, după un tranzitoriu punctele tind să se alinieze aproximativ pe o dreaptă.

Exerciţii şi verificarea înţelegerii

1. Derivaţi relaţia $e^{(k+1)} = Be^{(k)}$ pornind de la faptul că şi soluţia exactă verifică $x = Bx + c$.
2. Pentru $B = \mathrm{diag}(0.2, -0.8)$, descrieţi convergenţa componentelor erorii şi identificaţi componenta asimptotic dominantă.
3. Daţi un exemplu de matrice B cu $\rho(B) < 1$, dar $\|B\|_\infty > 1$.
4. Explicaţi de ce o metodă care converge în 10^4 iteraţii poate fi totuşi preferabilă unei metode directe pentru un sistem uriaş şi rar.
5. Arătaţi că alegerea formală $P = A$ conduce la convergenţă într-un pas, dar nu oferă un algoritm util dacă inversarea lui P este la fel de dificilă ca problema iniţială.

Mini-proiect: efectul rarităţii

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi matricea Laplacianului discret în 2D pe o grilă $N \times N$, astfel încât dimensiunea sistemului să fie $n = N^2$. Comparaţi memoria matricei rare cu memoria unei copii dense. Implementaţi o iteraţie simplă şi măsuraţi costul unui singur pas. Scopul este să observaţi că, pentru probleme rare, costul relevant este adesea proporţional cu numărul de elemente nenule, nu cu n^2.

Rezolvare orientativă

Eroare pe două moduri proprii

Pentru $B = \mathrm{diag}(0.2, -0.8)$ şi eroarea iniţială $e^{(0)} = (1, 1)^T$, avem $e^{(k)} = (0.2^k, (-0.8)^k)^T$. Prima componentă dispare foarte repede; a doua alternează în semn şi scade mult mai lent. Asimptotic, rata este controlată de $0.8 = \rho(B)$. Dacă am privi doar primele iteraţii ale primei componente, am putea concluziona greşit că metoda este mult mai rapidă.

Situaţie practică: criteriul de oprire

Pentru un sistem mare, calcularea reziduului poate necesita încă un produs matrice–vector, adică o operaţie comparabilă cu o iteraţie. Unele implementări actualizează reziduul recursiv şi îl recalculează exact doar periodic, pentru a evita acumularea erorilor. Alegerea criteriului de oprire este astfel şi o decizie de cost, nu numai una teoretică.

Exemplu rezolvat: eroarea se propagă prin aceeaşi matrice

Considerăm o iteraţie staţionară $x^{(k+1)} = Bx^{(k)} + c$ şi presupunem că x^* este punct fix, deci $x^* = Bx^* + c$. Scăzând cele două relaţii obţinem $e^{(k+1)} = Be^{(k)}$, unde $e^{(k)} = x^{(k)} - x^*$. Prin inducţie, $e^{(k)} = B^k e^{(0)}$. Această identitate este mai importantă decât formula iteraţiei însăşi: convergenţa pentru orice punct iniţial este echivalentă cu $B^k \to 0$.

Pentru $B = \mathrm{diag}(1/2, -1/3)$ avem $B^k = \mathrm{diag}(2^{-k}, (-3)^{-k})$. Prima componentă a erorii scade ca 2^{-k}, iar a doua ca 3^{-k} şi îşi schimbă semnul. Rata asimptotică este dominată de componenta cea mai lentă, asociată valorii proprii de modul maxim, aici $\rho(B) = 1/2$.

De ce criteriul de oprire nu trebuie confundat cu teorema de convergenţă

Teorema spune ce se întâmplă când $k \to \infty$. În program trebuie să decidem după un număr finit de paşi dacă aproximaţia este suficient de bună. Norma diferenţei $\|x^{(k+1)} - x^{(k)}\|$ poate deveni mică şi pentru o iteraţie care stagnează. Reziduul $\|b - Ax^{(k)}\|$ verifică direct sistemul, dar legătura sa cu eroarea depinde de condiţionare. Un criteriu bun trebuie interpretat, nu aplicat mecanic.

Sarcină de implementare: schelet generic de iteraţie

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o funcţie care primeşte o funcţie de actualizare $\Phi(x)$, un criteriu de reziduu şi parametrii `tol`, `maxit`. Păstraţi într-un istoric reziduul, norma pasului şi, când soluţia exactă este cunoscută, eroarea. Funcţia trebuie să întoarcă şi motivul opririi: toleranţă atinsă, număr maxim de iteraţii sau apariţia valorilor nenumerice. Acest schelet va putea fi reutilizat pentru Jacobi, Gauss–Seidel şi metode de optimizare.

Exerciţii suplimentare

1. Pentru $B = \begin{pmatrix} 0 & 1/2 \\ 1/4 & 0 \end{pmatrix}$ calculaţi valorile proprii şi decideţi dacă iteraţia converge.
2. Daţi un exemplu cu $\|B\|_\infty > 1$, dar $\rho(B) < 1$. Ce arată exemplul despre condiţiile suficiente bazate pe norme?
3. Arătaţi că dacă există o normă subordonată cu $\|B\| < 1$, atunci punctul fix este unic.
4. Dacă $x^{(0)} - x^*$ este vector propriu al lui B, descrieţi exact eroarea după k paşi.
5. Construiţi numeric o iteraţie cu $\rho(B)$ foarte aproape de 1 şi observaţi convergenţa lentă.

5.1 Jacobi, Gauss-Seidel şi metode relaxate

Jacobi, Gauss-Seidel şi metode relaxate

În această secţiune vom considera câteva metode iterative liniare clasice. Dacă intrările diagonale ale lui A sunt diferite de zero, putem extrage în fiecare ecuaţie necunoscuta corespunzătoare, obţinând sistemul liniar echivalent

$$x_i = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j \right], \qquad i = 1, \cdots, n. \tag{5.13}$$

În metoda Jacobi, odată ce a fost aleasă o valoare iniţială arbitrară $x^{(0)}$, $x^{(k+1)}$ se calculează prin formulele

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} \right], \qquad i = 1, \cdots, n. \tag{5.14}$$

Acest lucru înseamnă să efectuăm următoarea descompunere pentru A.

$$P = D, \qquad N = D - A = E + F, \tag{5.15}$$

unde D este matricea diagonală a intrărilor diagonale din A, E este matricea triunghiulară inferioară a intrărilor $e_{ij} = -a_{ij}$ dacă $i > j$, $e_{ij} = 0$ dacă $i \leq j$, iar F este matricea triunghiulară superioară a intrărilor $f_{ij} = -a_{ij}$ dacă $j > i$, $f_{ij} = 0$ dacă $j \leq i$. Ca o consecinţă,

$$A = D - (E + F), \tag{5.16}$$

Două convenţii echivalente pentru splitting

În capitol folosim $A = D - E - F$, unde $E = -A_{\text{inf}}$ şi $F = -A_{\text{sup}}$. În multe texte se scrie $A = D + L + U$, cu $L = A_{\text{inf}}$ şi $U = A_{\text{sup}}$. Cele două notaţii sunt identice prin $L = -E$, $U = -F$; de exemplu

$$B_J = D^{-1}(E + F) = -D^{-1}(L + U), \qquad B_{GS} = (D - E)^{-1}F = -(D + L)^{-1}U.$$

Vom păstra convenţia $D - E - F$ în corpul capitolului; atunci când un supliment foloseşte $D + L + U$, legătura de mai sus trebuie aplicată explicit.

iar matricea de iteraţie a metodei Jacobi este astfel dată de

$$B_J = D^{-1}(E+F) = I_n - D^{-1}A. \tag{5.17}$$

Metoda Jacobi relaxată

O generalizare a metodei Jacobi este metoda relaxată (sau JOR), în care, după introducerea unui parametru de relaxare ω, iteraţia Jacobi este înlocuită cu

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k)}\right] + (1-\omega)x_i^{(k)}, \qquad i = 1, \cdots, n. \tag{5.18}$$

Matricea de iteraţie corespunzătoare este

$$B_{J_\omega} = \omega B_J + (1-\omega)I_n. \tag{5.19}$$

Scrisă în termenii vectorului rezidual, metoda JOR corespunde la

$$x^{(k+1)} = x^{(k)} + \omega D^{-1}r^{(k)}. \tag{5.20}$$

Această metodă este consistentă cu sistemul $Ax = b$ pentru orice $\omega \neq 0$, iar pentru $\omega = 1$ coincide cu metoda Jacobi.

Metoda Gauss–Seidel

Metoda Gauss-Seidel se deosebeşte de metoda Jacobi prin faptul că la pasul $k+1$ se folosesc valorile deja disponibile ale lui $x^{(k+1)}$ pentru a actualiza soluţia, astfel încât, se are

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right], \qquad i = 1, \cdots, n. \tag{5.21}$$

Această metodă echivalează cu efectuarea următoarei descompuneri pentru A în

$$P = D - E, \qquad N = F, \tag{5.22}$$

iar matricea de iteraţie asociată este

$$B_{GS} = (D-E)^{-1}F. \tag{5.23}$$

Metoda Gauss–Seidel cu suprarelaxare

Pornind de la metoda Gauss-Seidel, prin analogie cu ceea ce s-a făcut pentru iteraţiile Jacobi, introducem metoda relaxării succesive (sau metoda SOR)

$$x_i^{(k+1)} = \frac{\omega}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right] + (1-\omega)x_i^{(k)}, \quad i = 1, \cdots, n. \tag{5.24}$$

Metoda SOR poate fi scrisă în formă vectorială sub forma

$$(I - \omega D^{-1}E)x^{(k+1)} = [(1-\omega)I_n + \omega D^{-1}F]x^{(k)} + \omega D^{-1}b, \tag{5.25}$$

din care matricea de iteraţie este

$$B(\omega) = (I - \omega D^{-1}E)^{-1}[(1-\omega)I + \omega D^{-1}F]. \tag{5.26}$$

Înmulţind cu D ambele părţi ale lui

$$(I - \omega D^{-1}E)x^{(k+1)} = [(1-\omega)I_n + \omega D^{-1}F]x^{(k)} + \omega D^{-1}b, \tag{5.27}$$

şi amintind că $A = D - (E + F)$, rezultă următoarea formă a metodei SOR

$$x^{(k+1)} = x^{(k)} + \left(\frac{1}{\omega}D - E\right)^{-1} r^{(k)}. \tag{5.28}$$

Este compatibilă pentru orice $\omega \neq 0$, iar pentru $\omega = 1$ coincide cu metoda Gauss-Seidel. În special, dacă $\omega \in (0, 1)$, metoda se numeşte sub-relaxare, în timp ce dacă $\omega > 1$ se numeşte supra-relaxare.

Completare didactică: Jacobi, Gauss–Seidel şi relaxarea

Scriem $A = D + L + U$, unde D este diagonala, L partea strict inferioară şi U partea strict superioară. Metoda Jacobi foloseşte numai valorile din iteraţia precedentă, ceea ce conduce la $Dx^{(k+1)} = b - (L + U)x^{(k)}$. Gauss–Seidel foloseşte imediat componentele deja actualizate, deci $(D + L)x^{(k+1)} = b - Ux^{(k)}$.

Această diferenţă aparent mică schimbă matricea de iteraţie şi, adesea, viteza de convergenţă. Jacobi este mai uşor de paralelizat deoarece toate componentele noi pot fi calculate independent din vectorul vechi. Gauss–Seidel este secvenţial în forma sa simplă, dar profită de informaţia nouă în interiorul aceleiaşi iteraţii.

Exemplu comparativ

Considerăm

$$A = \begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 3 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 4 \\ 7 \end{pmatrix},$$

cu soluţia exactă $x_\star = (1, 2, 3)^T$. Pornind de la $x^{(0)} = 0$, Jacobi produce $x_1^{(1)} = 1/2$, $x_2^{(1)} = 1$, $x_3^{(1)} = 7/3$. Gauss–Seidel foloseşte deja $x_1^{(1)}$ în a doua ecuaţie şi apoi $x_2^{(1)}$ în a treia; astfel informaţia se propagă mai repede prin vector.

Relaxarea

O iteraţie relaxată combină valoarea veche cu actualizarea propusă: $x^{(k+1)} = (1-\omega)x^{(k)} + \omega\widetilde{x}^{(k+1)}$. Pentru $0 < \omega < 1$ avem subrelaxare; pentru $\omega > 1$, suprarelaxare. Parametrul nu este un simplu factor cosmetic: el schimbă spectrul matricei de iteraţie şi poate accelera sau distruge convergenţa.

Laborator: trei metode, acelaşi sistem

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi Jacobi, Gauss–Seidel şi o versiune relaxată. Pentru matricea tridiagonală SPD de mai sus, reprezentaţi reziduul relativ în funcţie de numărul iteraţiei. Testaţi $\omega = 0.5, 1, 1.2, 1.5, 1.9$ pentru metoda relaxată relevantă şi observaţi sensibilitatea la parametru. Nu comparaţi doar numărul de iteraţii: comparaţi şi timpul efectiv.

Exerciţii şi verificarea înţelegerii

1. Scrieţi explicit matricele de iteraţie B_J şi B_{GS} pentru o matrice generică 2×2 cu diagonală nenulă.
2. Efectuaţi două iteraţii Jacobi şi două Gauss–Seidel pentru exemplul 3×3 de mai sus.
3. Explicaţi de ce metoda Jacobi necesită păstrarea vectorului vechi dacă actualizările sunt făcute componentă cu componentă.
4. Ce se întâmplă dacă un element diagonal al lui A este zero? Poate o permutare a ecuaţiilor remedia situaţia?
5. Discutaţi avantajele Jacobi într-o arhitectură paralelă şi avantajele Gauss–Seidel într-o implementare secvenţială.

Mini-proiect: rezolvarea unei probleme Poisson 2D

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Discretizaţi $-\Delta u = f$ pe un pătrat folosind schema cu cinci puncte. Implementaţi Jacobi şi Gauss–Seidel direct pe grilă, fără a forma neapărat matricea completă. Vizualizaţi suprafaţa aproximată la diferite iteraţii. Comparaţi numărul de iteraţii când grila este rafinată şi formulaţi o observaţie despre scalabilitatea metodelor staţionare simple.

Rezolvare orientativă

Două iteraţii, comparate numeric

Pentru sistemul din secţiune şi $x^{(0)} = 0$, Jacobi dă $x^{(1)} = (0.5, 1, 2.3333)^T$. A doua iteraţie este aproximativ $x^{(2)} = (0.75, 1.7083, 2.6667)^T$.
Gauss–Seidel foloseşte imediat valorile noi: $x_1^{(1)} = 0.5$, apoi $x_2^{(1)} = (4 + 0.5)/4 = 1.125$, iar $x_3^{(1)} = (7 + 1.125)/3 \approx 2.7083$. A doua iteraţie dă aproximativ $(0.7813, 1.8724, 2.9575)^T$. Soluţia exactă este $(1, 2, 3)^T$, iar avantajul propagării imediate a informaţiei se vede deja după doi paşi.

Situaţie practică: paralelism versus număr de iteraţii

Chiar dacă Gauss–Seidel are nevoie de mai puţine iteraţii, Jacobi poate fi mai rapid pe hardware masiv paralel deoarece fiecare componentă nouă depinde doar de vectorul vechi. Performanţa reală nu se deduce numai din raza spectrală; trebuie considerate arhitectura, costul sincronizării şi accesul la memorie.

Exemplu comparativ calculat pas cu pas

Fie

$$A = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix},$$

cu soluţia exactă $x^* = (0.1, 0.6)^T$. Pornim din $x^{(0)} = (0, 0)^T$. Pentru Jacobi, $x_1^{(k+1)} = (1 - x_2^{(k)})/4$ şi $x_2^{(k+1)} = (2 - 2x_1^{(k)})/3$. Rezultă $x_J^{(1)} = (0.25, 2/3)^T$ şi $x_J^{(2)} = (1/12, 1/2)^T$.

Pentru Gauss–Seidel folosim imediat componenta nouă $x_1^{(k+1)}$ în a doua ecuaţie. Astfel $x_{GS}^{(1)} = (0.25, 0.5)^T$, iar $x_{GS}^{(2)} = (0.125, 7/12)^T \approx (0.125, 0.5833)^T$. Ambele metode se apropie de soluţie, dar Gauss–Seidel foloseşte mai repede informaţia produsă în iteraţia curentă.

În acest exemplu matricea de iteraţie Jacobi are valori proprii $\pm\sqrt{1/6}$, deci $\rho(B_J) \approx 0.4082$. Pentru Gauss–Seidel, raza spectrală este $1/6 \approx 0.1667$. Diferenţa explică rata asimptotică mai bună observată aici. Nu trebuie însă generalizat că Gauss–Seidel este întotdeauna „de un anumit număr de ori” mai rapid; comportamentul depinde de matrice.

Sarcină practică: comparaţie pe aceeaşi problemă

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi Jacobi şi Gauss–Seidel cu acelaşi criteriu de oprire bazat pe reziduul relativ. Pentru matrici strict diagonal dominante de dimensiuni $n = 50, 100, 200$, porniţi din acelaşi vector şi raportaţi numărul de iteraţii, timpul şi evoluţia reziduului. Reprezentaţi $\log_{10}$ al reziduului în funcţie de iteraţie. Nu comparaţi metodele folosind toleranţe diferite.

Exerciţii suplimentare

1. Calculaţi a treia iteraţie Jacobi şi Gauss–Seidel pentru exemplul de mai sus.
2. Determinaţi explicit B_J şi B_{GS} şi verificaţi razele spectrale indicate.
3. Ce se schimbă dacă reordonăm ecuaţiile şi necunoscutele? Testaţi o permutare simultană.
4. Formulaţi metoda Jacobi ponderată şi identificaţi valorile $\omega = 1$ şi $\omega \neq 1$.
5. Daţi un exemplu de matrice pentru care Jacobi diverge. Urmăriţi numeric creşterea reziduului.

5.2 Convergenţa metodelor Jacobi şi Gauss-Seidel

Convergenţa metodelor Jacobi şi Gauss-Seidel

Există clase speciale de matrici pentru care este posibil să se stabilească a priori unele rezultate de convergenţă pentru metodele examinate în secţiunea anterioară. Primul rezultat în această direcţie este următorul.

Teoremă 5.2.1. *Dacă A este o matrice dominantă strict diagonală pe linii, metodele Jacobi şi Gauss-Seidel sunt convergente.*

Demonstraţie. Pentru Jacobi, $B_J = D^{-1}(E+F)$ şi, prin urmare,

$$\|B_J\|_\infty = \max_{1\le i\le n} \frac{\sum_{j\neq i} |a_{ij}|}{|a_{ii}|} < 1.$$

Rezultă $\rho(B_J) \le \|B_J\|_\infty < 1$, deci metoda Jacobi converge.

Pentru Gauss–Seidel, fie $B_{GS} = (D-E)^{-1}F$ şi presupunem $B_{GS}v = \lambda v$ pentru $v \neq 0$. Alegem i astfel încât $|v_i| = \|v\|_\infty$. Ecuaţia pe componenta i a relaţiei $Fv = \lambda(D-E)v$ este

$$\sum_{j>i}(-a_{ij})v_j = \lambda\left(a_{ii}v_i + \sum_{j<i} a_{ij}v_j\right).$$

Dacă $\lambda = 0$, nu avem nimic de demonstrat. Pentru $\lambda \neq 0$, împărţind prin $|v_i|$ şi folosind $|v_j| \le |v_i|$, obţinem

$$|\lambda|\left(|a_{ii}| - \sum_{j<i}|a_{ij}|\right) \le \sum_{j>i}|a_{ij}|.$$

Prin diagonal-dominanţa strictă, numitorul este pozitiv şi

$$|\lambda| \le \frac{\sum_{j>i}|a_{ij}|}{|a_{ii}| - \sum_{j<i}|a_{ij}|} < 1.$$

Aşadar toate valorile proprii ale lui B_{GS} au modul strict mai mic decât 1, deci $\rho(B_{GS}) < 1$ şi metoda Gauss–Seidel converge. □

Teoremă 5.2.2. *Dacă A şi $2D - A$ sunt matrici simetrice şi pozitiv definite, atunci metoda Jacobi este convergentă şi*

$$\rho(B_J) = \|B_J\|_A = \|B_J\|_D < 1.$$

Demonstraţie. Deoarece A este SPD, $a_{ii} = e_i^T A e_i > 0$ pentru orice i, deci matricea diagonală D este SPD. Pentru metoda Jacobi avem descompunerea $A = P - N$ cu $P = D$ şi matricea de iteraţie $B_J = I - D^{-1}A$. Ipoteza suplimentară este exact $2P - A = 2D - A \succ 0$. Toate ipotezele primului rezultat general sunt astfel îndeplinite, iar acel rezultat dă simultan convergenţa şi egalităţile $\rho(B_J) = \|B_J\|_A = \|B_J\|_D < 1$. □

Teoremă 5.2.3. *Dacă A este simetrică şi pozitiv definită, metoda Gauss–Seidel este convergentă monoton în raport cu norma $\|\cdot\|_A$.*

Demonstraţie. Pentru Gauss–Seidel luăm $P = D - E$, astfel încât $B_{GS} = I - P^{-1}A$. Simetria lui $A = D - E - F$ implică $E^T = F$ şi, prin urmare, $P^T = D - F$. Rezultă

$$P + P^T - A = (D-E) + (D-F) - (D-E-F) = D.$$

Cum A este SPD, toate elementele sale diagonale sunt pozitive, deci $D \succ 0$. Ipoteza celui de-al doilea rezultat general este satisfăcută; acesta arată că P este inversabilă, $\|B_{GS}\|_A < 1$ şi

$$\|e^{(k+1)}\|_A < \|e^{(k)}\|_A$$

pentru orice eroare nenulă. În particular, metoda converge pentru orice aproximaţie iniţială. □

Teoremă 5.2.4 (Relaţia spectrală Jacobi–Gauss–Seidel pentru cazul tridiagonal)**.** *Fie $A \in \mathbb{R}^{n\times n}$ tridiagonală, cu $a_{ii} \neq 0$. Pentru matricile de iteraţie Jacobi şi Gauss–Seidel definite mai sus avem*

$$\rho(B_{GS}) = \rho(B_J)^2.$$

Mai precis, fiecare valoare proprie nenulă μ a lui B_{GS} este pătratul unei valori proprii λ a lui B_J, iar reciproc, pentru fiecare $\lambda \in \sigma(B_J)$ avem $\lambda^2 \in \sigma(B_{GS})$.

Demonstraţie. Valorile proprii ale lui $B_J = D^{-1}(E+F)$ sunt zerourile lui

$$p_n(\lambda) = \det\Big(\lambda D - E - F\Big),$$

iar valorile proprii ale lui $B_{GS} = (D-E)^{-1}F$ sunt zerourile lui

$$q_n(\mu) = \det\Big(\mu(D-E) - F\Big),$$

deoarece $D-E$ este triunghiulară cu diagonală nenulă şi deci inversabilă.

Pentru o matrice tridiagonală, determinanţii principali ai celor două expresii satisfac recurenţe de ordin doi. Notând cu p_k şi q_k determinanţii submatricelor principale de ordin k, avem $p_0 = q_0 = 1$ şi

$$\begin{aligned} p_k(\lambda) &= \lambda a_{kk} p_{k-1}(\lambda) - a_{k,k-1}a_{k-1,k}p_{k-2}(\lambda), \\ q_k(\mu) &= \mu a_{kk} q_{k-1}(\mu) - \mu a_{k,k-1}a_{k-1,k}q_{k-2}(\mu). \end{aligned}$$

Arătăm prin inducţie că, pentru orice $\lambda \in \mathbb{C}$,

$$q_k(\lambda^2) = \lambda^k p_k(\lambda).$$

Identitatea este imediată pentru $k = 0, 1$. Dacă este adevărată pentru $k-1$ şi $k-2$, atunci

$$\begin{aligned} q_k(\lambda^2) &= \lambda^2 a_{kk}\lambda^{k-1}p_{k-1}(\lambda) - \lambda^2 a_{k,k-1}a_{k-1,k}\lambda^{k-2}p_{k-2}(\lambda) \\ &= \lambda^k\Big(\lambda a_{kk}p_{k-1}(\lambda) - a_{k,k-1}a_{k-1,k}p_{k-2}(\lambda)\Big) \\ &= \lambda^k p_k(\lambda). \end{aligned}$$

Dacă $p_n(\lambda) = 0$, rezultă $q_n(\lambda^2) = 0$, deci λ^2 este valoare proprie a lui B_{GS}. Reciproc, dacă $q_n(\mu) = 0$ şi $\mu \neq 0$, alegem o rădăcină pătrată complexă λ cu $\lambda^2 = \mu$; identitatea precedentă dă $p_n(\lambda) = 0$. Valorile proprii nenule ale lui B_{GS} sunt astfel pătratele valorilor proprii corespunzătoare ale lui B_J, iar valorile proprii nule nu modifică raza spectrală. Prin urmare $\rho(B_{GS}) = \rho(B_J)^2$. □

În consecinţă, în cazul tridiagonal cele două metode converg sau diverg simultan. Când converg, $\rho(B_{GS}) = \rho(B_J)^2 < \rho(B_J)$ dacă $0 < \rho(B_J) < 1$, iar Gauss–Seidel are o rată asimptotică mai bună. Pentru o matrice tridiagonală SPD, convergenţa Gauss–Seidel demonstrată mai sus implică astfel şi convergenţa Jacobi.

Convergenţa metodelor relaxate

Teoremă 5.2.5. *Dacă A este simetrică şi pozitiv definită, atunci metoda JOR este convergentă dacă $0 < \omega < 2/\rho(D^{-1}A)$.*

Demonstraţie. Avem $B_{J_\omega} = I - \omega D^{-1}A$. Deoarece A este SPD, toate elementele diagonale a_{ii} sunt pozitive, deci D este SPD. Matricea $D^{-1}A$ nu este în general simetrică, dar este similară cu matricea simetrică pozitiv definită

$$D^{-1/2}AD^{-1/2},$$

întrucât $D^{-1}A = D^{-1/2}(D^{-1/2}AD^{-1/2})D^{1/2}$. Prin urmare, valorile proprii μ_i ale lui $D^{-1}A$ sunt reale şi strict pozitive. Valorile proprii ale lui B_{J_ω} sunt $1 - \omega\mu_i$. Condiţia $\rho(B_{J_\omega}) < 1$ este echivalentă cu

$$-1 < 1 - \omega\mu_i < 1 \quad \text{pentru toate } i,$$

adică $0 < \omega\mu_i < 2$ pentru toate i. Este suficient şi necesar ca $0 < \omega < 2/\mu_{\max} = 2/\rho(D^{-1}A)$. □

Convergenţa metodelor relaxate

Teoremă 5.2.6. *Dacă metoda Jacobi este convergentă, atunci metoda JOR converge pentru orice* $0 < \omega \leq 1$.

Demonstraţie. Avem $B_{J_\omega} = \omega B_J + (1-\omega)I$. Dacă λ_k este o valoare proprie, eventual complexă, a lui B_J, atunci

$$\mu_k = \omega\lambda_k + 1 - \omega$$

este valoarea proprie corespunzătoare a lui B_{J_ω}. Convergenţa Jacobi înseamnă $|\lambda_k| < 1$ pentru toate k. Scriem $\lambda_k = r_k e^{i\theta_k}$, cu $0 \leq r_k < 1$. Pentru $0 < \omega \leq 1$,

$$\begin{aligned} |\mu_k|^2 &= \omega^2 r_k^2 + 2\omega(1-\omega) r_k \cos\theta_k + (1-\omega)^2 \\ &\leq (\omega r_k + 1 - \omega)^2 < 1, \end{aligned}$$

ultima inegalitate fiind strictă deoarece $r_k < 1$ şi $\omega > 0$. Astfel $\rho(B_{J_\omega}) < 1$, deci metoda JOR converge. □

Convergenţa metodelor relaxate

Teoremă 5.2.7. *Pentru orice* $\omega \in \mathbb{R}$ *avem* $\rho(B(\omega)) \geq |\omega - 1|$*; prin urmare, metoda SOR nu converge dacă* $\omega \leq 0$ *sau* $\omega \geq 2$.

Demonstraţie. Din

$$B(\omega) = (I - \omega D^{-1}E)^{-1}\Big((1-\omega)I + \omega D^{-1}F\Big)$$

observăm că primul factor are determinantul 1, fiind inversa unei matrice inferior triunghiulare cu diagonală unitară, iar al doilea factor este superior triunghiular cu toate elementele diagonale egale cu $1-\omega$. Aşadar

$$|\det B(\omega)| = |1-\omega|^n.$$

Dacă $\lambda_1, \ldots, \lambda_n$ sunt valorile proprii complexe ale lui $B(\omega)$, numărate cu multiplicitate algebrică, atunci

$$\prod_{i=1}^{n} |\lambda_i| = |1-\omega|^n.$$

Dacă toate ar satisface $|\lambda_i| < |1-\omega|$, produsul ar fi strict mai mic decât $|1-\omega|^n$, contradicţie. Prin urmare există cel puţin un i pentru care $|\lambda_i| \geq |1-\omega|$, deci $\rho(B(\omega)) \geq |1-\omega|$. Convergenţa ar necesita $\rho(B(\omega)) < 1$, deci este necesar $|1-\omega| < 1$, adică $0 < \omega < 2$. □

Completare didactică: cum demonstrăm convergenţa?

Pentru o metodă staţionară $x^{(k+1)} = Bx^{(k)} + c$, criteriul conceptual este $\rho(B) < 1$. În practică însă, calcularea exactă a spectrului lui B poate fi la fel de dificilă ca problema pe care încercăm să o rezolvăm. De aceea sunt importante condiţiile structurale suficiente, precum dominanţa diagonală strictă sau pozitiva definire în anumite metode.

Un exemplu în care spectrul se vede direct

Fie $A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$. Pentru Jacobi,

$$B_J = -D^{-1}(L+U) = \begin{pmatrix} 0 & -1/2 \\ -1/2 & 0 \end{pmatrix},$$

cu valori proprii $\pm 1/2$, deci $\rho(B_J) = 1/2$. Pentru Gauss–Seidel,

$$B_{GS} = -(D+L)^{-1}U = \begin{pmatrix} 0 & -1/2 \\ 0 & 1/4 \end{pmatrix},$$

şi $\rho(B_{GS}) = 1/4$. În acest exemplu, eroarea asimptotică a metodei Gauss–Seidel se contractă mai repede.

Dacă o matrice este strict diagonal dominantă pe linii, argumentele de tip normă infinit arată că metoda Jacobi converge. Pentru Gauss–Seidel există rezultate similare, iar pentru matrice SPD convergenţa sa este garantată. Aceste teoreme sunt utile tocmai fiindcă evită calculul explicit al valorilor proprii ale matricei de iteraţie.

Rata observată

Dacă o singură valoare proprie dominantă controlează regimul asimptotic, raportul dintre normele erorilor succesive tinde adesea către $\rho(B)$, în condiţii generice. Mai precis, urmărim raportul $\|e^{(k+1)}\|/\|e^{(k)}\|$. Cum eroarea exactă nu este disponibilă, în experimente se foloseşte de obicei reziduul. Panta graficului semilogaritmic oferă o estimare empirică a ratei de convergenţă.

Laborator: teorie versus experiment

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru mai multe matrice mici, calculaţi explicit B_J şi B_{GS} şi raza spectrală cu `max(abs(eig(B)))`. Rulaţi apoi metodele şi estimaţi rata din reziduuri. Includeţi: o matrice strict diagonal dominantă, una SPD dar nu puternic diagonal dominantă şi o matrice pentru care Jacobi diverge. Comparaţi predicţia spectrală cu experimentul.

Exerciţii şi verificarea înţelegerii

1. Verificaţi calculele pentru B_J şi B_{GS} din exemplul 2×2.
2. Demonstraţi că dacă $\|B\| < 1$ într-o normă matricială submultiplicativă, atunci metoda converge pentru orice $x^{(0)}$.
3. Explicaţi de ce $\rho(B) < 1$ este mai puţin restrictiv decât condiţia $\|B\|_\infty < 1$.
4. Construiţi o matrice 2×2 pentru care Jacobi diverge şi verificaţi că $\rho(B_J) > 1$.
5. Pentru o matrice diagonală, descrieţi Jacobi şi Gauss–Seidel. De ce ambele rezolvă sistemul într-o singură actualizare?

Mini-proiect: grila şi rata de convergenţă

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Pentru matricea Poisson 1D de ordin n, calculaţi numeric $\rho(B_J)$ pentru $n = 10, 20, 40, 80, 160$. Rulaţi Jacobi până la aceeaşi toleranţă şi comparaţi numărul de iteraţii. Formulaţi o concluzie despre deteriorarea convergenţei când grila este rafinată şi motivaţi necesitatea unor metode mai avansate sau a precondiţionării.

Rezolvare orientativă

Convergenţă verificată spectral

Pentru $A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$, B_J are valori proprii $\pm 1/2$, deci după k paşi o componentă generică a erorii este redusă aproximativ cu factorul $(1/2)^k$. Pentru B_{GS} valorile proprii sunt 0 şi $1/4$, deci regimul asimptotic este de ordin $(1/4)^k$. Dacă dorim o reducere cu 10^{-6}, estimarea grosieră cere aproximativ $k \approx 20$ paşi pentru factor $1/2$ şi $k \approx 10$ pentru factor $1/4$.

Situaţie practică: când o teoremă suficientă nu decide

O matrice poate să nu fie strict diagonal dominantă şi totuşi Jacobi să convergă. Eşecul unui criteriu suficient nu înseamnă divergenţă; înseamnă doar că teorema nu decide cazul. Atunci putem analiza matricea de iteraţie, transforma sistemul sau folosi o altă metodă cu garanţii structurale mai potrivite.

Demonstraţie comentată: de la raza spectrală la convergenţă

Pentru o metodă staţionară avem $e^{(k)} = B^k e^{(0)}$. Dacă B este diagonalizabilă, $B = V\Lambda V^{-1}$, atunci $B^k = V\Lambda^k V^{-1}$ şi fiecare valoare proprie este ridicată la puterea k. Dacă $|\lambda_i| < 1$ pentru toate valorile proprii, atunci $\Lambda^k \to 0$ şi, prin urmare, $B^k \to 0$. În cazul general, forma Jordan arată acelaşi lucru: termenii suplimentari sunt polinoame în k înmulţite cu λ^k, iar exponenţiala geometrică domină polinomul atunci când $|\lambda| < 1$.

Pentru reciprocă trebuie să ţinem cont că, deşi B este reală, o valoare proprie şi vectorul propriu asociat pot fi complecşi. Presupunem că $B^k \to 0$ pe $\mathbb{R}^n$ şi fie $Bz = \lambda z$, cu $z = x + iy \in \mathbb{C}^n \setminus \{0\}$. Atunci $B^k x \to 0$ şi $B^k y \to 0$, deci $B^k z \to 0$ în $\mathbb{C}^n$. Dar $B^k z = \lambda^k z$, de unde $|\lambda|^k \to 0$ şi, prin urmare, $|\lambda| < 1$. Cum aceasta este valabil pentru orice valoare proprie, rezultă $\rho(B) < 1$. Aşadar convergenţa pentru orice punct iniţial real este echivalentă cu $\rho(B) < 1$.

Rata asimptotică

Când componenta erorii asociată valorii proprii dominante este prezentă, raportul dintre erori succesive tinde în mod tipic către $\rho(B)$, într-un sens potrivit. De aceea o rază spectrală de 0.9 înseamnă o convergenţă mult mai lentă decât una de 0.2: după 20 de paşi,

$0.9^{20} \approx 0.12$, în timp ce 0.2^{20} este deja de ordin 10^{-14}.

Sarcină practică: estimarea experimentală a ratei

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru o matrice de iteraţie cunoscută, calculaţi $\rho(B)$ şi rulaţi metoda. După ce regimul tranzitoriu a trecut, estimaţi rata prin raportul $\|r_{k+1}\|/\|r_k\|$. Comparaţi această estimare cu $\rho(B)$. Repetaţi cu puncte iniţiale diferite; investigaţi cazul special în care eroarea iniţială are componentă foarte mică pe direcţia proprie dominantă.

Exerciţii suplimentare

1. Demonstraţi că $k^p q^k \to 0$ pentru orice $p \geq 0$ şi $0 < q < 1$; acesta este ingredientul folosit pentru blocurile Jordan.
2. Dacă B este simetrică, arătaţi că $\|B\|_2 = \rho(B)$. Ce simplificare aduce aceasta?
3. Pentru $B = \begin{pmatrix} 1/2 & 1 \\ 0 & 1/2 \end{pmatrix}$ calculaţi B^k şi verificaţi direct convergenţa în ciuda blocului Jordan.
4. De ce condiţia $\rho(B) < 1$ este exactă, în timp ce $\|B\| < 1$ într-o normă fixată este doar suficientă?
5. Construiţi o matrice cu $\rho(B) = 1$ pentru care unele puncte iniţiale converg, dar nu toate. Explicaţi de ce teorema cere convergenţă pentru orice punct iniţial.

Fişă de seminar şi laborator: metode iterative

Problema rezolvată 1: aceeaşi matrice, două descompuneri

Pentru $A = D - E - F$, metoda Jacobi foloseşte $P = D$, iar Gauss–Seidel foloseşte $P = D - E$. În forma generală $Px^{k+1} = (P - A)x^k + b$, ambele sunt cazuri ale aceleiaşi idei de splitting. Diferenţa este cât de mult din A plasăm în partea rezolvată exact la fiecare pas. Dacă P este mai apropiată de A, matricea de iteraţie poate avea spectru mai favorabil, dar rezolvarea cu P poate fi mai costisitoare. Această balanţă reapare în noţiunea modernă de precondiţionare.

Problema rezolvată 2: un criteriu suficient din dominanţă diagonală

Pentru Jacobi, $B_J = -D^{-1}(L + U)$ în convenţia uzuală. Dacă A este strict diagonal dominantă pe linii, atunci pe fiecare linie suma modulelor elementelor lui B_J este $\sum_{j \neq i} |a_{ij}|/|a_{ii}| < 1$. Prin urmare $\|B_J\|_\infty < 1$, deci $\rho(B_J) \leq \|B_J\|_\infty < 1$ şi metoda converge. Observaţi structura demonstraţiei: proprietate uşor verificabilă a coeficienţilor $\Rightarrow$ normă subunitară $\Rightarrow$ rază spectrală subunitară $\Rightarrow$ convergenţă.

Laborator integrator 5: faza de convergenţă

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi o familie $A(\theta)$ pentru care metoda Jacobi trece de la convergenţă la divergenţă pe măsură ce un parametru θ variază. Pentru fiecare θ, calculaţi $\rho(B_J)$ şi rulaţi 50 de iteraţii din acelaşi punct. Reprezentaţi într-o diagramă rata observată şi marcaţi pragul $\rho = 1$. Verificaţi şi cazuri în care norma $\|B\|_\infty$ nu este subunitară, dar metoda tot converge.

Set de probleme

1. Derivaţi matricile de iteraţie Jacobi şi Gauss–Seidel pentru un sistem 3×3 general.
2. Pentru matricea Poisson tridiagonală, investigaţi numeric cum creşte numărul de iteraţii Jacobi cu n.
3. De ce un criteriu bazat numai pe $\|x^{k+1} - x^k\|$ poate fi înşelător?
4. Arătaţi că dacă B are o valoare proprie λ cu $|\lambda| > 1$, există un punct iniţial pentru care iteraţia diverge.
5. Pentru B nediagonalizabil cu $\rho(B) < 1$, folosiţi blocul Jordan pentru a explica convergenţa.
6. Comparaţi Jacobi ponderat pentru mai multe valori ω pe aceeaşi matrice. Există un ω optim în experiment?
7. Ce avantaj de paralelizare are Jacobi faţă de Gauss–Seidel?
8. Ce avantaj de informaţie proaspătă are Gauss–Seidel faţă de Jacobi?

Proiect de capitol: explorator de convergenţă

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Realizaţi o aplicaţie care primeşte A, b şi construieşte automat matricile de iteraţie Jacobi şi Gauss–Seidel. Programul calculează razele spectrale, rulează metodele şi reprezintă reziduurile. Pentru cel puţin zece matrici din clase diferite, comparaţi predicţia spectrală cu comportamentul observat. Includeţi cel puţin două cazuri divergente şi unul foarte lent convergent.

Despre relaţia $\rho(B_{GS}) = \rho(B_J)^2$

Relaţia demonstrată pentru matricea tridiagonală model nu este valabilă pentru o matrice arbitrară. Ea aparţine unei clase speciale de matrici *consistently ordered*. În afara acestei clase, convergenţa Jacobi şi Gauss–Seidel trebuie analizată separat prin spectrul propriei matrice de iteraţie.

Pentru matricea Poisson 1D, unde $\rho(B_J) = \cos(\pi/(n+1))$, parametrul SOR optim este

$$\omega_{\text{opt}} = \frac{2}{1+\sqrt{1-\rho(B_J)^2}}, \qquad \rho(B_{\omega_{\text{opt}}}) = \omega_{\text{opt}} - 1.$$

Formula poate fi comparată direct cu rezultatele numerice obţinute pentru diferite valori ale lui ω.

Criteriul de oprire şi condiţionarea

Un reziduu mic nu garantează singur o eroare mică. Pentru $Ax = b$, $b \neq 0$, relaţia demonstrată în capitolul 4,

$$\frac{\|x - x^{(k)}\|}{\|x\|} \leq \kappa(A)\frac{\|r^{(k)}\|}{\|b\|}, \qquad r^{(k)} = b - Ax^{(k)},$$

arată de ce toleranţa pe reziduu trebuie interpretată împreună cu numărul de condiţionare.

Sarcini de aprofundare şi sinteză a metodelor iterative

Probleme de aprofundare: convergenţă, spectru şi criterii de oprire

1. Pentru sistemul $Ax = b$ cu $A = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix}$, construiţi matricile de iteraţie Jacobi şi Gauss–Seidel, calculaţi razele spectrale şi preziceţi care metodă converge mai repede asimptotic.
2. Daţi un exemplu 2×2 în care Jacobi converge, dar norma $\|B_J\|_\infty$ este mai mare sau egală cu 1. Ce arată exemplul despre condiţiile suficiente bazate pe norme?
3. Demonstraţi că dacă $\rho(B) < 1$, atunci eroarea $e^{(k)} = B^k e^{(0)}$ tinde la zero. Explicaţi de ce afirmaţia inversă este de asemenea adevărată pentru convergenţa din orice stare iniţială.
4. Pentru $B = V\,\mathrm{diag}(0.2, 0.7, 0.95)V^{-1}$, descrieţi componenta care domină eroarea după multe iteraţii. Cum se schimbă răspunsul dacă V este foarte prost condiţionată?
5. Arătaţi că dominanţa diagonală strictă pe linii este o condiţie suficientă pentru convergenţa Jacobi. Indicaţi ce normă este potrivită pentru demonstraţie.
6. Pentru o matrice SPD, explicaţi de ce Gauss–Seidel converge. Comparaţi această afirmaţie cu simpla dominanţă diagonală: care ipoteză este mai structurală?
7. Formulaţi un criteriu de oprire bazat pe reziduu şi unul bazat pe diferenţa dintre două iterate. Construiţi un exemplu conceptual în care al doilea criteriu poate fi înşelător.
8. Pentru o iteraţie cu factor asimptotic $q = 0.9$, estimaţi numărul de paşi necesar pentru a reduce eroarea cu un factor 10^{-6}. Repetaţi pentru $q = 0.99$ şi comentaţi diferenţa.
9. Explicaţi de ce o metodă cu multe iteraţii poate fi totuşi eficientă pentru un sistem rar de dimensiune foarte mare. Formulaţi costul în funcţie de numărul de elemente nenule.
10. Consideraţi o reordonare a necunoscutelor printr-o matrice de permutare. Discutaţi

de ce Jacobi este relativ insensibil la ordinea actualizărilor, în timp ce Gauss–Seidel poate fi afectat de ordonare.

11. Dacă este introdusă relaxarea $x^{(k+1)} = (1-\omega)x^{(k)} + \omega\Phi(x^{(k)})$, analizaţi pentru o problemă scalară cum alegerea lui ω poate accelera sau distruge convergenţa.
12. Pentru discretizarea 1D a lui $-u'' = f$, investigaţi teoretic sau numeric ce se întâmplă cu raza spectrală a lui Jacobi când numărul de puncte creşte. Ce consecinţă are asupra numărului de iteraţii?

Laborator integrator: Jacobi şi Gauss–Seidel pe problema Poisson

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Legaţi rata de convergenţă de dimensiunea grilei şi de spectrul matricei de iteraţie.

1. Discretizaţi $-u'' = f$ pe $(0,1)$ cu condiţii Dirichlet, alegând o soluţie exactă $u_\star$.
2. Pentru $N = 20, 40, 80, 160$, construiţi matricea rară şi implementaţi Jacobi şi Gauss–Seidel fără a folosi rutine iterative de bibliotecă.
3. Folosiţi acelaşi criteriu de oprire relativ pentru ambele metode şi raportaţi numărul de iteraţii, reziduul şi eroarea faţă de soluţia discretă exactă.
4. Pentru dimensiuni mici, calculaţi explicit $\rho(B_J)$ şi $\rho(B_{GS})$. Comparaţi cu rata observată din panta graficului semilogaritmic al reziduului.
5. Măsuraţi costul unui pas şi costul total. Explicaţi de ce metoda cu mai puţine iteraţii nu este automat mai rapidă dacă un pas este mai scump.
6. Testaţi două stări iniţiale foarte diferite şi verificaţi că rata asimptotică este aceeaşi, deşi faza tranzitorie se poate schimba.
7. Introduceţi o relaxare simplă şi experimentaţi cu mai multe valori ale lui ω. Identificaţi empiric o zonă favorabilă şi una nefavorabilă.
8. Comparaţi la final cu `A\b` doar ca referinţă. Discutaţi în ce regim o metodă iterativă devine atractivă.

De explicat la predare. De ce deteriorarea ratei odată cu rafinarea grilei motivează apariţia precondiţionării şi a metodelor iterative mai avansate?

Proiect de capitol: un sistem rar asociat unei reţele

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. În această sarcină, o *reţea* este formată din n noduri şi legături nedirecţionate între anumite perechi de noduri. Definim matricea de adiacenţă W prin $w_{ij} = 1$ dacă nodurile i şi j sunt legate şi $w_{ij} = 0$ altfel, iar $D = \operatorname{diag}(d_1, \dots, d_n)$, unde $d_i = \sum_j w_{ij}$

este numărul de vecini ai nodului i. Matricea $L = D - W$ se numeşte *Laplacianul reţelei*. Această definiţie este suficientă pentru proiect; nu presupunem cunoştinţe anterioare de teoria grafurilor. Consideraţi sistemul $(L + \alpha I)x = b$, cu $\alpha > 0$.

1. **Modelare.** Interpretaţi pur algebric componentele ecuaţiei i: arătaţi că $(Lx)_i = \sum_{j \sim i}(x_i - x_j)$, unde notaţia $j \sim i$ înseamnă că nodul j este legat de nodul i. Dacă doriţi o interpretare aplicată, puteţi privi x_i drept o mărime scalară ataşată nodului, iar b_i drept o sursă exterioară.
2. **Structură.** Verificaţi simetria, raritatea şi pozitiva definire a matricei.
3. **Algoritm.** Implementaţi Jacobi şi Gauss–Seidel folosind structura rară.
4. **Experiment.** Variaţi numărul de noduri şi numărul mediu de vecini ai unui nod, $\frac{1}{n}\sum_i d_i$; raportaţi numărul de elemente nenule şi timpul per iteraţie.
5. **Convergenţă.** Măsuraţi rata asimptotică şi comparaţi-o, pentru reţele mici, cu raza spectrală a matricei de iteraţie, noţiune deja folosită în analiza Jacobi şi Gauss–Seidel.
6. **Validare.** Comparaţi soluţia cu un solver direct doar pentru dimensiuni pentru care acesta este rezonabil.
7. **Interpretare.** Discutaţi relaţia dintre numărul şi distribuţia legăturilor din reţea şi dificultatea numerică observată.

Test de capitol

Test – Metode iterative pentru sisteme liniare

1. Calcul explicit. Pentru sistemul

$$\begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix} x = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \qquad x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

calculaţi primele două iteraţii Jacobi şi primele două iteraţii Gauss–Seidel. Scrieţi matricea de iteraţie pentru Jacobi şi verificaţi, folosind criteriul spectral din capitol, dacă metoda converge.

2. Analiza unui fragment MATLAB.

```
for i = 1:n
    xnew(i) = (b(i)-A(i,1:i-1)*xold(1:i-1) ...
                   -A(i,i+1:n)*xold(i+1:n))/A(i,i);
end
xold = xnew;
```

Care afirmaţii sunt corecte? Justificaţi.

A) fragmentul implementează o iteraţie Jacobi;

B) valorile nou calculate sunt folosite imediat în aceeaşi parcurgere;

C) metoda converge pentru orice matrice cu diagonală nenulă;

D) un criteriu de oprire trebuie adăugat pentru o implementare completă.

Capitolul 6

Sisteme dreptunghiulare şi aproximaţii

Extensie didactică: ce înseamnă un sistem dreptunghiular?

Până aici am lucrat în principal cu $A \in \mathbb{R}^{n\times n}$ inversabilă. În aplicaţii, această situaţie este doar una dintre posibilităţi. Pentru $A \in \mathbb{R}^{m\times n}$ apar trei regimuri geometrice: $m > n$ (mai multe ecuaţii decât necunoscute), $m = n$ şi $m < n$ (mai puţine ecuaţii decât necunoscute). Rangul matricei decide apoi dacă există soluţii şi câte.

Imagine geometrică

Coloanele lui A generează subspaţiul $\mathcal{R}(A) \subset \mathbb{R}^m$. Ecuaţia $Ax = b$ cere ca b să aparţină exact acestui subspaţiu. Dacă $b \notin \mathcal{R}(A)$, sistemul nu are soluţie exactă; problema celor mai mici pătrate înlocuieşte atunci b cu proiecţia sa ortogonală pe $\mathcal{R}(A)$. Dacă $m < n$ şi A are rang pe linii complet, sistemul poate avea infinit de multe soluţii, iar o regulă suplimentară – de exemplu norma minimă – selectează una dintre ele.

Această formulare leagă trei instrumente care vor apărea succesiv: ecuaţiile normale, factorizarea QR şi descompunerea SVD. Ele rezolvă aceeaşi geometrie, dar cu avantaje numerice diferite.

Exemplu geometric

Fie $A = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \in \mathbb{R}^{3\times 1}$ şi $b = (1, 2, 4)^T$. Nu există în general un scalar x pentru care $(x, x, x)^T = b$. Cea mai bună aproximaţie în norma euclidiană este proiecţia lui b pe dreapta generată de $(1, 1, 1)^T$, adică $x = (1+2+4)/3 = 7/3$. Reziduul este ortogonal pe coloană: $(1, 1, 1)^T(Ax-b) = 0$.

Laborator: clasificarea automată a unei probleme

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o rutină care, pentru o matrice A, raportează m, n, o estimare numerică a rangului şi clasifică problema ca pătratică, supradeterminată sau subdeterminată. Generaţi apoi exemple cu rang complet şi rang deficient. Pentru fiecare, folosiţi `rank` doar ca instru-

ment de diagnostic, nu ca metodă de rezolvare, şi comparaţi soluţiile produse de `A\b` şi `pinv(A)*b`.

Exerciţii şi verificarea înţelegerii

1. Pentru $A \in \mathbb{R}^{m\times n}$ cu rang r, precizaţi dimensiunile nucleului şi ale imaginii folosind teorema rang–nulitate.
2. Daţi un exemplu 3×2 cu soluţie exactă pentru un anumit b şi un alt b pentru care sistemul este incompatibil.
3. Daţi un exemplu 2×3 cu infinit de multe soluţii şi parametrizaţi mulţimea lor.
4. Explicaţi de ce o soluţie de normă minimă trebuie să fie ortogonală pe nucleul lui A.

Mini-proiect: localizare din măsurători redundante

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Consideraţi o mărime scalară x măsurată de mai mulţi senzori, fiecare furnizând $b_i = x + \varepsilon_i$. Formulaţi problema ca sistem supradeterminat şi derivaţi estimatorul celor mai mici pătrate. Apoi introduceţi ponderi diferite pentru senzori cu precizii diferite şi comparaţi estimatorii pe date simulate.

6.1 “Soluţia” sistemelor supradeterminate

Sisteme algebrice liniare

Să considerăm următorul sistem algebric de ecuaţii liniare:

$$x_1 + x_2 = 1,$$
$$x_1 - x_2 = 2.$$

Semnificaţia geometrică a acestui sistem este că se caută un punct de intersecţie a două drepte, vezi figura.

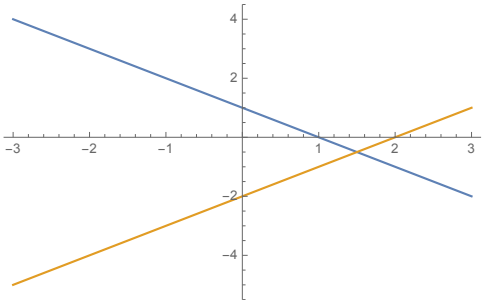


Figura 6.1: Interpretarea geometrică a sistemului format din dreptele $x_1 + x_2 = 1$ şi $x_1 - x_2 = 2$.

Eliminarea gaussiană

În mod clar, efectuând o eliminare gaussiană, înmulţim prima ecuaţie cu (-1) şi o adăugăm la a doua pentru a obţine următorul sistem echivalent

$$\begin{aligned} x_1 + x_2 &= 1, \\ -2\,x_2 &= 1. \end{aligned}$$

A doua ecuaţie ne dă $x_2 = -\frac{1}{2}$ şi, împreună cu prima ecuaţie, găsim şi $x_1 = \frac{3}{2}$.

Prin urmare, punctul de intersecţie al celor două drepte este punctul $(x_1, x_2) = (\frac{3}{2}, -\frac{1}{2})$.

Sisteme supradeterminate

Acum, să luăm în considerare următorul sistem algebric de ecuaţii liniare:

$$\begin{aligned} x_1 + x_2 &= 1, \\ x_1 - x_2 &= 2, \\ 3\,x_1 + x_2 &= 4. \end{aligned}$$

Punctul de intersecţie al acestor trei drepte este acelaşi cu cel din exemplul anterior, deoarece ultima ecuaţie este redundantă.

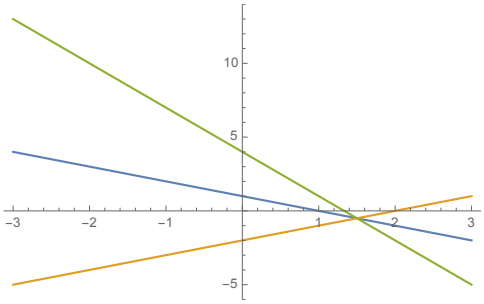


Figura 6.2: Sistem supradeterminat compatibil: a treia ecuaţie, $3x_1 + x_2 = 4$, este redundantă.

Sisteme supradeterminate

Dar ce se întâmplă dacă a treia ecuaţie nu este redundantă? Spunem că sistemul este supradeterminat. De exemplu

$$\begin{aligned} x_1 + x_2 &= 1, \\ x_1 - x_2 &= 2, \\ 3\,x_1 + x_2 &= 3. \end{aligned}$$

Nu există un punct de intersecţie a acestor trei drepte.

"Soluţia" sistemelor supradeterminate

Deci, sistemul algebric

$$\begin{aligned} x_1 + x_2 &= 1, \\ x_1 - x_2 &= 2, \\ 3\,x_1 + x_2 &= 3. \end{aligned}$$

nu are o soluţie.

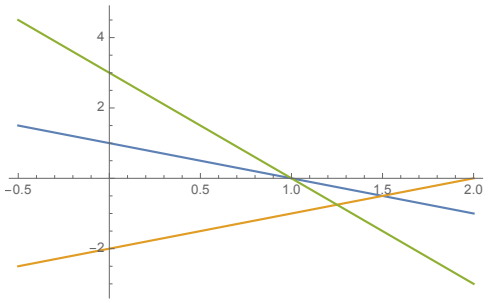


Figura 6.3: Sistem supradeterminat incompatibil: cele trei drepte nu au un punct comun.

Cu toate acestea, suntem în continuare interesaţi să găsim un punct (x_1, x_2) care este **“o soluţie aproximativă”**.

“Soluţia” sistemelor supradeterminate

Deci, sistemul algebric

$$\begin{aligned} x_1 + x_2 &= 1, \\ x_1 - x_2 &= 2, \\ 3\,x_1 + x_2 &= 3 \end{aligned}$$

nu are o soluţie.

Cu toate acestea, suntem în continuare interesaţi să găsim un punct (x_1, x_2) care este **“‘o soluţie aproximativă”**.

De ce?

“Soluţia” sistemelor supradeterminate

De ce?

Pentru a răspunde la această întrebare, trebuie să remarcăm că sistemul algebric

$$\begin{aligned} x_1 + x_2 &= 1, \\ x_1 - x_2 &= 2, \\ 3\,x_1 + x_2 &= 4 \end{aligned}$$

are o soluţie unică, în timp ce perturbată sistemul algebric perturbat

$$\begin{aligned} x_1 + x_2 &= 1, \\ x_1 - x_2 &= 2, \\ 3\,x_1 + x_2 &= 4.0001 \end{aligned}$$

nu are o soluţie.

De ce?

În practică, coeficienţii unui asemenea sistem provin adesea din măsurători şi, prin urmare, nu îi cunoaştem exact. O mică eroare în măsurători ar putea conduce la un sistem algebric incompatibil şi ne interesează să vedem care punct (x_1, x_2) satisface “mai bine” sistemul.

“Soluţia” sistemelor supradeterminate

Mai întâi de toate, să remarcăm că sistemul algebric

$$\begin{aligned} x_1 + x_2 &= 1, \\ x_1 - x_2 &= 2, \\ 3\,x_1 + x_2 &= 3. \end{aligned}$$

poate fi scris sub formă de matrice sub forma

$$A\,x = b,$$

unde $A = \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 3 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 2}$, $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$ şi $b = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \in \mathbb{R}^3$.

"Soluţia" sistemelor supradeterminate

Printr-o "soluţie" a sistemului supradeterminat

$$A\,x = b$$

înţelegem un vector $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$ astfel încât $A\,x$ să nu fie "atât de departe" de b. Cu alte cuvinte, un vector $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$ astfel încât $A\,x - b$ să nu fie "departe" de $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

Dar ce înseamnă că un vector nu este "atât de departe" de un alt vector?

Completare didactică: de la incompatibilitate la aproximaţie optimă

Într-un sistem supradeterminat $Ax = b$, cu $m > n$, fiecare ecuaţie adaugă o constrângere. Dacă datele provin din măsurători, zgomotul face foarte probabil ca toate constrângerile să nu poată fi satisfăcute simultan. În loc să eliminăm ecuaţii, folosim redundanţa pentru a găsi un vector care explică datele cât mai bine.

Alegerea normei euclidiene conduce la problema $\min_x \|Ax - b\|_2^2$. Condiţia de optimalitate este $A^T(Ax - b) = 0$, adică reziduul $r = b - Ax$ este ortogonal pe toate coloanele lui A. Aceasta este condiţia geometrică centrală, iar ecuaţiile normale $A^TAx = A^Tb$ sunt doar expresia sa algebrică.

Exemplu: media ca problemă LS

Dacă vrem să aproximăm valorile $1, 2, 4$ printr-o singură constantă x, minimizăm $(x-1)^2 + (x-2)^2 + (x-4)^2$. Derivând obţinem $6x - 14 = 0$, deci $x = 7/3$. Exact aceeaşi formulă rezultă din ecuaţiile normale pentru $A = (1, 1, 1)^T$. Media aritmetică este astfel cel mai simplu estimator de tip cele mai mici pătrate.

Unicitatea

Dacă A are rang coloană complet, atunci A^TA este SPD şi minimizatorul este unic. Dacă rangul este deficient, funcţia are direcţii plate: mai multe valori ale lui x produc acelaşi vector Ax şi acelaşi reziduu. În acest caz este nevoie fie de o selecţie suplimentară (de exemplu norma minimă), fie de regularizare.

Laborator: verificarea ortogonalităţii

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Generaţi o matrice $A \in \mathbb{R}^{20\times 4}$ cu rang complet şi un vector b aleator. Calculaţi o soluţie LS, apoi verificaţi numeric `norm(A'*(A*x-b))`. Comparaţi cu `norm(A*x-b)`. Prima mărime ar trebui să fie aproape de eroarea de rotunjire, în timp ce reziduul însuşi nu trebuie să fie mic în valoare absolută dacă b are o componentă importantă în afara imaginii lui A.

Exerciţii şi verificarea înţelegerii

1. Derivaţi ecuaţiile normale prin diferenţierea lui $f(x) = \tfrac{1}{2}\|Ax-b\|_2^2$.
2. Derivaţi aceleaşi ecuaţii folosind condiţia geometrică de ortogonalitate a reziduului.
3. Demonstraţi că A^TA este pozitiv definită dacă şi numai dacă A are coloane liniar independente.
4. Dacă $b \in \mathcal{R}(A)$ şi A are rang coloană complet, ce relaţie există între soluţia LS şi soluţia exactă?
5. Explicaţi de ce adăugarea unei ecuaţii redundante poate îmbunătăţi robusteţea statistică fără a creşte numărul necunoscutelor.

Mini-proiect: calibrarea unui senzor

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Simulaţi o relaţie $y = ax + b$ şi măsuraţi y pentru 30 de valori ale lui x, adăugând zgomot gaussian. Formulaţi sistemul supradeterminat pentru parametrii (a, b). Estimaţi parametrii, reprezentaţi dreapta şi reziduurile, apoi repetaţi experimentul de 100 de ori şi studiaţi distribuţia erorilor parametrilor.

Exemplu rezolvat: trei măsurători pentru doi parametri

Căutăm o dreaptă $y \approx a + bt$ prin punctele $(0,1)$, $(1,2)$ şi $(2,2)$. Sistemul exact ar fi

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix},$$

dar cele trei ecuaţii nu sunt compatibile: primele două ar impune $a = 1, b = 1$, ceea ce ar prezice valoarea 3 la $t = 2$. În sensul celor mai mici pătrate, ecuaţiile normale sunt

$$\begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix},$$

şi obţinem $a = 7/6$, $b = 1/2$. Valorile ajustate sunt $(7/6, 5/3, 13/6)^T$, iar reziduul $b - Ax = (-1/6, 1/3, -1/6)^T$ este ortogonal pe ambele coloane ale lui A: suma componentelor sale este zero şi produsul cu $(0, 1, 2)^T$ este de asemenea zero.

Această ortogonalitate este mai fundamentală decât formula ecuaţiilor normale. Vectorul Ax este proiecţia ortogonală a lui b pe spaţiul coloanelor lui A, iar reziduul este componenta rămasă în complementul ortogonal.

Sarcină practică: date redundante şi zgomot

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Generaţi 50 de observaţii din $y = 2 - 0.7t$ şi adăugaţi zgomot gaussian. Estimaţi parametrii folosind 2, 5, 10, 25 şi 50 de observaţii, repetând experimentul de 200 de ori. Reprezentaţi dispersia estimărilor lui a şi b. Discutaţi cum redundanţa poate reduce variabilitatea estimării fără a face sistemul exact compatibil.

Exerciţii suplimentare

1. Calculaţi norma reziduului în exemplul de mai sus şi verificaţi $A^T r = 0$.
2. Arătaţi că dacă $b \in \mathcal{R}(A)$, problema LS recuperează o soluţie exactă.
3. Dacă A are rang coloană complet, demonstraţi unicitatea proiecţiei şi a vectorului parametrilor.
4. Daţi un exemplu cu rang deficient în care vectorul proiectat Ax este unic, dar parametrul x nu este unic.
5. Formulaţi o problemă reală în care numărul măsurătorilor poate fi mărit fără a creşte numărul parametrilor.

6.2 Problema celor mai mici pătrate şi ajustarea datelor

Am văzut că soluţia sistemului liniar $Ax = b$ există şi este unică dacă $n = m$ şi A este nesingulară.

În această secţiune dăm un sens soluţiei unui sistem liniar în cazul supradeterminat, $m > n$.

“Soluţia” sistemelor supradeterminate

Aproximarea soluţiei sistemelor supradeterminate

Să presupunem că ni se dă un sistem liniar de forma

$$A\,x = b, \quad \text{unde} \quad A \in \mathbb{R}^{m\times n} \quad \text{şi} \quad b \in \mathbb{R}^m \quad \text{cu} \quad m > n.$$

Presupunem, de asemenea, că $\text{rank}(A) = n$.

În aceste condiţii, sistemul de ecuaţii liniare considerat poate fi incompatibil (nu are soluţie).

Observăm că un sistem supradeterminat nu are, în general, soluţie decât dacă partea dreaptă b este un element al lui $\text{Range}(A) := \{y \in \mathbb{R}^m | \exists\, x \in \mathbb{R}^n \text{ a. î. } Ax = y\}$.

Aproximarea soluţiei sistemelor supradeterminate

O abordare obişnuită pentru găsirea unei soluţii aproximative constă în găsirea acelui x pentru care se realizează valoarea minimă a normei reziduului $r = b - A\,x$, pe $\mathbb{R}^m$, adică

$$\text{(LS)} \qquad \min_{x\in\mathbb{R}^n} \|A\,x - b\|^2.$$

Aceasta este o problemă de minimizare a unei funcţii pătratice pe întreg spaţiu, funcţia obiectiv pătratică fiind dată de

$$f(x) = \left\langle A^T A\,x, x\right\rangle - 2\left\langle b, A\,x\right\rangle + \|b\|^2.$$

Problema celor mai mici pătrate

Dat fiind $A \in \mathbb{R}^{m\times n}$ cu $m \geq n$ şi $b \in \mathbb{R}^m$, spunem că $x^* \in \mathbb{R}^n$ este o soluţie în sensul celor mai mici pătrate dacă

$$\|Ax^* - b\|_2 \leq \|Ax - b\|_2 \qquad \text{pentru orice } x \in \mathbb{R}^n.$$

Echivalent, pentru $f(x) = \|Ax - b\|_2^2$, avem $f(x^*) = \min_{x\in\mathbb{R}^n} f(x)$.

Astfel, problema constă în minimizarea normei euclidiene a reziduului.

Existenţa minimizatorului

În dimensiune finită, Range(A) este un subspaţiu închis. Proiecţia ortogonală a lui b pe Range(A) există şi este unică; dacă $Ax^* = P_{\text{Range}(A)}b$, atunci x^* este o soluţie LS. Dacă A are rang coloană complet, această soluţie este unică.

Ecuaţiile care caracterizează soluţia pot fi deduse geometric. Dacă x^* este o soluţie de cele mai mici pătrate, atunci Ax^* este proiecţia ortogonală a lui b pe spaţiul coloanelor lui A. Reziduul $r^* = b - Ax^*$ este, prin urmare, ortogonal pe fiecare coloană a lui A, adică

$$A^T(b - Ax^*) = 0.$$

Echivalent,

$$A^T Ax^* = A^T b, \tag{6.1}$$

sistem cunoscut sub numele de *ecuaţiile normale*.

Sistemul este nesingular dacă A are rang complet şi, în acest caz, soluţia celor mai mici pătrate există şi este unică.

Observăm că $B = A^T A$ este o matrice simetrică şi pozitiv definită. Astfel, pentru a rezolva ecuaţiile normale, se poate calcula mai întâi factorizarea Cholesky $B = H^T H$ şi apoi se pot rezolva cele două sisteme $H^T y = A^T b$ şi $Hx^* = y$. Cu toate acestea, din cauza erorilor de rotunjire, calculul lui $A^T A$ poate fi afectat de o pierdere de cifre semnificative, cu o pierdere consecventă a pozitivei definiri sau a nesingularităţii matricei, aşa cum se întâmplă în următorul exemplu (implementat în MATLAB) în care, pentru o matrice A cu rang complet, matricea corespunzătoare $fl(ATA)$ se dovedeşte a fi singulară

$$A = \begin{pmatrix} 1 & 1 \\ 2^{-27} & 0 \\ 0 & 2^{-27} \end{pmatrix}, \qquad fl(A^T A) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}. \tag{6.2}$$

Prin urmare, pentru matrici prost condiţionate, formarea explicită a lui $A^T A$ poate amplifica dificultăţile numerice. O metodă alternativă, care evită formarea explicită a lui $A^T A$, va fi introdusă ulterior; aici reţinem doar motivul pentru care ecuaţiile normale trebuie utilizate cu atenţie.

Completare didactică: problema celor mai mici pătrate

Problema celor mai mici pătrate apare frecvent în aplicaţiile algebrei liniare. Ea apare atunci când un sistem $Ax = b$ nu are, în general, soluţie exactă şi dorim să alegem vectorul x pentru care reziduul $r = b - Ax$ are norma euclidiană minimă. Toate noţiunile folosite în această secţiune au fost introduse până în acest punct: proiecţie ortogonală, norme, condiţionare şi factorizarea Cholesky.

Dacă $A \in \mathbb{R}^{m\times n}$ are coloane liniar independente, atunci soluţia x^* este caracterizată de ortogonalitatea reziduului faţă de spaţiul coloanelor lui A:

$$A^T(b - Ax^*) = 0,$$

adică prin ecuaţiile normale $A^TAx^* = A^Tb$. Matricea A^TA este simetrică şi pozitiv definită, deoarece pentru $z \neq 0$ avem $z^TA^TAz = \|Az\|_2^2 > 0$. Prin urmare, ecuaţiile normale au soluţie unică şi pot fi rezolvate cu factorizarea Cholesky.

De ce ecuaţiile normale trebuie folosite cu atenţie

Din rezultatele capitolului despre condiţionare, pentru o matrice cu rang coloană complet avem $\kappa_2(A^TA) = \kappa_2(A)^2$. Aşadar, formarea lui A^TA poate accentua sensibilitatea numerică a problemei. În acest capitol folosim ecuaţiile normale deoarece instrumentele necesare sunt deja disponibile; o alternativă care evită formarea explicită a lui A^TA va fi introdusă în capitolul despre factorizarea QR.

Exemplu: ajustarea unei drepte

Pentru punctele $(0, 1)$, $(1, 2)$, $(2, 2)$ căutăm $y \approx a + bt$. Matricea şi vectorul datelor sunt

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}, \qquad y = \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}.$$

Ecuaţiile normale sunt

$$\begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 5 \\ 6 \end{pmatrix},$$

de unde $a = 7/6$ şi $b = 1/2$. Reziduul este ortogonal pe ambele coloane ale lui A, ceea ce oferă şi o verificare directă a rezultatului.

Problema ponderată

Dacă observaţiile au importanţe diferite, introducem ponderi $w_i > 0$ şi minimizăm

$$\sum_{i=1}^{m} w_i(a_i^Tx - b_i)^2 = \|W^{1/2}(Ax - b)\|_2^2, \qquad W = \operatorname{diag}(w_1, \ldots, w_m).$$

Aceasta este problema celor mai mici pătrate pentru matricea $\widetilde{A} = W^{1/2}A$ şi vectorul $\widetilde{b} = W^{1/2}b$. Ecuaţiile normale devin $A^TWAx = A^TWb$.

Laborator: ecuaţiile normale şi condiţionarea

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi matrici dreptunghiulare cu coloane liniar independente şi rezolvaţi problema celor mai mici pătrate prin ecuaţiile normale, folosind Cholesky pentru A^TA. Construiţi apoi familii de matrici cu două coloane tot mai apropiate una de alta. Pentru fiecare caz calculaţi $\kappa_2(A)$, $\kappa_2(A^TA)$, reziduul şi, atunci când aţi construit datele pornind de la o soluţie cunoscută, eroarea soluţiei. Verificaţi numeric relaţia dintre cele două numere de condiţionare.

Exerciţii şi verificarea înţelegerii

1. Rezolvaţi complet problema de ajustare a dreptei din exemplu şi verificaţi $A^Tr = 0$.
2. Arătaţi că suma reziduurilor este zero atunci când modelul include o coloană constantă şi ponderile sunt egale.
3. Demonstraţi că, dacă A are coloane liniar independente, A^TA este pozitiv definită.
4. Formulaţi problema ponderată ca o problemă standard pentru $\widetilde{A} = W^{1/2}A$ şi $\widetilde{b} = W^{1/2}b$.
5. Ce se întâmplă cu unicitatea dacă două coloane ale lui A sunt identice? Justificaţi prin ecuaţiile normale.

Mini-proiect: ajustare polinomială

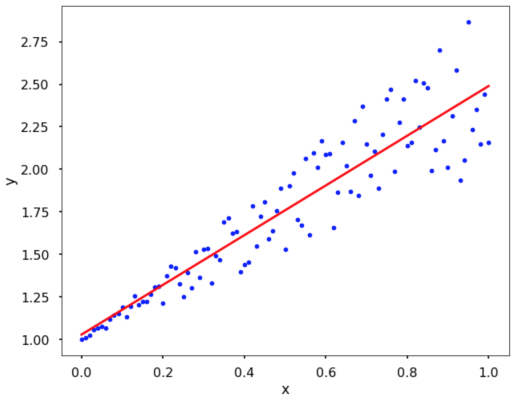
Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Generaţi date dintr-o funcţie netedă pe $[-1, 1]$ şi adăugaţi zgomot. Ajustaţi polinoame de grade $1, 3, 5$ folosind ecuaţiile normale. Pentru fiecare grad raportaţi reziduul, condiţionarea matricei modelului şi, pe un set separat de puncte unde funcţia iniţială este cunoscută, eroarea de predicţie. Explicaţi de ce un reziduu mai mic pe datele folosite la ajustare nu garantează automat o predicţie mai bună.

6.3 Ajustarea datelor (data fitting)

Un domeniu în care se utilizează problema cel mai mici pătrate este corelarea datelor.

Ajustarea datelor (data fitting)



Ajustare liniară a datelor în plan

Date n puncte în $\mathbb{R}^2$, obiectivul este de a găsi o dreaptă de forma

$$y = a\,x + b$$

care se potriveşte cel mai bine cu acestea. Aceasta înseamnă că trebuie să găsim a şi b care să definească această dependenţă liniară.

Relaţiile liniare care trebuie corelate sunt

$$y_i = a\,x_i + b, \qquad i = 1, 2, ..., n,$$

adică, sistemul care trebuie "rezolvat" este

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

Deci, găsiţi a şi b “soluţie" pentru

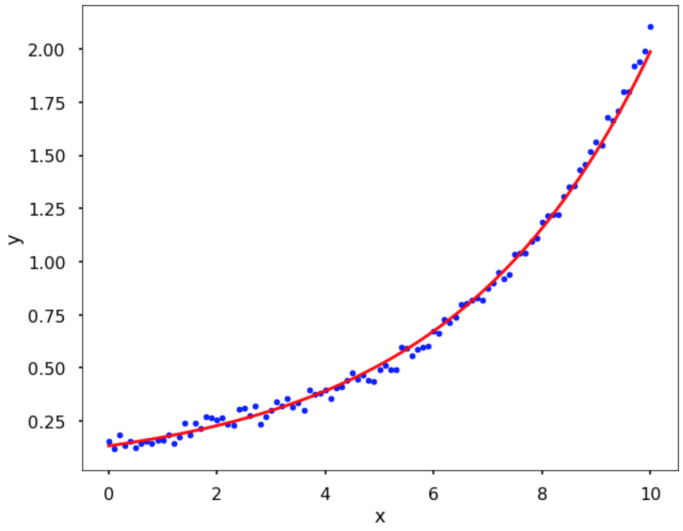
$$\underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{pmatrix}}_{:=X} \begin{pmatrix} a \\ b \end{pmatrix} = \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{:=y}.$$

Soluţia problemei celor mai mici pătrate este

$$\begin{pmatrix} a \\ b \end{pmatrix} = (X^T X)^{-1} X^T y.$$

Ajustarea datelor (data fitting)

Unul dintre domeniile în care se utilizează problema celor mai mici pătrate este corelarea datelor.



Ajustare liniară a datelor

Să presupunem că ni se dă un set de date (s_i, t_i), $i = 1, 2, ..., m$, unde $s_i \in \mathbb{R}^n$ şi $t_i \in \mathbb{R}$, şi să presupunem că o relaţie liniară de forma

$$t_i = \langle s_i, x \rangle, \qquad i = 1, 2, ..., m,$$

este căutată. Găsiţi x pentru a putea aproxima această dependenţă liniară!

Deci, problema este de a găsi vectorul de parametri $x \in \mathbb{R}^n$ care rezolvă problema

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{m} (\langle s_i, x \rangle - t_i)^2.$$

Aceasta este o problemă (LS) scrisă ca

$$\min_{x \in \mathbb{R}^n} \|S\,x - t\|^2,$$

unde $S = \begin{pmatrix} - & - & s_1^T & - & - \\ - & - & s_2^T & - & - \\ & & \vdots & & \\ - & - & s_m^T & - & - \end{pmatrix}$, $t = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{pmatrix}$.

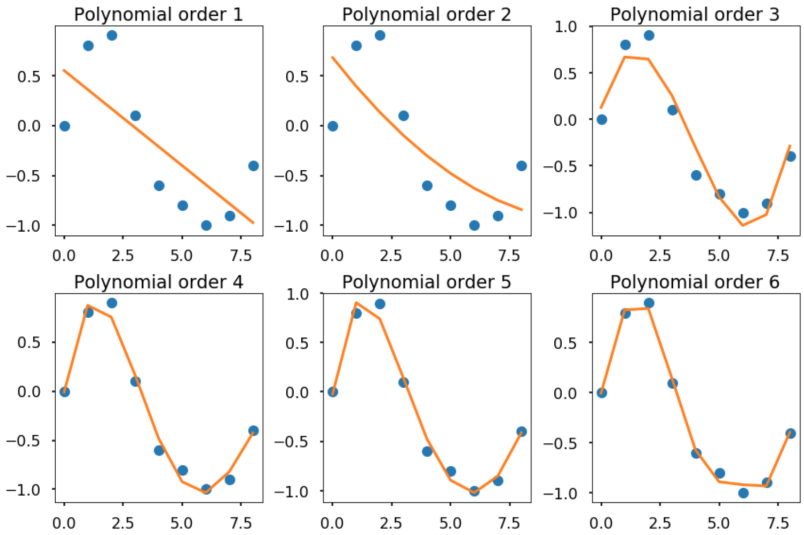
Alte situaţii

Alte situaţii

Mai multe despre corelarea datelor

Abordarea celor mai mici pătrate poate fi utilizată şi în cazul ajustărilor neliniare. Să presupunem, de exemplu, că ni se dă un set de puncte în $\mathbb{R}^2$: (u_i, y_i), $i = 1, 2, ..., m$, şi că ştim a priori că aceste puncte sunt aproximativ legate prin intermediul unui polinom de grad cel mult d; adică există $a_0, a_1, ..., a_d$ astfel încât

$$\sum_{j=0}^{d} a_j u_i^j \approx y_i, \qquad i = 1, ..., m.$$



Problema celor mai mici pătrate asociată este următoarea: se caută $a_0, a_1, ..., a_d$ astfel încât vectorul coeficienţilor să fie soluţia celor mai mici pătrate a sistemului liniar

$$\begin{pmatrix} 1 & u_1 & u_1^2 & \cdots & u_1^d \\ 1 & u_2 & u_2^2 & \cdots & u_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & u_m & u_m^2 & \cdots & u_m^d \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_d \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}.$$

(LS) este, desigur, bine definită dacă $m \geq d+1$. Matricea este aşa-numita matrice Vandermonde, despre care se ştie că este de rang $d+1$ dacă $d+1$ din u_i-uri sunt diferite între ele.

Completare didactică: de la un nor de puncte la un model

Într-o problemă de ajustare a datelor (data fitting) trebuie separate două decizii: alegerea *familiei de modele* şi estimarea *parametrilor*. Algebra liniară rezolvă foarte bine a doua problemă atunci când modelul depinde liniar de parametri, chiar dacă dependenţa de variabila independentă este neliniară. De exemplu, $p(t) = a_0 + a_1 t + \cdots + a_d t^d$ este neliniar în t, dar liniar în vectorul parametrilor $(a_0, \ldots, a_d)$.

Construirea matricei de design

Pentru date (t_i, y_i), fiecare observaţie furnizează o linie $(1, t_i, t_i^2, \ldots, t_i^d)$ a matricei Vandermonde. Astfel, alegerea bazei de funcţii este transformată într-o alegere a coloanelor lui A. Putem înlocui monomii cu alte funcţii $\phi_j(t)$ şi obţine $A_{ij} = \phi_j(t_i)$ fără a schimba formalismul LS.

Monomii de grad mare pot produce matrici prost condiţionate. O strategie simplă este rescalarea variabilei t într-un interval precum $[-1, 1]$; una mai avansată este folosirea unei baze polinomiale mai bine condiţionate.

Diagnostic prin reziduuri

Un reziduu mic nu dovedeşte că modelul este corect. Reprezentarea reziduurilor în funcţie de t poate revela structură neexplicată: curbură, heteroscedasticitate, puncte aberante sau depen-

denţe periodice. Algebra liniară oferă estimatorul pentru modelul ales; validitatea modelului rămâne o problemă de interpretare.

Laborator: ajustare şi validare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Folosiţi un set de date real sau simulat. Împărţiţi datele în 70% pentru ajustare şi 30% pentru validare. Ajustaţi mai multe familii liniare în parametri şi calculaţi eroarea pe ambele seturi. Reprezentaţi şi reziduurile. Nu selectaţi modelul doar după eroarea de antrenare.

Exerciţii şi verificarea înţelegerii

1. Scrieţi matricea de design pentru un polinom de grad 3 prin cinci puncte $t_1, \dots, t_5$.
2. Formulaţi ca problemă LS modelul $y \approx a \sin t + b \cos t + c$.
3. Formulaţi modelul $y \approx a e^{\alpha t}$ atunci când α este cunoscut şi când este necunoscut. În care caz problema este liniară în parametri?
4. Explicaţi de ce duplicarea identică a tuturor observaţiilor nu schimbă minimizatorul LS neponderat.
5. Daţi un exemplu de reziduuri mici pe datele observate, dar predicţii slabe în afara intervalului de date.

Mini-proiect: consum sau temperatură

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Alegeţi o serie temporală simplă: temperatură zilnică, consum energetic, trafic sau un set de date similar. Construiţi un model liniar în parametri care include trend şi una sau două componente periodice. Estimaţi parametrii prin LS, separaţi intervalul de instruire de cel de test şi discutaţi ce informaţie poate şi ce informaţie nu poate captura modelul.

Exemplu rezolvat: ajustare pătratică cu reziduuri nenule

Considerăm datele $(-1, 2)$, $(0, 1)$, $(1, 2)$ şi $(2, 6)$ şi modelul $p(t) = a + bt + ct^2$. Matricea de design şi vectorul observaţiilor sunt

$$A = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix}, \qquad y = \begin{pmatrix} 2 \\ 1 \\ 2 \\ 6 \end{pmatrix}.$$

Rezolvarea ecuaţiilor normale conduce la

$$a = \frac{17}{20}, \qquad b = \frac{1}{20}, \qquad c = \frac{5}{4}.$$

Vectorul $A(a,b,c)^T - y$ este $(1,-3,3,-1)^T/20$, iar suma pătratelor reziduurilor este $1/20$. Semnele alternante arată că polinomul trece când deasupra, când sub date; observaţia relevantă este că reziduul este ortogonal pe fiecare dintre coloanele $1, t, t^2$ ale matricei de design.

Modelul nu este acelaşi lucru cu algoritmul

Alegerea gradului polinomului este o decizie de modelare, distinctă de alegerea algoritmului numeric. În acest capitol folosim ecuaţiile normale şi instrumentele deja introduse; metode alternative vor fi studiate ulterior. Un algoritm numeric bun nu poate compensa un model prost ales, iar un model adecvat poate fi compromis de o matrice de design prost scalată.

Sarcină practică: selecţia gradului

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Generaţi 80 de puncte dintr-o funcţie netedă şi adăugaţi zgomot. Împărţiţi datele în set de antrenare şi set de test. Pentru gradele 1–15, ajustaţi un polinom, calculaţi eroarea pe ambele seturi şi $\kappa_2(A)$. Reprezentaţi cele trei mărimi în funcţie de grad. Repetaţi după rescalarea lui t în $[-1,1]$. Discutaţi separat supraînvăţarea şi condiţionarea bazei monomiale.

Exerciţii suplimentare

1. Verificaţi coeficienţii exemplului prin calculul explicit al lui A^TA şi A^Ty.
2. Construiţi matricea de design pentru modelul $a + b\sin t + c\cos t$.
3. Explicaţi de ce modelul ae^{bt} este neliniar în parametrii (a,b), dar devine parţial liniar dacă b este cunoscut.
4. Comparaţi folosirea monomilor $1, t, t^2, \dots$ cu o bază de polinoame ortogonale pe un interval scalat.
5. Ce informaţie vă poate oferi graficul reziduurilor pe care norma globală $\|r\|_2$ nu o arată?

Fişă de seminar: sisteme dreptunghiulare şi modelarea datelor

Problema rezolvată 1: model trigonometric liniar în parametri

Datele periodice pot fi modelate prin $y(t) \approx a + b\sin t + c\cos t$. Deşi modelul este neliniar ca funcţie de t, este liniar în parametrii necunoscuţi. Pentru observaţii (t_i, y_i), linia i a matricei de design este $(1, \sin t_i, \cos t_i)$. Putem deci formula direct problema celor mai mici pătrate, care

este liniară în parametrii necunoscuţi. Trebuie făcută această distincţie: „neliniar în variabila independentă” nu înseamnă automat „problemă neliniară de optimizare”.

Dacă adăugăm o frecvenţă necunoscută, $y \approx a + b\sin(\omega t) + c\cos(\omega t)$, problema devine neliniară în ω. O strategie posibilă este să fixăm temporar ω, să rezolvăm liniar în (a, b, c) şi să explorăm separat frecvenţa. Acesta este un exemplu de structură separabilă.

Problema rezolvată 2: rang deficient şi parametri nedeterminabili

Dacă matricea de design conţine două coloane identice, de exemplu modelul este scris redundant ca $a + bt + ct$, atunci numai suma $b + c$ este identificabilă din date. Există infinit de multe perechi (b, c) cu aceeaşi predicţie. Problema nu este lipsa observaţiilor, ci redundanţa parametrizării. Matricea de design nu are rang coloană complet, $A^T A$ este singulară şi ecuaţiile normale nu determină unic vectorul parametrilor.

Laborator integrator 6: modelare şi validare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Alegeţi un set de date temporal de minimum 100 de puncte. Construiţi trei modele liniare în parametri: trend liniar, trend pătratic şi trend plus componentă trigonometrică. Separaţi 70% din date pentru ajustare şi 30% pentru test. Pentru fiecare model folosiţi ecuaţiile normale, rezolvate cu o metodă deja studiată pentru sistemul rezultat, şi raportaţi reziduul de antrenare, eroarea de test, condiţionarea matricei de design şi graficele reziduurilor. Selectaţi un model şi justificaţi alegerea.

Set de probleme

1. Formulaţi problema celor mai mici pătrate asociată ajustării $y \approx a + b/t$ pentru $t_i \neq 0$.
2. Modelul $y \approx a\exp(bt)$ este liniar în parametri? Ce se întâmplă dacă aplicăm logaritmul şi datele au zgomot aditiv?
3. Arătaţi că duplicarea tuturor observaţiilor multiplică funcţia LS cu o constantă şi nu schimbă minimizatorul.
4. Pentru observaţii cu varianţe diferite, argumentaţi alegerea ponderilor inverse varianţei.
5. Construiţi un model cu cinci parametri şi numai trei observaţii. Ce tip de sistem obţineţi?
6. Explicaţi diferenţa dintre interpolare şi ajustare prin cele mai mici pătrate.
7. Ce informaţie despre model poate sugera un grafic al reziduurilor cu structură sinusoidală?
8. De ce extrapolarea poate fi slabă chiar dacă ajustarea pe intervalul de date este excelentă?

Proiect de capitol: raport de ajustare a datelor

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Alegeţi un set de date public sau simulat cu interpretare clară. Formulaţi cel puţin trei modele candidate, construiţi matricele de design şi rezolvaţi prin ecuaţiile normale. Includeţi analiza rangului şi condiţionării, separarea train/test, reziduuri şi o discuţie despre ce parametri au interpretare robustă. Raportul trebuie să explice de ce un model a fost preferat, nu doar să prezinte o eroare minimă.

Sarcini de aprofundare şi sinteză pentru cele mai mici pătrate

Probleme de aprofundare: proiecţie, ecuaţii normale şi ajustarea datelor

1. Pentru $A \in \mathbb{R}^{m\times n}$ cu rang coloană complet, demonstraţi că un minimizator al lui $\|Ax-b\|_2^2$ satisface $A^T(Ax-b)=0$. Interpretaţi geometric relaţia $A^T(b-Ax)=0$.
2. Dacă coloanele lui A sunt ortonormate, deduceţi direct formula soluţiei LS şi formula proiecţiei lui b pe spaţiul coloanelor lui A.
3. Demonstraţi că, dacă A are rang coloană complet, matricea A^TA este simetrică pozitiv definită şi ecuaţiile normale au soluţie unică.
4. Folosind rezultatele despre condiţionare din capitolul precedent, explicaţi de ce formarea lui A^TA poate amplifica dificultăţile numerice atunci când coloanele lui A sunt aproape dependente.
5. Pentru o dreaptă $y=a+bt$, deduceţi ecuaţiile normale manual şi precizaţi condiţia asupra punctelor t_i pentru unicitatea parametrilor.
6. Pentru un model pătratic $y=a+bt+ct^2$, construiţi explicit matricea de design şi ecuaţiile normale pentru patru observaţii date.
7. Arătaţi că, într-un model liniar cu termen liber, suma reziduurilor este zero la soluţia LS. Interpretaţi această proprietate.
8. Construiţi un set de date cu un outlier sever şi calculaţi ajustarea unei drepte înainte şi după introducerea lui. Explicaţi de ce suma pătratelor este sensibilă la valori aberante.
9. Comparaţi două modele de grade diferite pe aceleaşi date. De ce un model cu reziduu de antrenare mai mic nu este automat mai bun?
10. Propuneţi o împărţire simplă a datelor în set de ajustare şi set de verificare şi descrieţi cum aţi folosi eroarea pe al doilea set pentru alegerea gradului.

Laborator integrator: ajustare polinomială prin ecuaţii normale

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Studiaţi simultan alegerea modelului, scalarea datelor şi efectul condiţionării.

1. Generaţi date zgomotoase dintr-o funcţie netedă pe un interval şi păstraţi separat o parte dintre puncte pentru verificare.
2. Construiţi matricile de design pentru gradele $1, 2, 5, 10$.
3. Calculaţi numărul de condiţionare al lui A şi al lui $A^T A$ şi comparaţi valorile.
4. Rezolvaţi ecuaţiile normale folosind Cholesky atunci când matricea este pozitiv definită.
5. Repetaţi după transformarea variabilei independente astfel încât valorile ei să aparţină intervalului $[-1, 1]$ şi comparaţi condiţionarea şi coeficienţii.
6. Pentru fiecare grad raportaţi reziduul pe datele folosite la ajustare şi eroarea pe datele păstrate pentru verificare.
7. Adăugaţi câţiva outlieri şi repetaţi experimentul. Discutaţi separat efectul asupra parametrilor şi asupra predicţiilor.
8. Reprezentaţi grafic datele, curbele ajustate şi reziduurile pentru două grade reprezentative.

Proiect de capitol: calibrarea unui model liniar din date

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Alegeţi un set de date cu una sau două variabile explicative şi un răspuns măsurat.

1. **Modelare.** Formulaţi un model liniar sau polinomial şi construiţi matricea de design.
2. **Calcul manual.** Pentru un subset mic de observaţii scrieţi explicit ecuaţiile normale.
3. **Existenţă şi unicitate.** Verificaţi rangul coloanelor şi pozitiva definire a lui $A^T A$.
4. **Algoritm.** Rezolvaţi ecuaţiile normale printr-o metodă directă deja studiată.
5. **Scalare.** Rescalaţi variabilele şi comparaţi numerele de condiţionare.
6. **Validare.** Păstraţi o parte dintre observaţii în afara ajustării şi comparaţi predicţiile modelului pe acestea.
7. **Robusteţe empirică.** Perturbaţi uşor datele şi observaţi variaţia coeficienţilor.
8. **Interpretare.** Separaţi concluziile despre adecvarea modelului de concluziile despre dificultatea numerică a calculului.

Test de capitol

Test – Sisteme dreptunghiulare şi aproximaţii

1. Ajustare prin cele mai mici pătrate. Se măsoară valorile

$$(t_i, y_i) \in \{(0,1), (1,2), (2,2)\}.$$

Se caută dreapta $y \approx a + bt$. Construiţi matricea A şi vectorul y, deduceţi ecuaţiile normale şi determinaţi coeficienţii a, b. Calculaţi valorile ajustate şi reziduul.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi. Pentru problema

$$\min_x \|Ax - b\|_2,$$

A) o soluţie minimizează distanţa dintre Ax şi b;

B) ecuaţiile normale sunt $A^T Ax = A^T b$;

C) dacă coloanele lui A sunt liniar independente, soluţia celor mai mici pătrate este unică;

D) orice problemă supradeterminată are o soluţie exactă a lui $Ax = b$.

Capitolul 7

Regularizarea problemei celor mai mici pătrate şi eliminarea zgomotului

Regularizarea Problemei celor mai mici pătrate

Atunci când A este subdeterminată, adică atunci când există mai puţine ecuaţii decât variabile, există mai multe soluţii optime pentru problema celor mai mici pătrate şi nu este clar care dintre aceste soluţii optime este cea care trebuie luată în considerare.

În modelul de optimizare ar trebui încorporat un anumit tip de informaţii prealabile despre x.

O modalitate de a face acest lucru este de a lua în considerare o problemă penalizată în care o funcţie de regularizare $R(\cdot)$ este adăugată la funcţia obiectiv.

Cele mai mici pătrate regularizate

RLS

Problema celor mai mici pătrate cu regularizare (RLS) are forma

$$\text{(RLS)} \qquad \min_{x\in\mathbb{R}^n} \|A\,x - b\|^2 + \lambda\, R(x),$$

unde $\lambda > 0$ este parametrul de regularizare. Pe măsură ce λ devine mai mare, funcţia de regularizare primeşte o pondere mai mare.

În multe cazuri, se consideră că regularizarea este pătratică. În special $R(x) = \|D\,x\|^2$, cu $D \in \mathbb{R}^{p\times n}$ dat. Funcţia de regularizare pătratică urmăreşte să controleze norma lui $D\,x$ şi este formulată după cum urmează:

$$\text{(RLS)} \qquad \min_{x\in\mathbb{R}^n} \|A\,x - b\|^2 + \lambda\, \|D\,x\|^2,$$

sau, după dezvoltarea pătratelor, prin minimizarea funcţiei

$$f_{\text{RLS}}(x) = \left\langle (A^TA + \lambda D^TD)x, x \right\rangle - 2\langle A^Tb, x\rangle + \|b\|_2^2.$$

Problema poate fi scrisă ca problema celor mai mici pătrate pentru sistemul augmentat

$$\begin{pmatrix} A \\ \sqrt{\lambda}\, D \end{pmatrix} x \approx \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Aplicând ecuaţiile normale deja deduse, obţinem

$$(A^TA + \lambda D^TD)x = A^Tb.$$

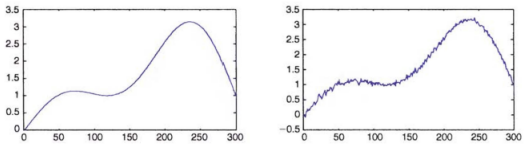


Figure 3.2. *A signal (left image) and its noisy version (right image).*

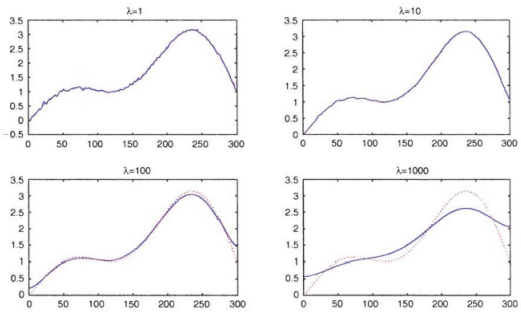


Figure 3.3. *Four reconstructions of a noisy signal by RLS solutions.*

Dacă matricea $A^T A + \lambda D^T D$ este pozitiv definită, sistemul are soluţie unică, iar aceasta este minimizatorul problemei regularizate.

Prin urmare, dacă D şi λ sunt astfel încât $A^T A + \lambda D^T D \succ 0$, atunci soluţia RLS este dată de

$$x_{RLS} = (A^T A + \lambda D^T D)^{-1} A^T b.$$

Alte situaţii

Eliminarea zgomotului dintr-un semnal

Problema eliminării zgomotului

Să presupunem că observăm o versiune bruiată a unui semnal necunoscut $x \in \mathbb{R}^n$:

$$b = x + w.$$

Aici x este un semnal necunoscut, w este un vector de zgomot necunoscut, iar b este vectorul măsurătorilor cunoscute.
Problema eliminării zgomotului este următoarea: Având în vedere b, găsiţi o estimare "bună" a lui x.

Dacă am folosi numai termenul de fidelitate faţă de date, problema celor mai mici pătrate ar da soluţia $x = b$.

Pentru a găsi o problemă mai relevantă, vom adăuga un termen de regularizare. Pentru aceasta, trebuie să exploatăm unele informaţii a priori despre semnal.

De exemplu, am putea şti că semnalul este neted într-un anumit sens. În acest caz, putem adăuga o penalizare pătratică, care este suma pătratelor diferenţelor dintre componentele consecutive ale vectorului; adică funcţia de regularizare este

$$R(x) = \sum_{i=1}^{n-1} (x_i - x_{i+1})^2.$$

Această funcţie pătratică poate fi scrisă şi sub forma $R(x) = \|L\,x\|^2$, unde $L \in \mathbb{R}^{(n-1)\times n}$ este

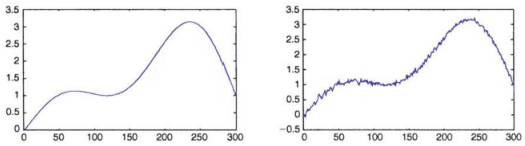


Figure 3.2. *A signal (left image) and its noisy version (right image).*

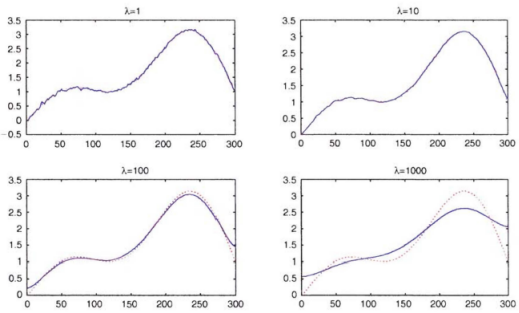


Figure 3.3. *Four reconstructions of a noisy signal by RLS solutions.*

dată de

$$L = \begin{pmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 \end{pmatrix}$$

Problema celor mai mici pătrate cu regularizare care rezultă este

$$\min_{x \in \mathbb{R}^n} \|x - b\|^2 + \lambda \|L\, x\|^2,$$

iar soluţia sa optimă este dată de

$$x_{\text{RLS}}(\lambda) = (I_n + \lambda L^T L)^{-1} b,$$

unde $\lambda > 0$ este un parametru de regularizare dat (bun).

Am putea găsi un astfel de $\lambda > 0$?

Alte situaţii

Aplicaţie de pe telefonul dumneavoastră

Transmiteţi prin WhatsApp sau prin alt canal de comunicaţii un mesaj vocal făcând un zgomot de fundal, de exemplu, sunetul cheilor sau al unui clopoţel. Salvaţi fişierul cu numele `sunet_bruiat.ogg` şi folosiţi codul de mai jos.

Pentru rezolvarea sistemului $(I+\lambda L^T L)x = b$ folosiţi metodele studiate până aici care exploatează structura matricei. Matricea este rară, simetrică şi pozitiv definită, deci o metodă adaptată acestei structuri este preferabilă unei formule bazate pe determinanţi.

Comparaţi timpul necesar reconstruirii semnalului. Acesta oferă o indicaţie asupra costului metodei de rezolvare a sistemului.

Observaţi că pentru unele metode apare la final un bruiaj în plus, ceva nou! Aceasta indică eroarea cu care se rezolvă sistemul prin metoda aleasă. Practic, "auziţi eroarea"! Eroarea se reflectă în semnalul final.

Folosiţi atât metode directe cât şi metode iterative!

```
[b,fs] = audioread('sunet_bruiat.ogg');
%incarcam sunetul cu zgomot
%Verificam sunetul si afisam semnalul descris de acesta
sound(b, fs);
plot(b);
hold on
% Afisam lungimea pentru a intelege magnitudinea datelor
n = length(b)
% Construim matricele In si L folosind matrici rare (sparse)
%deoarece au dimensiuni
% foarte mari, dar majoritatea elementelor sunt nule
idn = speye(n);
e = ones(n-1,1);
L = spdiags([e -e], [0 1], n-1, n);
% Definim lambda (putem incepe de la o valoare mica
%si sa tot crestem)
lambda = 5;

% Folosim metoda celor mai mici patrate cu regularizare
%patratica (vezi curs) si ajungem la format
% x = inv(In + lambda*L'*L) *b, dar nu folosim aceasta forma,
%deoarece matricea este prea mare.
% Fie folosim operatorul \ care implementeaza deja cea mai
%eficienta metoda de rezolvare a unui
% sistem liniar, luand in calcul si cazul matricelor rare:
x = (speye(n) + lambda*(L'*L)) \ b;
% Ascultam semnalul filtrat:
sound(x, fs);
plot(x);
hold on
```

Atenţie la împărţirea artificială pe blocuri

Matricea $I + \lambda L^T L$ este tridiagonală, iar blocurile vecine sunt cuplate prin elementele de pe diagonalele secundare. Rezolvarea independentă a blocurilor diagonale ar elimina aceste cuplaje şi ar rezolva o altă problemă. Se va păstra matricea întreagă şi se va folosi o metodă pentru sisteme rare/tridiagonale sau operatorul MATLAB \ numai ca referinţă de verificare.

Extensie didactică: regularizarea ca informaţie a priori

O problemă poate fi foarte sensibilă la perturbaţiile datelor. Regularizarea introduce explicit informaţie suplimentară despre soluţia considerată plauzibilă. În forma Tikhonov,

$$\min_x \Big(\|Ax - b\|_2^2 + \lambda \|Dx\|_2^2 \Big), \qquad \lambda > 0,$$

primul termen măsoară fidelitatea faţă de date, iar al doilea penalizează anumite caracteristici ale soluţiei prin matricea dată D.

Dacă $D = I$, sunt penalizate soluţiile cu normă mare. Dacă D este matricea diferenţelor de

ordinul întâi,

$$D = \begin{pmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix},$$

atunci Dx conţine diferenţele $x_{i+1} - x_i$, iar penalizarea descurajează variaţiile rapide între componente vecine.

Problema regularizată se poate scrie ca problema celor mai mici pătrate pentru sistemul augmentat

$$\begin{pmatrix} A \\ \sqrt{\lambda} D \end{pmatrix} x \approx \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

Prin ecuaţiile normale obţinem

$$(A^T A + \lambda D^T D)x = A^T b.$$

Dacă matricea din stânga este pozitiv definită, soluţia este unică. În cazul particular $D = I$, aceasta este pozitiv definită pentru orice $\lambda > 0$, deoarece

$$z^T(A^T A + \lambda I)z = \|Az\|_2^2 + \lambda \|z\|_2^2 > 0 \qquad (z \neq 0).$$

Compromisul controlat de λ

Pentru λ mic, soluţia urmăreşte mai fidel datele şi poate reacţiona puternic la zgomot. Pentru λ mare, penalizarea domină şi soluţia poate deveni prea mică sau prea netedă. Parametrul λ controlează deci compromisul dintre fidelitatea faţă de date şi proprietatea impusă prin D.

Exemplu diagonal

Considerăm

$$A = \operatorname{diag}(1, 10^{-3}), \qquad b = \begin{pmatrix} 1 \\ 2 \cdot 10^{-3} \end{pmatrix}, \qquad D = I.$$

Pentru regularizarea Tikhonov, sistemul este diagonal şi soluţia este

$$x_1(\lambda) = \frac{1}{1+\lambda}, \qquad x_2(\lambda) = \frac{2 \cdot 10^{-6}}{10^{-6} + \lambda}.$$

Formula arată direct că o componentă slab determinată de date este amortizată mult mai puternic decât una bine determinată.

Laborator: netezirea unui semnal

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Generaţi $x_{\text{true}}(t) = \sin(2\pi t) + 0.3\sin(8\pi t)$ pe o grilă şi adăugaţi zgomot. Luaţi $A = I$ şi construiţi matricea D a diferenţelor de ordinul întâi. Rezolvaţi sistemul regularizat pentru

mai multe valori ale lui λ. Pentru fiecare valoare raportaţi $\|x-b\|_2$, $\|Dx\|_2$ şi, deoarece în simulare cunoaşteţi semnalul iniţial, $\|x-x_{\text{true}}\|_2$.

Exerciţii şi verificarea înţelegerii

1. Arătaţi echivalenţa dintre problema Tikhonov şi problema celor mai mici pătrate pentru sistemul augmentat.
2. Scrieţi matricea diferenţelor de ordinul întâi pentru un vector de lungime 5 şi determinaţi nucleul ei.
3. Demonstraţi că $A^TA+\lambda I$ este pozitiv definită pentru orice $\lambda>0$.
4. Pentru exemplul diagonal calculaţi soluţia pentru $\lambda=10^{-6},10^{-4},10^{-2}$ şi interpretaţi rezultatele.
5. Explicaţi de ce o valoare foarte mare a lui λ poate produce o soluţie puţin sensibilă la zgomot, dar nepotrivită pentru descrierea datelor.

Mini-proiect: restaurarea unui semnal

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Alegeţi un semnal 1D, adăugaţi zgomot şi comparaţi regularizarea cu $D=I$ şi cu matricea diferenţelor de ordinul întâi. Variaţi λ pe o grilă logaritmică. Pentru fiecare caz raportaţi fidelitatea faţă de date, mărimea penalizării şi, când semnalul iniţial este cunoscut, eroarea reală. Formulaţi o concluzie despre rolul lui D şi al lui λ.

Sarcini de aprofundare: regularizare şi modelare neliniară

Probleme de aprofundare: penalizare, zgomot şi compromis

1. Pentru problema $\min_x\{\|Ax-b\|_2^2+\lambda\|x\|_2^2\}$ deduceţi sistemul liniar satisfăcut de minimizator şi demonstraţi unicitatea pentru $\lambda>0$.
2. Pentru penalizarea generală $\|Dx\|_2^2$, deduceţi sistemul $(A^TA+\lambda D^TD)x=A^Tb$.
3. Analizaţi cazul scalar $ax\approx b$ cu penalizare λx^2. Scrieţi soluţia explicită şi studiaţi limitele $\lambda\to 0$ şi $\lambda\to\infty$.
4. Pentru D egală cu matricea diferenţelor finite de ordinul întâi, identificaţi vectorii pentru care $Dx=0$ şi interpretaţi ce tip de semnale sunt favorizate de penalizare.
5. Construiţi un exemplu diagonal în care datele sunt perturbate cu zgomot şi comparaţi soluţiile pentru mai multe valori ale lui λ.
6. Explicaţi de ce alegerea lui λ reprezintă un compromis între apropierea de date şi respectarea informaţiei a priori.
7. Pentru un semnal constant perturbat cu zgomot, explicaţi de ce penalizarea diferenţelor succesive este naturală. Cum s-ar schimba interpretarea pentru o penalizare

directă $\|x\|_2^2$?

8. Formulaţi problema de ajustare a unui cerc la puncte (x_i, y_i) ca problema celor mai mici pătrate în varianta neliniară, fără a propune încă un algoritm de rezolvare.
9. Formulaţi analog problema localizării unui punct din distanţele faţă de mai multe repere şi discutaţi situaţiile geometrice în care datele nu determină bine poziţia.
10. Comparaţi conceptual o problemă liniară LS cu una neliniară LS: ce rămâne aceeaşi în funcţia obiectiv şi ce se schimbă în dependenţa faţă de necunoscute?

Laborator integrator: eliminarea zgomotului prin regularizare

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Observaţi experimental compromisul fidelitate–netezime folosind numai instrumentele introduse până aici.

1. Construiţi un semnal sintetic $x_\star$ cu variaţii lente şi generaţi $b = x_\star + \eta$ pentru trei niveluri de zgomot.
2. Construiţi matricea diferenţelor finite de ordinul întâi D.
3. Pentru o grilă de valori λ, rezolvaţi sistemele $(I + \lambda D^T D)x_\lambda = b$ folosind Cholesky.
4. Pentru fiecare λ calculaţi $\|x_\lambda - b\|_2$, $\|Dx_\lambda\|_2$ şi, deoarece experimentul este sintetic, $\|x_\lambda - x_\star\|_2$.
5. Reprezentaţi separat dependenţa celor trei mărimi de λ şi explicaţi de ce minimele lor nu coincid.
6. Reprezentaţi câteva reconstrucţii corespunzătoare unei regularizări prea slabe, uneia rezonabile şi uneia prea puternice.
7. Repetaţi pentru cele trei niveluri de zgomot şi observaţi cum se modifică intervalul valorilor utile ale lui λ.
8. Verificaţi pentru fiecare soluţie ecuaţia liniară regularizată calculând vectorul dintre membrul stâng şi membrul drept şi raportând cea mai mare componentă în valoare absolută.

Proiect de capitol: formularea unei probleme de localizare

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Consideraţi poziţia necunoscută $p = (x, y)$ şi distanţe măsurate faţă de mai multe repere cunoscute.

1. **Modelare.** Scrieţi reziduurile neliniare şi funcţia obiectiv ca sumă de pătrate.
2. **Geometrie.** Desenaţi cercurile corespunzătoare distanţelor exacte pentru trei repere şi identificaţi poziţia în cazul ideal.

3. **Zgomot.** Perturbaţi distanţele şi arătaţi geometric de ce cercurile nu mai au, în general, un punct comun.
4. **Ponderi.** Introduceţi ponderi diferite pentru măsurători şi explicaţi ce semnificaţie fizică pot avea.
5. **Date insuficiente.** Analizaţi ce se întâmplă dacă reperele sunt prea puţine sau dispuse într-o geometrie nefavorabilă.
6. **Regularizare.** Dacă există o poziţie aproximativă p_0, propuneţi un termen de penalizare care să favorizeze soluţiile apropiate de p_0.
7. **Pregătire pentru capitolele următoare.** Identificaţi ce dificultate suplimentară apare faţă de LS liniar şi ce tip de algoritm ar trebui construit ulterior pentru minimizarea efectivă.

Test de capitol

Test – Regularizarea problemei celor mai mici pătrate

1. Calcul explicit. Pentru

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1.01 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ 2 \end{pmatrix},$$

consideraţi problema Tikhonov

$$\min_x \Big(\|Ax - b\|_2^2 + \lambda \|x\|_2^2 \Big), \qquad \lambda > 0.$$

Deduceţi sistemul liniar satisfăcut de minimizator şi calculaţi soluţia pentru $\lambda = 0.1$. Explicaţi rolul termenului $\lambda \|x\|_2^2$.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi.

A) regularizarea introduce un compromis între potrivirea datelor şi mărimea soluţiei;

B) alegerea lui λ nu influenţează soluţia;

C) pentru $\lambda > 0$, matricea $A^T A + \lambda I$ este pozitiv definită;

D) regularizarea este utilă când datele sunt perturbate sau problema este sensibilă.

Capitolul 8

Factorizarea QR

Definiţie 8.0.1. *O matrice $A \in \mathbb{R}^{m\times n}$, cu $m \geq n$, admite o factorizare QR dacă există o matrice ortogonală $Q \in \mathbb{R}^{m\times m}$ şi o matrice trapezoidală superioară $R \in \mathbb{R}^{m\times n}$, cu rânduri nule începând de la al $(n+1)$-lea, astfel încât $A = QR$.*

Este de asemenea posibil să se genereze o versiune redusă a factorizării QR, aşa cum este afirmat în rezultatul următor.

Teoremă 8.0.2 (Factorizarea QR redusă)**.** *Fie $A \in \mathbb{R}^{m\times n}$, cu $m \geq n$, şi presupunem că* $\operatorname{rank} A = n$. *Atunci există o factorizare*

$$A = \widetilde{Q}\widetilde{R},$$

unde $\widetilde{Q} \in \mathbb{R}^{m\times n}$ are coloane ortonormale, iar $\widetilde{R} \in \mathbb{R}^{n\times n}$ este triunghiulară superior. Dacă impunem $\widetilde{r}_{ii} > 0$ pentru $i = 1, \ldots, n$, această factorizare este unică. În plus,

$$A^T A = \widetilde{R}^T \widetilde{R},$$

deci $\widetilde{R}$ este factorul Cholesky al lui $A^T A$.

Demonstraţie. Existenţa va fi construită explicit mai jos prin ortogonalizarea coloanelor lui A. Pentru unicitate, presupunem că $A = Q_1 R_1 = Q_2 R_2$, cu aceleaşi proprietăţi şi diagonalele lui R_1, R_2 pozitive. Deoarece ambele familii de coloane ortonormale generează $\mathcal{R}(A)$, există o matrice ortogonală $C \in \mathbb{R}^{n\times n}$ astfel încât $Q_1 = Q_2 C$. Într-adevăr, putem lua $C = Q_2^T Q_1$, iar $Q_2 Q_2^T$ este proiecţia ortogonală pe $\mathcal{R}(A)$, deci

$$C^T C = Q_1^T Q_2 Q_2^T Q_1 = Q_1^T Q_1 = I.$$

Din $Q_2 C R_1 = Q_2 R_2$ şi $Q_2^T Q_2 = I$ rezultă $C = R_2 R_1^{-1}$. Aşadar C este simultan ortogonală şi triunghiulară superior. O asemenea matrice este diagonală cu elemente diagonale ± 1. Pozitivitatea diagonalelor lui R_1 şi R_2 obligă toate semnele să fie $+1$, deci $C = I$, $R_1 = R_2$ şi $Q_1 = Q_2$.

În final, $A^T A = R^T Q^T Q R = R^T R$. Matricea $A^T A$ este SPD deoarece A are rang coloană complet, iar R este triunghiulară superior cu diagonală pozitivă. Prin unicitatea factorizării Cholesky, R este factorul Cholesky al lui $A^T A$. □

Construcţia factorizării reduse prin Gram–Schmidt

Putem demonstra direct existenţa factorizării speciale apoi să considerăm că mulţimea vectorilor liniari independenţi daţi de coloanele lui $\widetilde{Q}$ este completată până la o bază iar R se obţine adăgând linii nule.

Presupunem aplicarea algoritmului Gram–Schmidt asupra coloanelor matricei

$$A = (a_1 \mid \cdots \mid a_n).$$

Proiecţia unui vector a pe direcţia unui vector q se notează

$$\operatorname{proj}_q(a) = \frac{\langle q, a\rangle}{\langle q, q\rangle}\, q, \qquad \langle v, w\rangle = v^T w,$$

unde $\langle \cdot, \cdot \rangle$ reprezintă produsul scalar standard.

Demonstraţie

Procedura începe cu:

$$u_1 = a_1, \qquad q_1 = \frac{u_1}{\|u_1\|},$$

iar vectorii următori se obţin prin eliminarea componentelor pe direcţiile deja determinate:

$$\begin{aligned}
u_2 &= a_2 - \operatorname{proj}_{q_1} a_2, & q_2 &= \frac{u_2}{\|u_2\|},\\
u_3 &= a_3 - \operatorname{proj}_{q_1} a_3 - \operatorname{proj}_{q_2} a_3, & q_3 &= \frac{u_3}{\|u_3\|},\\
&\vdots\\
u_k &= a_k - \sum_{j=1}^{k-1} \operatorname{proj}_{q_j} a_k, & q_k &= \frac{u_k}{\|u_k\|}.
\end{aligned}$$

Demonstraţie

Rearanjarea acestor relaţii astfel încât vectorii a_i să fie pe partea stângă conduce la:

$$\begin{aligned}
a_1 &= q_1\|u_1\|,\\
a_2 &= \operatorname{proj}_{q_1} a_2 + q_2\|u_2\|,\\
a_3 &= \operatorname{proj}_{q_1} a_3 + \operatorname{proj}_{q_2} a_3 + q_3\|u_3\|,\\
&\vdots\\
a_k &= \sum_{j=1}^{k-1} \operatorname{proj}_{q_j} a_k + q_k\|u_k\|.
\end{aligned}$$

Demonstraţie

Rescriem

$$\begin{aligned}
a_1 &= q_1\|u_1\|,\\
a_2 &= \langle q_1, a_2\rangle q_1 + q_2\|u_2\|,\\
a_3 &= \langle q_1, a_3\rangle q_1 + \langle q_2, a_3\rangle q_2 + q_3\|u_3\|,\\
&\vdots\\
a_k &= \sum_{j=1}^{k-1} \langle q_j, a_k\rangle q_j + q_k\|u_k\|.
\end{aligned}$$

Toate aceste relaţii pot fi exprimate matriceal:

$$(q_1 \mid \ldots \mid q_n) \begin{pmatrix} \|u_1\| & \langle q_1, a_2\rangle & \langle q_1, a_3\rangle & \cdots \\ 0 & \|u_2\| & \langle q_2, a_3\rangle & \cdots \\ 0 & 0 & \|u_3\| & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Demonstraţie

Produsul matricial reproduce exact matricea originală A, astfel încât:

$$A = \widetilde{Q}\widetilde{R},$$

unde $\widetilde{Q} = (q_1 \mid \ldots \mid q_n)$ are coloane ortonormale, iar $\widetilde{R}$ este triunghiulară superior.
Cum $\widetilde{R} = \widetilde{Q}^T A$, obţinem explicit:

$$\widetilde{R} = \begin{pmatrix} \langle q_1, a_1\rangle & \langle q_1, a_2\rangle & \langle q_1, a_3\rangle & \cdots \\ 0 & \langle q_2, a_2\rangle & \langle q_2, a_3\rangle & \cdots \\ 0 & 0 & \langle q_3, a_3\rangle & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Observăm că:

$$\langle q_j, a_j\rangle = \|u_j\|, \qquad \langle q_j, a_k\rangle = 0 \text{ pentru } j > k,$$

iar relaţia relevantă pentru factorizarea redusă este

$$\widetilde{Q}^T\widetilde{Q} = I_n.$$

În general, dacă $m > n$, matricea $\widetilde{Q}$ nu este pătratică şi nu are inversă; în schimb, $\widetilde{Q}\widetilde{Q}^T$ este proiecţia ortogonală pe $\mathcal{R}(A)$. Dacă completăm coloanele lui $\widetilde{Q}$ la o bază ortonormală a lui $\mathbb{R}^m$, obţinem matricea pătratică ortogonală Q, pentru care $Q^{-1} = Q^T$.

Dacă A are rangul n (adică, rang complet), atunci vectorii coloană ai lui $\widetilde{Q}$ formează o bază ortonormală pentru spaţiul vectorial

$$\text{range}(A) = \{y \in \mathbb{R}^m \,:\, y = A\,x \quad \text{pentru} \quad x \in \mathbb{R}^n\}.$$

Ca o consecinţă, construirea factorizării QR poate fi interpretată şi ca o procedură pentru generarea unei baze ortonormale pentru un set dat de vectori.

Dacă A are rangul $r < n$, factorizarea QR nu conduce neapărat la o bază ortonormală pentru range(A). Totuşi, se poate obţine o factorizare de forma

$$Q^T A\, P = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix},$$

unde Q este ortogonală, P este o matrice de permutare şi R_{11} este o matrice triunghiulară superior nesingulară de ordin r.

În general, când folosim factorizarea QR, ne vom referi întotdeauna la forma sa redusă, deoarece are aplicaţie în rezolvarea sistemelor supradeterminate.

Factorii matriciali $\widetilde{Q}$ şi $\widetilde{R}$ pot fi calculaţi utilizând ortogonalizarea Gram-Schmidt.

În continuare, impunând ca $A = \widetilde{Q}\widetilde{R}$ şi folosind faptul că $\widetilde{Q}$ are coloane ortonormale (adică, $\widetilde{Q}^T\widetilde{Q} = I_n$), elementele lui $\widetilde{R}$ pot fi calculate cu uşurinţă.

De asemenea, este de remarcat faptul că, dacă A are rang complet, matricea A^TA este simetrică şi pozitiv definită, şi, prin urmare, admite o factorizare Cholesky unică de forma H^TH. Pe de altă parte, deoarece ortogonalitatea lui $\widetilde{Q}$ implică

$$H^TH = A^TA = \widetilde{R}^T\widetilde{Q}^T\widetilde{Q}\widetilde{R} = \widetilde{R}^T\widetilde{R}, \tag{8.1}$$

concluzionăm că $\widetilde{R}$ este, de fapt, factorul Cholesky H al lui A^TA.

Astfel, elementele diagonale ale lui $\widetilde{R}$ sunt nenule dacă şi numai dacă A are rang coloană complet.

Metoda Gram-Schmidt modificată

Metoda Gram-Schmidt are o utilitate practică redusă, deoarece vectorii generaţi îşi pierd independenţa liniară din cauza erorilor de rotunjire. Deşi este corectă din punct de vedere matematic, ea prezintă o dificultate majoră în practică: *este instabilă numeric*, în special atunci când vectorii sunt aproape dependenţi liniar sau când erorile de rotunjire se acumulează.

Pentru a îmbunătăţi stabilitatea, se utilizează *Metoda Gram–Schmidt modificată (Modified Gram–Schmidt, MGS)*, o variantă echivalentă teoretic, dar mai stabilă numeric.

În forma clasică, vectorul nou

$$u_k = a_k - \sum_{j=1}^{k-1} \langle q_j, a_k \rangle q_j$$

se obţine prin scăderea dintr-o singură operaţie a tuturor proiecţiilor pe vectorii ortonormali anteriori. Această operaţie implică scăderi între valori apropiate, ceea ce conduce la pierderea de precizie în aritmetica flotantă.

Metoda modificată reorganizează calculele astfel încât proiecţiile sunt eliminate *una câte una*, actualizând vectorul la fiecare pas:

$$\begin{aligned} u_k^{(0)} &= a_k, \\ u_k^{(j)} &= u_k^{(j-1)} - \langle q_j, u_k^{(j-1)} \rangle q_j, \qquad j = 1, 2, \ldots, k-1, \\ u_k &= u_k^{(k-1)}. \end{aligned}$$

Această procedură reduce riscul de anulări numerice şi limitează propagarea erorilor.

În varianta Gram–Schmidt, factorul $\widetilde{Q}$ nu poate fi suprascris pur şi simplu peste A. În varianta Householder, în schimb, triunghiul superior al lui A poate fi suprascris cu R, iar vectorii Householder se stochează compact sub diagonală; Q nu trebuie format explicit, ci se aplică prin produsul reflectorilor.

Extensie didactică: ortogonalizarea ca instrument numeric

Factorizarea QR scrie o matrice $A \in \mathbb{R}^{m\times n}$, de regulă cu $m \geq n$ şi rang coloană complet, sub forma $A = QR$, unde coloanele lui Q sunt ortonormale şi R este triunghiulară superior. În forma redusă, $Q \in \mathbb{R}^{m\times n}$ şi $R \in \mathbb{R}^{n\times n}$; în forma completă, Q este pătratică ortogonală.

De ce este ortogonalitatea atât de valoroasă?

O matrice ortogonală nu schimbă norma euclidiană: $\|Qx\|_2 = \|x\|_2$. Aplicarea lui Q sau Q^T nu amplifică geometric vectorii, iar transformările ortogonale sunt, în general, numeric stabile. Factorizarea QR mută dificultatea într-un sistem triunghiular $Rx = Q^Tb$.

Exemplu Gram–Schmidt

Luăm coloanele $a_1 = (1,1,0)^T$ şi $a_2 = (1,0,1)^T$. Avem $q_1 = a_1/\|a_1\| = (1,1,0)^T/\sqrt{2}$ şi $r_{12} = q_1^T a_2 = 1/\sqrt{2}$. Eliminând proiecţia, obţinem $u_2 = a_2 - r_{12} q_1 = (1/2, -1/2, 1)^T$, cu $\|u_2\| = \sqrt{6}/2$. Prin urmare $q_2 = (1,-1,2)^T/\sqrt{6}$ şi

$$R = \begin{pmatrix} \sqrt{2} & 1/\sqrt{2} \\ 0 & \sqrt{6}/2 \end{pmatrix}.$$

Verificarea $Q^T Q = I$ este la fel de importantă ca verificarea $QR = A$.

Clasic versus modificat

În aritmetică exactă, Gram–Schmidt clasic şi modificat produc aceeaşi descompunere, în limitele semnelor. În aritmetică finită, varianta modificată tinde să păstreze mai bine ortogonalitatea atunci când coloanele sunt aproape dependente. Pentru robusteţe şi performanţă în bibliotecile numerice, reflectorii Householder sunt de obicei instrumentul standard pentru factorizarea QR densă.

Un reflector Householder are forma $H = I - 2vv^T/(v^T v)$ şi este ortogonal şi simetric. Vectorul v se alege astfel încât Hx să anuleze toate componentele lui x sub prima. Aplicând succesiv reflectori pe coloane, construim R fără a forma neapărat explicit Q.

Laborator: pierderea ortogonalităţii

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi o matrice ale cărei coloane sunt aproape dependente, de exemplu $a_j = v + \varepsilon w_j$ cu ε mic. Implementaţi Gram–Schmidt clasic şi modificat. Pentru fiecare calculaţi $\|I - Q^T Q\|_2$ şi $\|A - QR\|_2$. Comparaţi cu `qr(A,0)`. Variaţi ε şi identificaţi momentul în care metoda clasică începe să piardă vizibil ortogonalitatea.

Exerciţii şi verificarea înţelegerii

1. Refaceţi complet exemplul cu cele două coloane şi verificaţi $A = QR$.
2. Demonstraţi că reflectorul Householder $H = I - 2vv^T/(v^T v)$ satisface $H^T = H$ şi $H^T H = I$.
3. Explicaţi de ce produsul mai multor matrici ortogonale este ortogonal.
4. Dacă $A = QR$ are rang coloană complet, arătaţi că R este inversabilă.
5. Ce informaţie despre rang poate fi dedusă din elementele diagonale ale lui R în aritmetică exactă?

Mini-proiect: implementarea reflectorilor Householder

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un

parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Implementaţi o factorizare QR bazată pe reflectori. Stocaţi vectorii Householder compact sub diagonala lui R şi evitaţi construirea matricilor H_k complete. Scrieţi o funcţie separată care aplică Q^T unui vector folosind reflectorii stocaţi. Comparaţi memoria şi timpul cu o implementare care formează explicit toate matricile.

Exemplu rezolvat: Gram–Schmidt pe o matrice dreptunghiulară

Fie

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = (a_1 \; a_2).$$

Avem $\|a_1\|_2 = \sqrt{2}$, deci $q_1 = (1,1,0)^T/\sqrt{2}$. Proiecţia celei de-a doua coloane pe q_1 este $r_{12} = q_1^T a_2 = 1/\sqrt{2}$. Eliminând această componentă,

$$u_2 = a_2 - r_{12} q_1 = \begin{pmatrix} 1/2 \\ -1/2 \\ 1 \end{pmatrix},$$

iar $\|u_2\|_2 = \sqrt{3/2}$. Prin urmare

$$q_2 = \frac{1}{\sqrt{6}} \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}.$$

Factorizarea redusă este $A = \widetilde{Q}\widetilde{R}$, cu

$$\widetilde{Q} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{6} \\ 1/\sqrt{2} & -1/\sqrt{6} \\ 0 & 2/\sqrt{6} \end{pmatrix}, \quad \widetilde{R} = \begin{pmatrix} \sqrt{2} & 1/\sqrt{2} \\ 0 & \sqrt{3/2} \end{pmatrix}.$$

Verificarea didactică trebuie să includă trei relaţii distincte: $A \approx QR$, $Q^TQ \approx I$ şi forma triunghiulară a lui R.

De ce ortogonalitatea este atât de valoroasă numeric?

Înmulţirea cu o matrice ortogonală nu modifică norma euclidiană: $\|Qx\|_2 = \|x\|_2$. Ca urmare, transformările ortogonale nu amplifică eroarea doar prin schimbarea coordonatelor. Aceasta este una dintre explicaţiile pentru robusteţea algoritmilor QR bazaţi pe reflectori Householder sau rotaţii Givens.

Gram–Schmidt clasic, modificat şi Householder

Cele trei proceduri pot produce aceeaşi factorizare în aritmetică exactă, dar nu au acelaşi comportament în virgulă mobilă. Gram–Schmidt clasic calculează toate proiecţiile faţă de vectorul iniţial al pasului şi poate pierde ortogonalitatea când coloanele sunt aproape dependente. Varianta modificată actualizează succesiv vectorul şi este mai robustă. Reflectorii Householder sunt, în general, alegerea standard pentru o factorizare QR densă stabilă.

Sarcină practică: pierderea ortogonalităţii

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi Gram–Schmidt clasic şi modificat. Construiţi o matrice ale cărei coloane sunt aproape dependente, de exemplu coloane de tip $(1, t, t^2, \ldots)$ evaluate în puncte foarte apropiate. Pentru fiecare algoritm calculaţi $\|I - Q^TQ\|_2$ şi $\|A - QR\|_2/\|A\|_2$. Comparaţi cu `qr(A,0)`. Creşteţi treptat condiţionarea şi identificaţi când diferenţele devin vizibile.

Exerciţii suplimentare

1. Verificaţi manual că cei doi vectori q_1, q_2 din exemplu sunt ortonormali.
2. Completaţi $\widetilde{Q}$ la o matrice ortogonală $Q \in \mathbb{R}^{3\times 3}$.
3. Arătaţi că dacă $A = QR$ şi Q este ortogonală, atunci $A^TA = R^TR$.
4. De ce semnele coloanelor lui Q şi ale liniilor corespunzătoare din R pot fi schimbate simultan fără a modifica produsul?
5. Formulaţi criteriul de rang complet în termenii diagonalei lui R.

8.1 Sisteme dreptunghiulare rezolvate cu factorizarea QR

Sisteme supradeterminate cu factorizarea QR

Teoremă 8.1.1. *Fie $A \in \mathbb{R}^{m\times n}$, cu $m \geq n$, o matrice de rang complet. Atunci soluţia unică a problemei*

$$\min_{x\in\mathbb{R}^n} \underbrace{\|Ax - b\|_2^2}_{:=\Phi(x)}$$

este dată de

$$x^* = \widetilde{R}^{-1}\widetilde{Q}^Tb, \tag{8.2}$$

unde $\widetilde{R} \in \mathbb{R}^{n\times n}$ şi $\widetilde{Q} \in \mathbb{R}^{m\times n}$ sunt matricile obţinute din factorizarea QR a lui A. Mai mult, minimul lui Φ este dat de

$$\Phi(x^*) = \sum_{i=n+1}^{m} [(Q^Tb)_i]^2. \tag{8.3}$$

Demonstraţie. Factorizarea redusă QR a lui A există şi, alegând diagonala pozitivă a lui R, este unică, deoarece A are rang complet. Astfel, există două matrici, $Q \in \mathbb{R}^{m\times m}$ şi $R \in \mathbb{R}^{m\times n}$, astfel încât $A = QR$, unde Q este ortogonală.

Deoarece matricile ortogonale păstrează produsul scalar euclidian, rezultă că

$$\|Ax - b\|_2^2 = \|Rx - Q^Tb\|_2^2. \tag{8.4}$$

Aducându-ne aminte că R este trapezoidală superioară, avem

$$\|Rx - Q^Tb\|_2^2 = \|\widetilde{R}x - \widetilde{Q}^Tb\|_2^2 + \sum_{i=n+1}^{m} [(Q^Tb)_i]^2, \tag{8.5}$$

aşa încât minimul este atins exact atunci când $\widetilde{R}x = \widetilde{Q}^T b$. Deoarece $\widetilde{R}$ este inversabilă, rezultă $x^* = \widetilde{R}^{-1}\widetilde{Q}^T b$, iar valoarea minimă este termenul $\sum_{i=n+1}^{m}[(Q^T b)_i]^2$. □

Dacă A nu are rang complet, tehnicile de rezolvare de mai sus eşuează, deoarece, în acest caz, dacă x^* este o soluţie a problemei

$$\min_{x\in\mathbb{R}^n} \underbrace{\|Ax-b\|_2^2}_{:=\Phi(x)},$$

atunci vectorul $x^* + z$, cu $z \in \ker(A)$, este de asemenea o soluţie. Prin urmare, trebuie introdusă o constrângere suplimentară pentru a impune unicitatea soluţiei.

De obicei, se impune ca x^* să aibă norma euclidiană minimă, astfel încât problema celor mai mici pătrate poate fi formulată astfel:

$$\begin{aligned} &\text{găsiţi } \; x^* \in \mathbb{R}^n \;\; \text{cu norma euclidiană minimă astfel încât} \\ &\|A\,x^* - b\|_2^2 = \min_{x\in\mathbb{R}^n} \underbrace{\|Ax-b\|_2^2}_{:=\Phi(x)}. \end{aligned} \tag{8.6}$$

Această problemă va fi rezolvată într-un capitol ulterior, după introducerea descompunerii în valori singulare. Până atunci, reţinem doar că lipsa rangului coloană complet conduce la neunicitate şi motivează selectarea soluţiei de normă euclidiană minimă.

Reflectori Householder

Definiţie 8.1.2. *Fie $v \in \mathbb{R}^m$, $\|v\|_2 = 1$. Matricea*

$$H = I_m - 2vv^T$$

se numeşte reflector Householder. Ea este simetrică şi ortogonală: $H^T = H$ şi $H^T H = I_m$.

Dacă folosim un vector nenormalizat $u \neq 0$, aceeaşi transformare se scrie

$$H = I_m - \frac{2}{u^T u}uu^T = I_m - \frac{1}{\beta}uu^T, \qquad \beta = \frac{\|u\|_2^2}{2}.$$

Alegerea stabilă a vectorului Householder

Fie $a \in \mathbb{R}^p$, $a \neq 0$, şi $e_1 = (1, 0, \ldots, 0)^T$. Alegerea

$$u = a + \operatorname{sign}(a_1)\|a\|_2 e_1, \qquad \operatorname{sign}(0) := 1,$$

evita atât cazul $u = 0$, cât şi anularea catastrofală care poate apărea în formula $a - \|a\|e_1$ atunci când $a_1 \approx \|a\|$. Pentru reflectorul asociat lui u avem

$$Ha = -\operatorname{sign}(a_1)\|a\|_2 e_1, \qquad \|u\|_2^2 = 2\|a\|_2\Big(\|a\|_2 + |a_1|\Big) > 0.$$

Dacă $a = 0$, nu este nevoie de reflector şi pasul se omite.

Alegerea semnului

Formula $u = a - \|a\|e_1$ dă $u = 0$ pentru $a = \|a\|e_1$ şi poate pierde precizie când $a_1 > 0$ este foarte apropiat de $\|a\|$. Alegerea semnului evită aceste două situaţii.

Factorizarea QR prin reflectori Householder

Fie $A \in \mathbb{R}^{m\times n}$, $m \geq n$. La pasul k considerăm subcoloana activă

$$z_k = A^{(k-1)}(k:m, k) \in \mathbb{R}^{m-k+1}.$$

Dacă $z_k \neq 0$, definim

$$\alpha_k = -\operatorname{sign}((z_k)_1)\|z_k\|_2, \qquad v_k = z_k - \alpha_k e_1,$$

şi reflectorul redus

$$\widetilde{H}_k = I - 2\frac{v_k v_k^T}{v_k^T v_k}.$$

Îl extindem la $\mathbb{R}^m$ prin $H_k = \operatorname{diag}(I_{k-1}, \widetilde{H}_k)$. Atunci

$$H_k A^{(k-1)}$$

are zerouri sub diagonală în coloana k, iar elementul diagonal nou este α_k. Dacă $z_k = 0$, luăm $H_k = I$.

După n paşi,

$$R = H_n \cdots H_1 A, \qquad Q = H_1 \cdots H_n, \qquad A = QR,$$

deoarece fiecare H_k este simetric şi ortogonal. Semnele elementelor diagonale ale lui R depind de convenţia de mai sus. Dacă se doreşte o factorizare redusă cu diagonală pozitivă, se schimbă simultan semnul coloanei corespunzătoare din Q şi al liniei corespunzătoare din R.

Stocare compactă

În implementarea Householder nu este necesară formarea explicită a lui Q. Triunghiul superior al lui A poate fi suprascris cu R, iar vectorii Householder pot fi stocaţi compact sub diagonală (de exemplu normalizaţi astfel încât prima lor componentă activă să fie 1). Aplicarea lui Q sau Q^T se face ulterior ca produs de reflectori.

Completare didactică: cele mai mici pătrate prin QR

Dacă $A = QR$ este o factorizare redusă cu $Q^TQ = I$ şi R triunghiulară superior, atunci $\|Ax - b\|_2 = \|QRx - b\|_2$. Completând Q la o bază ortonormală $\widehat{Q} = (Q, Q_\perp)$ şi folosind invarianta normei la transformări ortogonale, obţinem

$$\|Ax - b\|_2^2 = \|Rx - Q^Tb\|_2^2 + \|Q_\perp^T b\|_2^2.$$

Al doilea termen nu depinde de x, deci minimizatorul rezolvă sistemul triunghiular $Rx = Q^Tb$. Acesta este mecanismul prin care QR evită formarea lui A^TA.

Exemplu

Pentru matricea din exemplul anterior şi un $b \in \mathbb{R}^3$, calculăm mai întâi cele două componente ale lui Q^Tb, apoi rezolvăm sistemul 2×2 superior triunghiular. Componenta lui b ortogonală pe imaginea lui A rămâne exact reziduul minim.

O proiecţie fără matricea de proiecţie

Proiecţia ortogonală a lui b pe $\mathcal{R}(A)$ este QQ^Tb. Nu este nevoie să construim matricea densă QQ^T; aplicăm succesiv Q^T şi Q. În algoritmii numerici, evitarea matricilor intermediare este adesea la fel de importantă ca alegerea formulei corecte.

Pentru sisteme subdeterminate sau matrici cu rang deficient sunt necesare precauţii suplimentare. În acest capitol ne limităm la cazul cu rang coloană complet, pentru care factorul triunghiular al factorizării QR reduse este inversabil. Cazul general şi alegerea soluţiei de normă minimă vor fi tratate într-un capitol ulterior.

Laborator: normal equations versus QR

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi o familie de matrici dreptunghiulare cu două coloane aproape dependente, de exemplu $a_2 = a_1 + \varepsilon v$, unde v nu este coliniar cu a_1 şi ε scade. Pentru un b construit dintr-o soluţie cunoscută, rezolvaţi problema prin ecuaţii normale şi prin QR. Comparaţi eroarea în x, reziduul şi condiţionarea deja definită în capitolele anterioare.

Exerciţii şi verificarea înţelegerii

1. Derivaţi identitatea de descompunere a normei folosind matricea ortogonală completă $\widehat{Q}$.
2. Demonstraţi că reziduul LS este $r = (I - QQ^T)b$ când A are rang coloană complet.
3. Arătaţi că QQ^T este simetrică şi idempotentă.
4. Comparaţi costul formării lui A^TA cu costul unei factorizări QR pentru o matrice densă $m \times n$, $m \gg n$, la nivel de ordine de mărime.
5. De ce nu este recomandat să calculaţi $x = R^{-1}Q^Tb$ prin formarea explicită a lui R^{-1}?

Mini-proiect: ajustare polinomială stabilă

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Reluaţi proiectul de ajustare a datelor cu polinoame de grade crescătoare. Implementaţi două rezolvări: ecuaţii normale şi QR. Pentru date pe intervalul $[0, 10]$, repetaţi după rescalarea variabilei pe $[-1, 1]$. Comparaţi condiţionarea, coeficienţii şi predicţiile. Identificaţi separat efectul algoritmului şi efectul alegerii bazei.

Exemplu rezolvat: cele mai mici pătrate prin QR fără ecuaţii normale

Folosim matricea

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix},$$

şi factorizarea redusă $A = \widetilde{Q}\widetilde{R}$ calculată anterior. Problema $\min_x \|Ax-b\|_2$ devine $\min_x \|\widetilde{Q}\widetilde{R}x-b\|_2$. Completând $\widetilde{Q}$ la o matrice ortogonală şi folosind invarianta normei, minimizarea se reduce la sistemul triunghiular

$$\widetilde{R}x = \widetilde{Q}^T b.$$

Rezultatul este $x = (5/3, -1/3)^T$. Vectorul ajustat este $Ax = (4/3, 5/3, -1/3)^T$, deci reziduul $Ax - b = (1, -1, -1)^T/3$. Verificăm $A^T(Ax - b) = 0$.

Avantajul QR nu este că schimbă minimizatorul, ci că îl calculează fără a forma A^TA. Dacă A este rău condiţionată, evităm astfel pătrăţirea numărului de condiţionare asociată ecuaţiilor normale.

Sarcină de implementare: `leastSquaresQR`

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o funcţie care primeşte $A \in \mathbb{R}^{m\times n}$, $m \geq n$, calculează QR redus şi rezolvă sistemul triunghiular. Întoarceţi x, reziduul şi o estimare a rangului bazată pe diagonala lui R. Comparaţi cu ecuaţiile normale pe matrici bine şi rău condiţionate. Pentru fiecare test, raportaţi şi $\|A^Tr\|_2$, nu numai $\|r\|_2$.

Exerciţii suplimentare

1. Calculaţi explicit $\widetilde{Q}^Tb$ pentru exemplul de mai sus şi efectuaţi substituţia înapoi.
2. Arătaţi prin descompunerea ortogonală a lui b că norma minimă a reziduului este norma componentei lui b ortogonale pe $\mathcal{R}(A)$.
3. Ce se întâmplă în algoritm dacă o diagonală a lui R este foarte mică?
4. Explicaţi de ce QR cu pivotare de coloane este utilă pentru detectarea rangului numeric.
5. Comparaţi numărul de operaţii pentru ecuaţii normale plus Cholesky şi pentru QR Householder. De ce alegem uneori totuşi QR?

Fişă de seminar şi laborator: factorizarea QR

Problema rezolvată 1: reflector Householder în două dimensiuni

Fie $x = (3, 4)^T$, cu $\|x\| = 5$. Dorim un reflector care să transforme x într-un multiplu de e_1. Alegem, de exemplu, $v = x + 5e_1 = (8, 4)^T$ şi $H = I - 2vv^T/(v^Tv)$. Cum $v^Tv = 80$, obţinem

$$H = I - \frac{1}{40}\begin{pmatrix} 64 & 32 \\ 32 & 16 \end{pmatrix} = \begin{pmatrix} -0.6 & -0.8 \\ -0.8 & 0.6 \end{pmatrix}.$$

Atunci $Hx = (-5, 0)^T$. Semnul negativ nu este o problemă; important este anularea celei de-a doua componente. Alegerea semnului în construcţia lui v este făcută în practică astfel încât să evităm scăderea a două numere apropiate.

Problema rezolvată 2: QR şi determinant

Pentru o matrice pătratică $A = QR$, avem $|\det Q| = 1$, iar $\det R$ este produsul elementelor diagonale. Prin urmare $|\det A| = \prod_i |r_{ii}|$. Totuşi, ca şi în cazul LU, determinantul singur nu este un substitut pentru numărul de condiţionare atunci când vrem să evaluăm sensibilitatea numerică.

Laborator integrator 8: trei QR-uri

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi Gram–Schmidt clasic, Gram–Schmidt modificat şi QR Householder sau folosiţi o implementare proprie pentru primele două şi rutina standard pentru Householder. Testaţi matrici Vandermonde şi matrici cu coloane aproape dependente. Pentru fiecare raportaţi două erori: $\|A - QR\|/\|A\|$ şi $\|I - Q^TQ\|$. Arătaţi experimental că un reziduu de factorizare bun nu garantează automat ortogonalitate bună pentru Gram–Schmidt clasic.

Set de probleme

1. Verificaţi că reflectorul din exemplu este simetric şi ortogonal.
2. Arătaţi că un reflector Householder are valori proprii 1 şi -1.
3. Construiţi un reflector care transformă $(1, 2, 2)^T$ în $\pm 3e_1$.
4. Explicaţi de ce aplicarea succesivă a reflectorilor produce zerouri sub diagonală fără a reintroduce zerouri deja obţinute.
5. Ce avantaj au rotaţiile Givens pentru matrici rare faţă de reflectori globali?
6. De ce QR este preferată ecuaţiilor normale în LS rău condiţionat?
7. Cum poate pivotarea de coloane ajuta la estimarea rangului numeric?
8. Formulaţi rezolvarea LS prin QR redus într-un algoritm de maximum şase paşi.

Proiect de capitol: QR pentru regresie stabilă

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4.**

MATLAB: scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Implementaţi un algoritm pentru cele mai mici pătrate bazat pe QR şi aplicaţi-l la ajustarea polinoamelor în bază monomială pentru grade crescătoare. Comparaţi cu ecuaţiile normale. Măsuraţi condiţionarea lui A, eroarea de reconstrucţie şi diferenţa dintre coeficienţi. Repetaţi după rescalarea absciselor şi explicaţi ce îmbunătăţire provine din modelare şi ce îmbunătăţire provine din algoritm.

Sarcini de aprofundare şi sinteză pentru factorizarea QR

Probleme de aprofundare: ortogonalitate, stabilitate şi LS

1. Pentru două coloane $a_1, a_2 \in \mathbb{R}^m$, executaţi manual Gram–Schmidt clasic şi modificat. Arătaţi că în aritmetică exactă produc aceeaşi descompunere QR, până la convenţia semnelor.
2. Construiţi două coloane aproape paralele şi explicaţi de ce scăderea unor vectori apropiaţi poate produce pierdere de ortogonalitate în Gram–Schmidt clasic.
3. Demonstraţi că un reflector Householder $H = I - 2vv^T/(v^Tv)$ este ortogonal şi simetric. Ce valori proprii poate avea?
4. Pentru o matrice $A \in \mathbb{R}^{m\times n}$ cu rang coloană complet şi $A = QR$, deduceţi soluţia LS din minimizarea $\|Rx - Q^Tb\|_2$.
5. Arătaţi că $\kappa_2(QA) = \kappa_2(A)$ pentru Q ortogonală. Explicaţi de ce transformările ortogonale sunt atractive numeric.
6. Comparaţi costul formării explicite a lui Q cu aplicarea succesivă a reflectorilor Householder unui vector. În ce situaţii nu este nevoie de Q explicit?
7. Pentru o matrice pătratică nesingulară, arătaţi cum QR oferă o metodă de rezolvare a sistemului $Ax = b$. Comparaţi costul cu LU şi explicaţi de ce QR nu este metoda implicită pentru orice sistem pătratic.
8. Construiţi o matrice dreptunghiulară cu o coloană aproape combinaţie liniară a celorlalte. Urmăriţi diagonala lui R şi explicaţi ce indică apariţia unui element diagonal foarte mic despre independenţa numerică a coloanelor.
9. Dacă A este deja cu coloane ortonormate, ce factorizare QR obţineţi direct? Cum ar trebui să se comporte un algoritm stabil numeric?
10. Pentru o matrice Vandermonde pe puncte foarte apropiate, anticipaţi comportamentul defectului de ortogonalitate pentru CGS, MGS şi Householder.
11. Explicaţi de ce ecuaţiile normale pot păstra un reziduu LS rezonabil chiar atunci când coeficienţii sunt mult mai puţin precişi decât cei obţinuţi prin QR.
12. Formulaţi trei teste numerice pentru o factorizare calculată: defectul de ortogonalitate, reziduul factorizării şi reziduul problemei rezolvate.

Laborator integrator: când se pierde ortogonalitatea?

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă

este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Comparaţi CGS, MGS şi Householder pe matrici din ce în ce mai aproape de deficienţă de rang.

1. Construiţi o familie A_ε cu coloane de forma $a_2 = a_1 + \varepsilon z_2$, $a_3 = a_1 + \varepsilon z_3$, pentru $\varepsilon = 10^{-1}, \dots, 10^{-12}$.
2. Implementaţi CGS şi MGS. Pentru Householder folosiţi fie o implementare proprie, fie rutina MATLAB doar ca referinţă, în funcţie de etapa laboratorului.
3. Pentru fiecare metodă raportaţi $\|I - Q^T Q\|_2$ şi $\|A - QR\|_F / \|A\|_F$.
4. Alegeţi $x_\star$, formaţi $b = A_\varepsilon x_\star + \eta$ şi rezolvaţi problema LS cu cele trei QR-uri.
5. Comparaţi eroarea coeficienţilor, reziduul şi condiţionarea lui A_ε.
6. Repetaţi după scalarea coloanelor la norme comparabile. Se îmbunătăţeşte ortogonalitatea? Se schimbă problema fizică dacă reveniţi corect la variabilele iniţiale?
7. Reprezentaţi pe scară logaritmică defectul de ortogonalitate în funcţie de ε.
8. Formulaţi o concluzie despre diferenţa dintre echivalenţa în aritmetică exactă şi comportamentul în virgulă mobilă.

Proiect de capitol: regresie polinomială stabilă

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Ajustaţi aceeaşi funcţie pe un interval prin polinoame de grade crescătoare.

1. **Date.** Generaţi date zgomotoase dintr-o funcţie netedă şi separaţi un set de validare.
2. **Matrici.** Construiţi matricile Vandermonde pentru gradele $3, 6, 10, 14$ pe variabila originală şi pe o variabilă rescalată la $[-1, 1]$.
3. **Calcul manual.** Pentru un caz mic, executaţi primul reflector Householder.
4. **Algoritm.** Rezolvaţi LS prin QR şi comparaţi cu ecuaţiile normale.
5. **Diagnostic.** Raportaţi condiţionarea, defectul de ortogonalitate şi diagonala lui R.
6. **Validare.** Comparaţi eroarea de antrenare şi validare şi identificaţi semne de supraînvăţare.
7. **Interpretare.** Discutaţi separat efectul gradului modelului, al scalării şi al algoritmului de rezolvare.

Test de capitol

Test – Factorizarea QR

1. Calcul explicit şi verificare critică. Pentru

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

construiţi factorizarea QR redusă prin procedeul Gram–Schmidt prezentat în capitol. Verificaţi explicit relaţiile $Q^TQ = I$ şi $A = QR$.

2. Cod MATLAB. Pentru o problemă supradeterminată cu coloane liniar independente se foloseşte:

```
[Q,R] = qr(A,0);
x = R\(Q'*b);
```

Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi:

A) codul foloseşte o factorizare QR redusă;

B) Q^Tb reprezintă coordonatele relevante pentru problema proiectată;

C) R este triunghiulară inferioară;

D) metoda evită formarea explicită a matricei A^TA.

Capitolul 9

Metode de descreştere şi metoda gradientului

Scop

Dorim să construim o soluţie aproximativă pentru problema de minim

$$\min_{x\in\mathbb{R}^n} f(x), \quad \text{unde} \quad f:\mathbb{R}^n\to\mathbb{R} \ \text{ este de clasă } \ C^1 \ \text{ pe } \ \mathbb{R}^n.$$

De ce?

Pentru că după cum am văzut, rezolvarea aproximativă a unui sistem de ecuaţii se reduce la o problemă de optimizare, adică găsirea unei valori minime a unei funcţii pătratice.

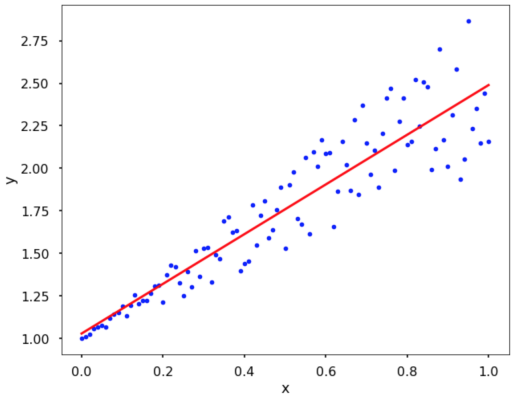
Corelarea datelor

$$f:\mathbb{R}^2\to\mathbb{R}, f(z)=\langle A\,z,z\rangle+2\langle b,z\rangle+c, \qquad A\in\mathbb{R}^{2\times 2}, \ \ b\in\mathbb{R}^2, \ \ c\in\mathbb{R}.$$

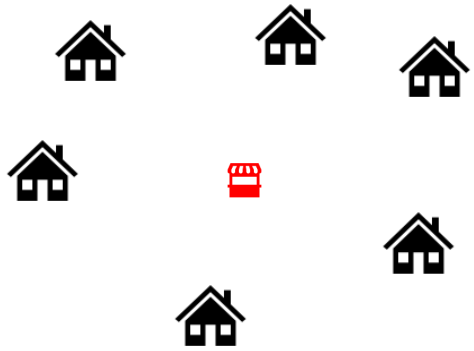
Alte probleme

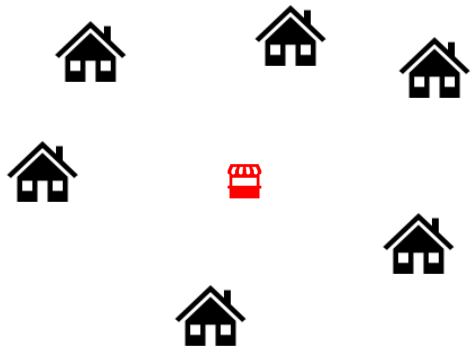
Alte probleme

$$\min_{x\in\mathbb{R}^n, r\in\mathbb{R}} \sum_{i=1}^{m} (\|x-a_i\|^2-r^2)^2.$$









Alte probleme

$$\min_{x\in\mathbb{R}^n, r\in\mathbb{R}} \sum_{i=1}^{m} \omega_i(\|x-a_i\|^2 - r^2)^2.$$

De ce aproximativ?

În majoritatea problemelor, abordările analitice obişnuite nu pot fi aplicate în practică din următoarele motive:

- ar putea fi o sarcină foarte dificilă să se rezolve sistemul de ecuaţii (de obicei neliniare) $\nabla f(x) = 0$;
- chiar dacă este posibilă găsirea tuturor punctelor staţionare, s-ar putea să existe un număr infinit de puncte staţionare, iar sarcina de a-l găsi pe cel care corespunde valorii minime a funcţiei este o problemă de optimizare care, prin ea însăşi, ar putea fi la fel de dificilă ca problema iniţială.

Extensie didactică: un sistem SPD ca problemă de minimizare

Pentru $A = A^T \succ 0$, rezolvarea sistemului $Ax = b$ este echivalentă cu minimizarea funcţiei strict convexe

$$f(x) = \frac{1}{2}x^T Ax - b^T x.$$

Într-adevăr, $\nabla f(x) = Ax - b$, deci punctul staţionar satisface exact sistemul. Hessiana este $\nabla^2 f = A \succ 0$, astfel încât punctul staţionar este unicul minim global.

De la algebră la geometrie

Curbele de nivel ale lui f sunt elipsoizi centraţi în soluţia $x_\star = A^{-1}b$. Valorile proprii ale lui A controlează curburile pe direcţiile principale. Dacă raportul $\lambda_{\max}/\lambda_{\min}$ este mare, elipsoizii sunt foarte alungiţi, iar o metodă care urmează gradientul poate zigzaga. Astfel, numărul de condiţionare capătă o interpretare geometrică directă.

Dacă definim reziduul liniar $r = b - Ax$, atunci direcţia de gradient negativ este $-\nabla f(x) = r$. Pentru sisteme SPD, metoda de cea mai rapidă descreştere poate fi astfel scrisă doar cu produse matrice–vector şi produse scalare; nu este nevoie de factorizare. Acesta este prototipul unor metode iterative mai avansate.

Exemplu 2×2

Fie $A = \mathrm{diag}(1, 10)$ şi $b = (1, 10)^T$, deci soluţia este $(1, 1)^T$. Curbele de nivel sunt mult mai înguste pe direcţia a doua. Pornind din origine, gradientul iniţial este $(-1, -10)^T$, dominat de direcţia cu curbură mare. După un pas optim, următorul gradient tinde să schimbe direcţia, producând traiectoria caracteristică de zigzag.

Laborator: vizualizarea unei metode

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru o funcţie pătratică în două variabile, reprezentaţi curbele de nivel şi punctele generate de metoda gradientului. Repetaţi pentru $A = \mathrm{diag}(1, 2)$, $\mathrm{diag}(1, 20)$ şi pentru o matrice rotită $Q^T \,\mathrm{diag}(1, 20) Q$. Comparaţi geometria şi numărul de paşi necesar pentru aceeaşi reducere a gradientului.

Exerciţii şi verificarea înţelegerii

1. Demonstraţi echivalenţa dintre $Ax = b$ şi minimizarea funcţiei pătratice atunci când A este SPD.
2. Completaţi pătratul şi arătaţi că $f(x) - f(x_\star) = \frac{1}{2}(x - x_\star)^T A (x - x_\star)$.
3. Deduceţi că eroarea în norma energetică $\|e\|_A = (e^T A e)^{1/2}$ măsoară direct diferenţa de valoare a funcţiei obiectiv.
4. Ce se schimbă dacă A este simetrică, dar are o valoare proprie negativă?
5. Explicaţi de ce produsul matrice–vector este operaţia dominantă într-o metodă de gradient pentru sisteme rare.

Mini-proiect: aceeaşi matrice, două paradigme

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Pentru o familie de matrice SPD rare, rezolvaţi $Ax = b$ cu Cholesky pentru matrice rare şi cu metoda gradientului. Comparaţi timpul, memoria, reziduul şi efectul unei toleranţe diferite. Identificaţi regimul în care metoda directă este convenabilă şi regimul în care o metodă iterativă devine atractivă.

Exemplu rezolvat: un sistem SPD văzut ca minimizare

Considerăm

$$A = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Sistemul $Ax = b$ are soluţia $x^* = (1/11, 7/11)^T$. Asociem funcţia

$$f(x) = \frac{1}{2} x^T A x - b^T x.$$

Gradientul este $\nabla f(x) = Ax - b$, deci punctele staţionare sunt exact soluţiile sistemului. Deoarece A este SPD, Hessiana este $A > 0$, iar funcţia este strict convexă; prin urmare punctul staţionar este unicul minim global.

Pornim din $x_0 = 0$. Gradientul este $g_0 = -b$. Direcţia celei mai mari descreşteri este $-g_0 = b$. Pentru o funcţie pătratică, pasul exact pe această direcţie este

$$\alpha_0 = \frac{b^T b}{b^T A b} = \frac{5}{20} = \frac{1}{4}.$$

Astfel $x_1 = (1/4, 1/2)^T$. Punctul nu este încă soluţia, dar funcţia a scăzut optim de-a lungul dreptei $x_0 + \alpha b$.

Geometrie

Nivelurile lui f sunt elipse centrate în x^*. Valorile proprii ale lui A determină axele şi excentricitatea acestor elipse. Când $\kappa_2(A)$ este mare, valea este îngustă şi metoda gradientului poate zigzaga între laturile ei. Astfel condiţionarea sistemului linear reapare ca geometrie a problemei de optimizare.

Sarcină practică: traiectoria pe curbe de nivel

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru o matrice SPD 2×2, reprezentaţi curbele de nivel ale lui f şi suprapuneţi punctele generate de metoda gradientului cu pas exact. Repetaţi pentru $A = \mathrm{diag}(1, 2)$ şi $A = \mathrm{diag}(1, 100)$. Folosiţi aceeaşi eroare iniţială şi comparaţi numărul de iteraţii necesar pentru aceeaşi toleranţă.

Exerciţii suplimentare

1. Verificaţi soluţia exactă a sistemului şi valoarea lui $f(x^*)$.
2. Derivaţi formula pasului exact minimizând funcţia unidimensională $\phi(\alpha) = f(x_k - \alpha g_k)$.
3. Arătaţi că pentru $A = I$ metoda cu pas exact ajunge la soluţie într-o singură iteraţie.
4. Explicaţi ce se pierde dacă A nu este simetrică atunci când încercăm să folosim aceeaşi funcţie pătratică.
5. Construiţi o matrice SPD cu o condiţionare prescrisă şi observaţi forma elipselor de nivel.

9.1 Metoda direcţiilor de descreştere

Direcţii de descreştere

Considerăm problema de minim

$$\min_{x \in \mathbb{R}^n} f(x), \quad \text{unde} \quad f : \mathbb{R}^n \to \mathbb{R} \ \text{ este o funcţie de clasă } C^1 \ \text{ pe } \ \mathbb{R}^n.$$

În loc să încercăm să găsim expresia analitică a unui punct staţionar, vom construi un algoritm iterativ pentru găsirea punctelor staţionare.

Algoritmii iterativi pe care îi vom lua în considerare în această secţiune au forma $x_{k+1} = x_k + t_k\, d_k, \quad k = 0, 1, 2, ...$, unde d_k este aşa-numita direcţie, iar t_k este mărimea pasului.

Definiţie [Direcţia de descreştere]

Fie $f : \mathbb{R}^n \to \mathbb{R}$ de clasă C^1 pe $\mathbb{R}^n$. Un vector $0 \neq d \in \mathbb{R}^n$ se numeşte direcţie de descreştere a lui f în x dacă derivata direcţională $f'(x; d)$ este negativă, ceea ce înseamnă că

$$f'(x; d) = \Big\langle \nabla f(x), d \Big\rangle < 0.$$

Lemă [Proprietatea locală a direcţiilor de descreştere]

Fie $f : \mathbb{R}^n \to \mathbb{R}$ o funcţie de clasă C^1 pe $\mathbb{R}^n$. Să presupunem că d este o direcţie de descreştere a lui f în x. Atunci există $\varepsilon > 0$ astfel încât

$$f(x + t\, d) < f(x) \qquad \text{pentru orice} \qquad t \in (0, \varepsilon].$$

Demonstraţie. Deoarece $f'(x; d) < 0$, din definiţia derivatei direcţionale rezultă că

$$\lim_{t \to 0^+} \frac{f(x + t\, d) - f(x)}{t} = f'(x; d) < 0.$$

Prin urmare, există $\varepsilon > 0$ astfel încât

$$\frac{f(x + t\, d) - f(x)}{t} < 0 \qquad \text{pentru orice} \qquad t \in (0, \varepsilon],$$

ceea ce implică cu uşurinţă rezultatul dorit. □

Schema metodei direcţiilor de descreştere

Algoritmul metodei direcţiilor de descreştere

- **Iniţializare:** Alegem $x_0 \in \mathbb{R}^n$ arbitrar;
- **Etapa generală:** Pentru orice $k = 0, 1, 2, ...$, se
  - alege o direcţie de descreştere d_k;
  - găseşte o mărime a pasului t_k care să satisfacă $f(x_k + t_k d_k) < f(x_k)$;
  - Se pune $x_{k+1} = x_k + t_k d_k$;
  - se verifică dacă un criteriu de oprire este satisfăcut, atunci STOP şi x_{k+1} este ieşirea.

Atât de frumos şi atât de neclar (doar conceptual)!

Multe detalii lipsesc din descrierea schematică de mai sus a algoritmului :

- Care este punctul de plecare?
- Cum se alege direcţia de descreştere?
- Ce mărime a pasului trebuie să fie luată?
- Care este criteriul de oprire?

Clarificări

- Punctul de pornire poate fi ales arbitrar, în absenţa unei informaţii deja ştiute despre soluţia optimă.

- Un exemplu de criteriu de oprire popular este $\|\nabla f(x_{k+1})\| \leq \varepsilon$.
- Principala diferenţă între diferitele metode este alegerea direcţiei de descreştere .
- Vom presupune că mărimea pasului este aleasă astfel încât $f(x_{k+1}) < f(x_k)$. Încă nu este clar!
  - mărime constantă a pasului: $t_k = \bar{t}, k = 0, 1, 2, ...$; Util pentru probleme simple.
    - O constantă mare poate face ca algoritmul să nu fie descrescător;
    - O constantă mică poate cauza o convergenţă lentă a metodei.
  - căutarea exactă pe linie: t_k este un minimizator al lui f de-a lungul razei $x_k + t\, d_k$, adică $t_k = \operatorname{argmin}_{t \geq 0} f(x_k + t\, d_k)$.
    - Pare mai atractivă la prima vedere;
    - Dar, nu este întotdeauna posibil să găsim minimizatorul exact.
  - procedura de reducere succesivă a pasului (backtracking) este un compromis între ultimele două abordări;

Backtracking

Metoda necesită trei parametri: $s > 0$, $\alpha \in (0, 1)$ $\beta \in (0, 1)$. Alegerea lui t_k se face prin următoarea procedură. În primul rând, t_k se stabileşte ca fiind egal cu presupunerea iniţială s. Apoi, atâta timp cât

$$f(x_k) - f(x_k + t_k d_k) < -\alpha\, t_k \left\langle \nabla f(x_k), d_k \right\rangle,$$

se stabileşte $t_k = \beta\, t_k$.

Mărimea pasului se alege sub forma $t_k = s\, \beta^{i_k}$, unde i_k este cel mai mic număr întreg nenegativ pentru care condiţia

$$f(x_k) - f(x_k + s\, \beta^{i_k} d_k) \geq -\alpha\, s\, \beta^{i_k} \left\langle \nabla f(x_k), d_k \right\rangle$$

este satisfăcută.

Validitatea condiţiei suficiente de scădere

Fie $f : \mathbb{R}^n \to \mathbb{R}$ o funcţie de clasă C^1 pe $\mathbb{R}^n$ şi fie $x \in \mathbb{R}^n$. Să presupunem că $0 \neq d \in \mathbb{R}^n$ este o direcţie de descreştere a lui f în x şi fie $\alpha \in (0, 1)$. Atunci există $\varepsilon > 0$ astfel încât

$$f(x) - f(x + t\, d) \geq -\alpha\, t \left\langle \nabla f(x), d \right\rangle \quad \text{pentru orice} \qquad t \in (0, \varepsilon].$$

Demonstraţie. Deoarece f de clasă C^1 pe $\mathbb{R}^n$ rezultă că

$$f(x + t\, d) = f(x) + t \left\langle \nabla f(x), d \right\rangle + o(t\|d\|),$$

şi, prin urmare

$$f(x) - f(x + t\, d) = -\alpha\, t \left\langle \nabla f(x), d \right\rangle - (1 - \alpha)\, t \left\langle \nabla f(x), d \right\rangle - o(t\|d\|),$$

Deoarece d este o direcţie de descreştere a lui f la x avem

$$\lim_{t \to 0^+} \frac{(1 - \alpha)\, t \left\langle \nabla f(x), d \right\rangle + o(t\|d\|)}{t} = (1 - \alpha) \left\langle \nabla f(x), d \right\rangle < 0.$$

Prin urmare, există $\varepsilon > 0$ astfel încât pentru toate $t \in (0, \varepsilon]$ să existe inegalitatea $(1 - \alpha)\, t \left\langle \nabla f(x), d \right\rangle + o(t\|d\|) < 0$, ceea ce conduce la rezultatul dorit.

□

Completare didactică: direcţia nu este suficientă, trebuie ales şi pasul

Pentru o funcţie diferenţiabilă, un vector d este direcţie de descreştere în x dacă $\nabla f(x)^T d < 0$. Această condiţie este locală: garantează că pentru paşi suficient de mici avem $f(x + td) < f(x)$. Nu spune însă cât de mare poate fi t.

Alegerea $d = -\nabla f(x)$ este întotdeauna de descreştere dacă gradientul este nenul, deoarece $\nabla f(x)^T d = -\|\nabla f(x)\|^2 < 0$. Dar un pas prea mare poate depăşi regiunea de descreştere. De aceea, algoritmul are două componente distincte: alegerea direcţiei şi alegerea lungimii pasului.

Backtracking şi condiţia Armijo

Pornim de la un pas iniţial $t_0 > 0$ şi îl reducem succesiv, de exemplu $t \leftarrow \beta t$ cu $0 < \beta < 1$, până când $f(x + td) \le f(x) + \alpha t \nabla f(x)^T d$, unde $0 < \alpha < 1$. Partea dreaptă cere o descreştere proporţională cu panta locală. Pentru o direcţie de descreştere, un pas suficient de mic va satisface condiţia sub ipoteze uzuale de regularitate.

Laborator: backtracking de la zero

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi o funcţie care primeşte f, gradientul, x, direcţia d, t_0, α şi β. Returnaţi pasul acceptat şi numărul de reduceri. Testaţi pe funcţia Rosenbrock şi pe funcţii pătratice. Reprezentaţi numărul de backtracking-uri pe parcursul iteraţiilor.

Exerciţii şi verificarea înţelegerii

1. Folosind definiţia derivatei direcţionale, justificaţi existenţa unui interval de paşi mici care produc descreştere când $\nabla f(x)^T d < 0$.
2. Pentru $f(x) = x^2$ şi $x = 1$, analizaţi pasul în direcţia $d = -2$ şi determinaţi exact intervalul de $t > 0$ pentru care funcţia scade.
3. De ce condiţia $d = -\nabla f$ nu impune o normalizare a direcţiei? Cum se reflectă scalarea lui d în alegerea lui t?
4. Explicaţi rolurile separate ale parametrilor α şi β în backtracking.
5. Daţi un exemplu în care pasul fix $t = 1$ funcţionează prost, dar backtracking-ul găseşte rapid un pas acceptabil.

Mini-proiect: hartă a paşilor

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Pentru o funcţie convexă în două variabile, rulaţi metoda direcţiilor de descreştere din mai multe puncte iniţiale şi salvaţi pasul acceptat la fiecare iteraţie. Vizualizaţi punctele şi valorile paşilor. Încercaţi să explicaţi zonele în care backtracking-ul reduce frecvent pasul prin curbura funcţiei.

Exemplu rezolvat: de ce un pas prea mare poate creşte funcţia

Pentru $f(x) = x^2$, în punctul $x = 1$ avem $\nabla f(1) = 2$ şi direcţia de gradient negativ este $d = -2$. Pe dreapta $x(t) = 1 - 2t$,

$$f(x(t)) = (1 - 2t)^2.$$

Condiţia $f(x(t)) < f(1) = 1$ este echivalentă cu $|1 - 2t| < 1$, deci $0 < t < 1$. Orice pas suficient de mic produce descreştere, dar pasul $t = 2$, de exemplu, duce în $x = -3$ şi creşte funcţia la 9. Direcţia corectă nu este suficientă; trebuie controlată şi lungimea pasului.

Pentru condiţia Armijo, $f(x + td) \le f(x) + \alpha t \nabla f(x)^T d$, partea dreaptă nu cere doar o scădere oarecare, ci una proporţională cu panta locală. Parametrul α este de obicei mic, iar factorul β controlează cât de agresiv reducem pasul în backtracking.

Sarcină de implementare: backtracking reutilizabil

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi o funcţie `backtracking(f,grad,x,d,t0,alpha,beta)` care verifică mai întâi dacă $g^T d < 0$. Dacă nu, returnaţi un mesaj de eroare. Înregistraţi numărul de reduceri şi valoarea funcţiei la fiecare încercare. Testaţi pe o funcţie pătratică şi pe Rosenbrock. Investigaţi efectul alegerilor $\beta = 0.1, 0.5, 0.9$ asupra numărului de evaluări.

Exerciţii suplimentare

1. Pentru exemplul scalar determinaţi ce paşi satisfac Armijo pentru $\alpha = 10^{-4}$.
2. Demonstraţi din derivata direcţională că dacă $\nabla f(x)^T d < 0$, atunci există paşi pozitivi suficient de mici cu descreştere.
3. Ce se întâmplă dacă multiplicăm direcţia cu 100? Cum trebuie reinterpretat pasul?
4. Explicaţi de ce backtracking nu cere rezolvarea exactă a unei probleme unidimensionale.
5. Implementaţi şi comparaţi un pas fix, pas exact pentru o funcţie pătratică şi backtracking.

9.2 Metoda gradientului

Metoda gradientului

În metoda gradientului, direcţia de descreştere este aleasă ca fiind

$$d_k = -\nabla f(x_k), \quad k = 0, 1, 2... .$$

Aceasta este o alegere bună, deoarece

$$f'(x_k; -\nabla f(x_k)) = -\left\langle \nabla f(x_k), \nabla f(x_k) \right\rangle = -\|\nabla f(x_k)\|^2 < 0.$$

Derivata direcţională minimă între toate direcţiile normalizate

Fie $f : \mathbb{R}^n \to \mathbb{R}$ o funcţie de clasă C^1 pe $\mathbb{R}^n$ şi fie $x \in \mathbb{R}^n$ un punct nestaţionar. Atunci o soluţie optimă a

$$\min_{d \in \mathbb{R}^n} \{ f'(x; d) \ : \ \|d\| = 1 \}$$

este $d = -\dfrac{\nabla f(x)}{\|\nabla f(x)\|}$.

Demonstraţie. Din moment ce $f'(x; d) = \left\langle \nabla f(x), d \right\rangle$, problema este aceeaşi ca în cazul în care

$$\min_{d \in \mathbb{R}^n} \{ \left\langle \nabla f(x), d \right\rangle \ : \ \|d\| = 1 \}$$

Din inegalitatea Cauchy-Schwarz avem

$$\left\langle \nabla f(x), d \right\rangle \geq -\|\nabla f(x)\| \, \|d\| = -\|\nabla f(x)\|.$$

Astfel, $-\|\nabla f(x)\|$ este o limită inferioară a valorii optime a problemei de minimizare. Pe de altă parte, introducând $d = -\dfrac{\nabla f(x)}{\|\nabla f(x)\|}$ în funcţia obiectiv obţinem că

$$f'(x; -\frac{\nabla f(x)}{\|\nabla f(x)\|}) = \left\langle \nabla f(x), -\frac{\nabla f(x)}{\|\nabla f(x)\|} \right\rangle = -\|\nabla f(x)\|,$$

şi astfel ajungem la concluzia că limita inferioară $-\|\nabla f(x)\|$ se atinge la $d = -\dfrac{\nabla f(x)}{\|\nabla f(x)\|}$, ceea ce implică rezultatul dorit. □

Metoda gradientului

- **Date de intrare**: $\varepsilon > 0$, toleranţa prescrisă.
- **Iniţializare**: Se alege $x_0 \in \mathbb{R}^n$ în mod arbitrar.
- **Etapa generală**: Pentru orice $k = 0, 1, 2, ...$ se execută următorii paşi:
  - Se alege o mărime a pasului t_k printr-una dintre procedurile menţionate mai sus pentru
    $$g(t) = f(x_k - t\,\nabla f(x_k)).$$
  - Se pune $x_{k+1} = x_k - t_k \nabla f(x_k)$.
  - Dacă $\|\nabla f(x_{k+1})\| \leq \varepsilon$, se opreşte algoritmul, iar x_{k+1} este aproximaţia returnată

Metoda gradientului se poate comporta destul de rău. Ca un exemplu, să considerăm problema de minimizare

$$\min_{x,y \in \mathbb{R}^n} x^2 + \frac{1}{100}\, y^2,$$

şi să presupunem că utilizăm metoda gradientului cu vectorul iniţial $(\frac{1}{100}, 1)^T$.

Pentru această problemă, un răspuns parţial poate fi obţinut folosind noţiunea de număr de condiţionare.

Completare didactică: metoda gradientului şi pasul optim pentru funcţii pătratice

Metoda gradientului are forma $x^{(k+1)} = x^{(k)} - t_k \nabla f(x^{(k)})$. Pentru funcţia pătratică asociată unui sistem SPD, $\nabla f(x) = Ax - b = -r$, astfel încât actualizarea poate fi scrisă $x^{(k+1)} = x^{(k)} + t_k r^{(k)}$.

Dacă alegem la fiecare iteraţie pasul care minimizează exact funcţia pe dreapta $x^{(k)} + tr^{(k)}$, obţinem

$$t_k = \frac{(r^{(k)})^T r^{(k)}}{(r^{(k)})^T A r^{(k)}}.$$

Formula rezultă prin minimizarea unei funcţii pătratice scalare în t. Ea nu necesită valori proprii, dar cere un produs suplimentar $Ar^{(k)}$.

Ortogonalitatea gradientelor succesive

Cu pas exact pe direcţia gradientului negativ, derivata funcţiei în direcţia liniei este zero în noul punct. Rezultă $(r^{(k+1)})^T r^{(k)} = 0$. Această ortogonalitate explică în parte zigzagul: noua direcţie este perpendiculară pe precedenta în norma euclidiană, nu conjugată faţă de A.

Laborator: pas fix versus pas exact

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi metoda pentru funcţii pătratice SPD în două versiuni: pas fix şi pas exact. Pentru pas fix testaţi valori sub şi peste pragul de stabilitate sugerat de spectrul lui A. Pentru pas exact, verificaţi numeric ortogonalitatea reziduurilor succesive. Reprezentaţi $\|r_k\|$ şi $f(x_k) - f(x_\star)$.

Exerciţii şi verificarea înţelegerii

1. Derivaţi formula pasului exact prin diferenţierea lui $\phi(t) = f(x + tr)$.
2. Demonstraţi ortogonalitatea $r_{k+1}^T r_k = 0$ pentru pasul exact.
3. Pentru $A = I$, arătaţi că metoda cu pas exact ajunge la soluţie într-un singur pas.
4. Pentru $A = \mathrm{diag}(1, 100)$, explicaţi de ce direcţiile succesive pot oscila între axele elipsei.
5. Comparaţi informaţia necesară pentru gradient descent cu cea necesară pentru Cholesky. Care metodă poate lucra doar cu o funcţie care calculează Av?

Mini-proiect: implementare fără construirea explicită a matricei (fără formarea explicită a matricei)

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite

până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Alegeţi o discretizare 2D pentru care puteţi calcula acţiunea Av direct din vecinii fiecărui nod, fără a forma matricea. Implementaţi metoda gradientului în această formă fără construirea explicită a matricei (fără formarea explicită a matricei). Comparaţi memoria cu versiunea care construieşte explicit matricea rară (sparse) şi verificaţi că rezultatele coincid până la toleranţa numerică.

Calcul suplimentar: ortogonalitatea gradientelor succesive

Pentru funcţia pătratică $f(x) = \frac{1}{2}x^TAx - b^Tx$ cu A SPD, metoda gradientului cu pas exact foloseşte $x_{k+1} = x_k - \alpha_k g_k$, $g_k = Ax_k - b$. Pasul exact este ales astfel încât derivata lui $\phi(\alpha) = f(x_k - \alpha g_k)$ să fie zero în α_k. Dar

$$\phi'(\alpha) = -g_k^T\Big(A(x_k - \alpha g_k) - b\Big) = -g_k^T g(\alpha).$$

În punctul minim de pe dreaptă, $g_k^T g_{k+1} = 0$. Aşadar gradientele consecutive sunt ortogonale în produsul scalar euclidian. Geometric, metoda atinge o curbă de nivel într-un punct în care noul gradient este perpendicular pe direcţia parcursă.

Această proprietate nu implică terminare rapidă. Într-o vale foarte alungită, ortogonalitatea poate produce tocmai traiectoria în zigzag caracteristică. Direcţiile se schimbă mult, dar progresul de-a lungul axei lente poate fi modest.

Sarcină practică: verificaţi teorema în virgulă mobilă

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru matrici SPD de dimensiuni 20 şi condiţionări diferite, rulaţi metoda cu pas exact şi salvaţi $g_k^T g_{k+1}/(\|g_k\|\|g_{k+1}\|)$. În aritmetică exactă valoarea este zero; numeric va fi mică, dar nu exact zero. Observaţi cum se comportă când condiţionarea creşte şi iteraţiile devin numeroase.

Exerciţii suplimentare

1. Derivaţi formula $\alpha_k = (g_k^T g_k)/(g_k^T A g_k)$.
2. Calculaţi două iteraţii pentru $A = \begin{pmatrix} 4 & 1 \\ 1 & 3 \end{pmatrix}$, $b = (1, 2)^T$, $x_0 = 0$.
3. Verificaţi direct ortogonalitatea lui g_0 şi g_1 în exemplul precedent.
4. De ce proprietatea nu rămâne exactă dacă folosim un pas aproximativ ales prin Armijo?
5. Comparaţi direcţiile metodei gradientului cu cele ale gradientului conjugat; ce informaţie suplimentară foloseşte a doua metodă?

9.3 Numărul de condiţionare pentru funcţii pătratice. Rescalarea problemei.

Se consideră problema de minimizare pătratică

$$\min_{x\in\mathbb{R}^n}\{f(x) := \Big\langle A\,x, x\Big\rangle\}, \quad \text{unde} \quad A \succ 0.$$

Soluţia optimă este, evident, $x^* = 0$. Metoda gradientului cu pas exact are forma

$$x_{k+1} = x_k + t_k d_k,$$

unde $d_k = -\nabla f(x_k) = -2\,A\,x_k$ este direcţia antigradient în x_k, iar pasul t_k este găsit ca fiind

$$t_k = \frac{\|d_k\|^2}{2\Big\langle A\,d_k, d_k\Big\rangle}.$$

Deci,

$$\begin{aligned} f(x_{k+1}) &= \Big\langle A\,x_{k+1}, x_{k+1}\Big\rangle = \Big\langle A\,(x_k + t_k d_k), (x_k + t_k d_k)\Big\rangle \\ &= \Big\langle A\,x_k, x_k\Big\rangle + 2\,t_k\Big\langle A\,x_k, d_k\Big\rangle + t_k^2\Big\langle A\,d_k, d_k\Big\rangle \\ &= \Big\langle A\,x_k, x_k\Big\rangle - t_k\Big\langle d_k, d_k\Big\rangle + t_k^2\Big\langle A\,d_k, d_k\Big\rangle. \end{aligned}$$

Introducând în ultima relaţie expresia pentru t_k dată mai sus, obţinem că

$$\begin{aligned} f(x_{k+1}) &= \Big\langle A\,x_k, x_k\Big\rangle - \frac{1}{4}\frac{\Big\langle d_k, d_k\Big\rangle^2}{\Big\langle A\,d_k, d_k\Big\rangle} = \Big\langle A\,x_k, x_k\Big\rangle\left(1 - \frac{1}{4}\frac{\Big\langle d_k, d_k\Big\rangle^2}{\Big\langle A\,d_k, d_k\Big\rangle\Big\langle A\,x_k, x_k\Big\rangle}\right) \\ &= f(x_k)\left(1 - \frac{\Big\langle d_k, d_k\Big\rangle^2}{\Big\langle A\,d_k, d_k\Big\rangle\Big\langle A^{-1}\,d_k, d_k\Big\rangle}\right) \end{aligned}$$

Inegalitatea lui Kantorovici

Fie A o matrice $n \times n$ pozitiv definită. Atunci pentru orice $0 \neq x \in \mathbb{R}^n$ are loc inegalitatea

$$\frac{\Big\langle x, x\Big\rangle^2}{\Big\langle A\,x, x\Big\rangle\Big\langle A^{-1}\,x, x\Big\rangle} \geq \frac{4\,\lambda_{\max}\lambda_{\min}}{(\lambda_{\max} + \lambda_{\min})^2}.$$

Se notează $m = \lambda_{\min}$ şi $M = \lambda_{\max}$. Valorile proprii ale matricei $A + M\,m\,A^{-1}$ sunt $\lambda_i + \frac{M\,m}{\lambda_i}$, $i = 1, ..., n$. Este uşor de demonstrat că maximul funcţiei unidimensionale $\varphi(t) = t + \frac{M\,m}{t}$ pe $[m, M]$ este atins în punctele m şi M cu o valoare corespunzătoare a funcţiei φ de $M + m$ şi, prin urmare, din moment ce $m \leq \lambda_i(A) \leq M$, rezultă că valorile proprii ale lui $A + M\,m\,A^{-1}$ sunt mai mici sau egale cu $(M + m)$. Astfel,

$$A + M\,m\,A^{-1} \preceq (M + m)I_n \;\text{ în sensul că } A + M\,m\,A^{-1} - (M + m)I_n \preceq 0$$

Adică:

$$\Big\langle A\,x, x\Big\rangle + M\,m\Big\langle A^{-1}\,x, x\Big\rangle \leq (M + m)\Big\langle x, x\Big\rangle,$$

care, combinată cu inegalitatea simplă $\alpha\,\beta \le \frac{1}{4}(\alpha+\beta)^2\ \forall\,\alpha,\beta\in\mathbb{R}$ rezultă

$$\begin{aligned}\left\langle A\,x,x\right\rangle M\,m\left\langle A^{-1}\,x,x\right\rangle &\le \frac{1}{4}[\left\langle A\,x,x\right\rangle + M\,m\left\langle A^{-1}\,x,x\right\rangle]^2\\ &\le \frac{(M+m)^2}{4}\left\langle x,x\right\rangle^2,\end{aligned}$$

care, după o simplă rearanjare a termenilor, conduce la rezultatul dorit.

Revenind la analiza ratei de convergenţă a metodei gradientului, rezultă, folosind inegalitatea lui Kantorovici, că

$$f(x_{k+1}) \le \left(1-\frac{4\,m\,M}{(M+m)^2}\right) f(x_k) = \left(\frac{M-m}{M+m}\right)^2 f(x_k),$$

unde $M=\lambda_{\max}(A)$, $m=\lambda_{\min}(A)$.

Rezumând, avem

Analiza ratei de convergenţă pentru funcţii pătratice

Fie x_k şirul generat de metoda gradientului cu pas exact pentru rezolvarea problemei de minimizare pătratică

$$\min_{x\in\mathbb{R}^n}\{f(x) := \left\langle A\,x,x\right\rangle\},\quad \text{unde}\quad A\succ 0.$$

Atunci, pentru orice $k=0,1,...$

$$f(x_{k+1}) \le \left(\frac{M-m}{M+m}\right)^2 f(x_k),$$

unde $M=\lambda_{\max}(A)$, $m=\lambda_{\min}(A)$.

Prin iteraţie, pentru orice $k=0,1,...,$

$$f(x_k) \le c^k\,f(x_0),\quad \text{unde } c=\left(\frac{M-m}{M+m}\right)^2 = \left(\frac{\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}-1}{\frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}+1}\right)^2.$$

Numărul $\kappa = \dfrac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ se numeşte *număr de condiţionare* al lui A.

Matricele cu un număr mare de condiţioare se numesc *rău condiţionate*, iar matricele cu un număr mic de condiţionare (aproape de 1) se numesc *bine condiţionate*.

Completare didactică: condiţionarea funcţiei pătratice şi viteza de convergenţă

Pentru $f(x)=\frac{1}{2}x^TAx-b^Tx$, cu $A\succ 0$, numărul de condiţionare spectral $\kappa_2(A)=\lambda_{\max}/\lambda_{\min}$ măsoară raportul dintre curburile extreme. Când κ este aproape de 1, curbele de nivel sunt aproape circulare şi gradientul indică eficient către minim. Când κ este mare, curbele de nivel sunt foarte alungite şi metoda poate progresa lent de-a lungul direcţiei cu curbură mică.

Pentru un pas fix t, eroarea satisface $e^{(k+1)}=(I-tA)e^{(k)}$. Convergenţa pentru orice eroare iniţială cere $|1-t\lambda_i|<1$ pentru toate valorile proprii, ceea ce este asigurat de $0<t<2/\lambda_{\max}$. Alegerea unui pas fără informaţie despre scala spectrului poate deci conduce la divergenţă.

O alegere echilibrată pentru o problemă spectral cunoscută este $t = 2/(\lambda_{\min} + \lambda_{\max})$, care minimizează cel mai mare dintre modulele $|1 - t\lambda_i|$ pe intervalul spectral. Factorul rezultat este $(\kappa - 1)/(\kappa + 1)$. Chiar şi această alegere devine lentă când κ este mare.

Laborator: spectru şi rata observată

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru matrici diagonale $A = \operatorname{diag}(1, \kappa)$ cu $\kappa = 2, 10, 100, 1000$, aplicaţi gradient descent cu pasul $2/(1 + \kappa)$. Estimaţi din date factorul asimptotic de reducere a erorii şi comparaţi cu $(\kappa - 1)/(\kappa + 1)$. Reprezentaţi traiectoriile în plan.

Exerciţii şi verificarea înţelegerii

1. Derivaţi condiţia $0 < t < 2/\lambda_{\max}$ pentru pas fix folosind diagonalizarea spectrală.
2. Determinaţi pasul optim constant prin egalarea modulelor erorilor la capetele intervalului spectral.
3. Ce se întâmplă cu factorul $(\kappa - 1)/(\kappa + 1)$ când $\kappa \to \infty$?
4. Explicaţi de ce o schimbare de coordonate poate modifica dramatic condiţionarea funcţiei chiar dacă minimizatorul fizic reprezintă aceeaşi soluţie.

Mini-proiect: estimarea valorilor proprii extreme

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Într-o problemă mare nu doriţi să calculaţi întregul spectru. Implementaţi metoda puterii pentru estimarea lui $\lambda_{\max}$ al unei matrice SPD şi o strategie simplă pentru o estimare a lui $\lambda_{\min}$ pe exemple mici. Folosiţi estimările pentru a alege un pas fix şi comparaţi cu un pas ales arbitrar.

Exemplu: condiţionare mare şi zigzag

Pentru $A = \operatorname{diag}(1, 100)$ şi $b = 0$, funcţia este $f(x) = \frac{1}{2}(x_1^2 + 100x_2^2)$. Minimizerul este originea, iar $\kappa_2(A) = 100$. Din $x_0 = (1, 1)^T$, gradientul este $g_0 = (1, 100)^T$. Pasul exact este

$$\alpha_0 = \frac{1 + 10^4}{1 + 10^6} = \frac{10001}{1000001} \approx 0.0100.$$

Noua componentă x_2 devine foarte mică, dar x_1 rămâne aproape de 1. În iteraţia următoare gradientul este dominat de cealaltă direcţie şi traiectoria traversează din nou valea. Cu cât raportul valorilor proprii este mai mare, cu atât elipsele de nivel sunt mai alungite şi această alternanţă poate fi mai severă.

Teoria dă un factor de contracţie legat de $((\kappa-1)/(\kappa+1))^2$ pentru scăderea funcţiei în cazul pasului exact. Pentru κ apropiat de 1 factorul este mic; pentru $\kappa \gg 1$ se apropie de 1, semnalând convergenţă lentă.

Sarcină practică: experiment pe condiţionări controlate

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Rulaţi metoda pe $A = Q^T \operatorname{diag}(1,\kappa) Q$ în dimensiunea 2, pentru $\kappa = 2, 10, 10^2, 10^4$ şi o rotaţie fixă Q. Reprezentaţi traiectoria peste curbele de nivel şi numărul de iteraţii până la o toleranţă dată. Verificaţi dacă rata observată se deteriorează odată cu κ.

Exerciţii suplimentare

1. Calculaţi explicit x_1 pentru exemplul diagonal şi apoi g_1.
2. Ce se întâmplă dacă $A = cI$? Care este condiţionarea şi în câţi paşi converge metoda cu pas exact?
3. Interpretaţi valorile proprii mici şi mari ca curburi ale funcţiei.
4. Arătaţi că o rotaţie ortogonală nu schimbă valorile proprii şi nici numărul de condiţionare spectral.
5. Explicaţi de ce precondiţionarea urmăreşte gruparea valorilor proprii.

9.4 Rescalare

Rescalare

Problema matricelor prost condiţionate este una majoră şi au fost dezvoltate multe metode pentru a o evita. Una dintre cele mai populare abordări constă în “condiţionarea” problemei prin efectuarea unei transformări liniare corespunzătoare a variabilelor.

Mai precis, să considerăm problema de minimizare fără constrângeri

$$\min_{x \in \mathbb{R}^n} f(x).$$

Pentru o matrice nesingulară dată $S \in \mathbb{R}^{n \times n}$, efectuăm transformarea liniară $x = S\,y$ şi obţinem problema echivalentă

$$\min_{y \in \mathbb{R}^n} g(y) := f(S\,y).$$

Deoarece $\nabla g(y) = S^T \nabla f(S\,y) = S^T \nabla f(x)$, rezultă că metoda gradientului aplicată problemei transformate ia forma

$$y_{k+1} = y_k - t_k\, S^T \nabla f(S\,y_k).$$

Din moment ce $\nabla g(y) = S^T \nabla f(S\,y) = S^T \nabla f(x)$, rezultă că metoda gradientului aplicată problemei transformate ia forma

$$y_{k+1} = y_k - t_k\, S^T \nabla f(S\,y_k).$$

Înmulţind această ultimă egalitate cu S la stânga şi folosind notaţia $x_k = S\,y_k$, obţinem formula recursivă

$$x_{k+1} = x_k - t_k\, S\, S^T \nabla f(x_k).$$

Definind $D = S\,S^T$, obţinem următoarea versiune a metodei gradientului, pe care o numim metoda gradientului scalat cu matrice de scalare D (pozitiv definită):

$$x_{k+1} = x_k - t_k\, D \nabla f(x_k).$$

Direcţia $-D\nabla f(x_k)$ este o direcţie de descreştere a lui f în x_k atunci când $\nabla f(x_k) \neq 0$, deoarece

$$f'(x_k; -D\nabla f(x_k)) = -\Big\langle \nabla f(x_k), D\nabla f(x_k) \Big\rangle < 0.$$

Pentru a rezuma discuţia, dacă alegem în particular $S = D^{1/2}$, rădăcina pătrată simetrică pozitiv definită a lui D, atunci $S = S^T$ şi $SS^T = S^T S = D$. Metoda gradientului scalat cu D coincide astfel cu metoda gradientului obişnuit aplicată funcţiei $g(y) = f(D^{1/2}y)$.

Analiza convergenţei metodei gradientului (Opţional)

Vom prezenta o analiză a convergenţei metodei gradientului utilizată pentru problema de minimizare fără restricţii.

$$\min_{x\in\mathbb{R}^n} f(x).$$

Vom presupune că funcţia obiectiv f este de clasă C^1 şi că gradientul său ∇f este Lipschitz continuu pe $\mathbb{R}^n$, ceea ce înseamnă că

$$\|\nabla f(x) - \nabla f(y)\| \leq L\,\|x - y\| \quad \text{pentru orice} \quad x, y \in \mathbb{R}^n.$$

Reţineţi că dacă ∇f este Lipschitz cu constanta L, atunci este de asemenea Lipschitz cu constanta $\widetilde{L} \geq L$. Prin urmare, există un număr infinit de constante Lipschitz pentru o funcţie cu gradient Lipschitz.

Clasa funcţiilor cu gradient Lipschitz cu constanta L este notată cu $C_L^{1,1}(\mathbb{R}^n)$ sau pur şi simplu $C_L^{1,1}$.

- Funcţii liniare: $f(x) = \Big\langle a, x \Big\rangle$ cu $L = 0$.
- Funcţii pătratice: dacă $A = A^T$, atunci pentru $f(x) = \Big\langle A\,x, x \Big\rangle + 2\Big\langle b, x \Big\rangle + c$ putem lua $L = 2\,\|A\|$, deoarece

  $$\nabla f(x) = 2Ax + 2b, \qquad \|\nabla f(x) - \nabla f(y)\| \leq 2\,\|A\|\,\|x - y\|.$$

  Pentru o matrice generală, termenul pătratic depinde numai de $\operatorname{sym} A$, iar în estimare apare $\|A + A^T\|$.

Propoziţie Fie f o funcţie de clasă C^2 pe $\mathbb{R}^n$. Atunci următoarele două afirmaţii sunt echivalente
(a) $f \in C_L^{1,1}(\mathbb{R}^n)$.
(b) $\|\nabla^2 f(x)\| \leq L$ pentru orice $x \in \mathbb{R}^n$, unde $\|\cdot\|$ reprezintă norma spectrală.

(b)$\rightarrow$ (a). Să presupunem că $\|\nabla^2 f(x)\| \leq L$ pentru orice $x \in \mathbb{R}^n$. Atunci, prin teorema fundamentală a calculului integral, avem pentru orice $x, y \in \mathbb{R}^n$

$$\nabla f(y) = \nabla f(x) + \int_0^1 \nabla^2 f(x + t(y - x))\,(y - x)dt$$

$$= \nabla f(x) + \left(\int_0^1 \nabla^2 f(x + t(y-x))dt \right) (y-x),$$

Deci

$$\begin{aligned} \|\nabla f(y) - \nabla f(x)\| =& \| \left(\int_0^1 \nabla^2 f(x + t(y-x))dt \right) (y-x) \| \\ \leq & \| \int_0^1 \nabla^2 f(x + t(y-x))dt \| \, \|y-x\| \\ \leq & \int_0^1 \|\nabla^2 f(x + t(y-x))\| dt \, \|y-x\| \\ \leq & L \, \|y-x\|, \end{aligned}$$

care demonstrează rezultatul dorit, adică $f \in C_L^{1,1}$.

(a)→ (b). Să presupunem acum că $f \in C_L^{1,1}$. Atunci, prin teorema fundamentală a calculului integral, avem pentru toţi $d \in \mathbb{R}^n$ şi $\alpha > 0$

$$\nabla f(x + \alpha d) = \nabla f(x) + \int_0^\alpha \nabla^2 f(x + t\, d)\, d\, dt$$

Astfel,

$$\| \int_0^\alpha \nabla^2 f(x + t\, d)\, dt\, d\| = \|\nabla f(x + \alpha d) - f(x)\| \leq \alpha\, L\, \|d\|.$$

Împărţind cu α şi luând $\alpha \to 0^+$, obţinem

$$\|\nabla^2 f(x)\, d\| \leq L \|d\|,$$

ceea ce implică faptul că $\|\nabla^2 f(x)\,\| \leq L$.

Lema descreşterii

Pentru funcţiile $C_L^{1,1}$ vom folosi faptul că ele pot fi mărginite superior de o funcţie pătratică pe întregul spaţiu.

Propoziţie [Lema descreşterii]

Fie $f \in C_L^{1,1}(\mathbb{R}^n)$. Atunci, pentru orice $x, y \in \mathbb{R}^n$

$$f(y) \leq f(x) + \left\langle \nabla f(x), y - x \right\rangle + \frac{L}{2} \|x - y\|^2.$$

Din teorema fundamentală a calculului integral avem

$$f(y) - f(x) = \int_0^1 \left\langle \nabla f(x + t(y-x)), y - x \right\rangle dt.$$

Prin urmare,

$$f(y) - f(x) = \left\langle \nabla f(x), y - x \right\rangle + \int_0^1 \left\langle \nabla f(x + t(y-x)) - \nabla f(x), y - x \right\rangle dt.$$

Deci,

$$\left| f(y) - f(x) - \left\langle \nabla f(x), y - x \right\rangle \right| = \left| \int_0^1 \left\langle \nabla f(x + t(y-x)) - \nabla f(x), y - x \right\rangle dt \right|$$

$$\begin{aligned}
&\leq \int_0^1 |\left\langle \nabla f(x + t(y-x)) - \nabla f(x), y - x \right\rangle| \, dt \\
&\leq \int_0^1 \|\nabla f(x + t(y-x)) - \nabla f(x)\| \, \|y - x\| \, dt \\
&\leq \int_0^1 t \, L \, \|y - x\|^2 \, dt = \frac{L}{2} \|y - x\|^2.
\end{aligned}$$

Reţineţi că demonstraţia lemei descreşterii arată de fapt atât limitele superioare, cât şi cele inferioare ale funcţiei:

$$f(x) + \left\langle \nabla f(x), y - x \right\rangle - \frac{L}{2}\|y - x\|^2 \leq f(y) \leq f(x) + \left\langle \nabla f(x), y - x \right\rangle + \frac{L}{2}\|y - x\|^2.$$

Lema scăderii suficiente

Lema scăderii suficiente Să presupunem că $f \in C_L^{1,1}(\mathbb{R}^n)$. Atunci, pentru orice $x \in \mathbb{R}^n$ şi $t > 0$

$$f(x) - f(x - t\,\nabla f(x)) \geq t \left(1 - \frac{L\,t}{2}\right) \|\nabla f(x)\|^2.$$

Demonstraţie. Prin lema descreşterii avem

$$\begin{aligned}
f(x - t\,\nabla f(x)) &\leq f(x) - t\,\|\nabla f(x)\|^2 + \frac{L\,t^2}{2}\|\nabla f(x)\|^2 \\
&= f(x) - t\left(1 - \frac{L\,t}{2}\right)\|\nabla f(x)\|^2
\end{aligned}$$

□

Scopul nostru este acum să arătăm că există paşi viabili pentru fiecare dintre strategiile de selectare a mărimii paşilor:

- pas constant;
- căutarea exactă pe linie;
- reducere succesivă a pasului (backtracking).

În cazul unui pas constant, presupunem că $t_k = \bar{t} \in (0, \frac{2}{L})$. Înlocuind $x = x_k$, $t = \bar{t}$ în lema de scădere suficientă rezultă inegalitatea

$$f(x_k) - f(x_{k+1}) \geq \bar{t} \left(1 - \frac{L\,\bar{t}}{2}\right) \|\nabla f(x_k)\|^2.$$

Reţineţi că descreşterea funcţiei obiectiv la fiecare iteraţie a metodei gradientului este

$$\bar{t} \left(1 - \frac{L\,\bar{t}}{2}\right) \|\nabla f(x_k)\|^2$$

Dacă dorim să obţinem cea mai mare limită garantată a scăderii, atunci căutăm maximul lui $\bar{t}\left(1 - \frac{L\bar{t}}{2}\right)$ în $(0, \frac{2}{L})$. Acest maxim este atins pentru $\bar{t} = \frac{1}{L}$ şi, prin urmare, o alegere potrivită pentru mărimea pasului este $\frac{1}{L}$.

În acest caz

$$f(x_k) - f(x_{k+1}) \geq \frac{1}{L}\left(1 - \frac{L\frac{1}{L}}{2}\right)\|\nabla f(x_k)\|^2 \geq \frac{1}{2\,L}\|\nabla f(x_k)\|^2.$$

În cadrul pasului exact, formula iterativă a algoritmului este

$$x_{k+1} = x_k - t_k \nabla f(x_k),$$

unde $t_k = \text{argmin}_{t\geq 0} f(x_k - t\,\nabla f(x_k))$.
Prin definiţia lui t_k ştim că

$$f(x_k - t_k\nabla f(x_k)) \leq f(x_k - \frac{1}{L}\nabla f(x_k)),$$

şi astfel avem

$$f(x_k) - f(x_{k+1}) \geq f(x_k) - f(x_k - t_k\nabla f(x_k)) \geq \frac{1}{2\,L}\|\nabla f(x_k)\|^2.$$

Aceeaşi estimare ca şi în cazul mărimii constante a paşilor.
În cazul procedurii de backtracking căutăm pasul t_k suficient de mic astfel încât

$$f(x_k) - f(x_k - \frac{t_k}{\beta}\nabla f(x_k)) < \alpha\frac{t_k}{\beta}\|\nabla f(x_k)\|^2.$$

Înlocuind $x = x_k$, $t = \frac{t_k}{\beta}$ în lema scăderii suficiente obţinem că

$$f(x_k) - f(x_k - \frac{t_k}{\beta}\nabla f(x_k)) \geq \frac{t_k}{\beta}\left(1 - \frac{L\,t_k}{2\,\beta}\right)\|\nabla f(x_k)\|^2$$

care, combinat cu estimarea de mai sus, implică faptul că

$$\frac{t_k}{\beta}\left(1 - \frac{L\,t_k}{2\,\beta}\right)\|\nabla f(x_k)\|^2 < \alpha\frac{t_k}{\beta}\|\nabla f(x_k)\|^2$$

ceea ce este acelaşi lucru cu

$$t_k > \frac{2(1-\alpha)\beta}{L}.$$

În general, obţinem că în cadrul procedurii de backtracking avem

$$t_k \geq \min\{s, \frac{2(1-\alpha)\beta}{L}\},$$

ceea ce implică faptul că

$$f(x_k) - f(x_k - t_k\nabla f(x_k)) \geq \alpha\,\min\{s, \frac{2(1-\alpha)\beta}{L}\}\|\nabla f(x_k)\|^2.$$

Lemma

Fie $f \in C_L^{1,1}(\mathbb{R}^n)$. Fie $\{x_k\}_{k\geq 0}$ şirul generat de metoda gradientului pentru rezolvarea $\min_{x\in\mathbb{R}^n} f(x)$ cu una dintre următoarele strategii găsire a paşilor:

- pas constant $\bar{t} \in (0, \frac{2}{L})$,

- pas exact,
- procedură de backtracking cu parametrii $s \in (0,\infty)$, $\alpha \in (0,1)$ şi $\beta \in (0,1)$.

Atunci

$$f(x_k) - f(x_{k+1}) \geq M\|\nabla f(x_k)\|^2,$$

unde

$$M = \begin{cases} \bar{t}\left(1 - \frac{\bar{t}L}{2}\right) & \text{pas constant,} \\ \frac{1}{2\,L} & \text{pas exact,} \\ \alpha \min\{s, \frac{2(1-\alpha)\beta}{L}\} & \text{backtracking.} \end{cases}$$

Convergenţa metodei gradientului

Propoziţie

Fie $f \in C_L^{1,1}(\mathbb{R}^n)$. Fie $\{x_k\}_{k\geq 0}$ şirul generat de metoda gradientului pentru rezolvarea $\min_{x\in\mathbb{R}^n} f(x)$ cu una dintre următoarele strategii de găsire a paşilor:

- pas constant $\bar{t} \in (0, \frac{2}{L})$,
- pas exact,
- procedură de backtracking cu parametrii $s \in (0,\infty)$, $\alpha \in (0,1)$ şi $\beta \in (0,1)$.

Să presupunem că f este mărginit inferior pe $\mathbb{R}^n$, adică există $m \in \mathbb{R}$ astfel încât $f(x) > m$ pentru orice $x \in \mathbb{R}^n$. Atunci,

a) şirul $\{f(x_k)\}_{k\geq 0}$ este descrescător. În plus, pentru orice $k \geq 0$, $f(x_{k+1}) < f(x_k)$, cu excepţia cazului în care $\nabla f(x_k) = 0$.

b) $\nabla f(x_k) \to 0$ pentru $k \to \infty$.

Demonstraţie. a) Din lema anterioară avem că

$$f(x_k) - f(x_{k+1}) \geq M\|\nabla f(x_k)\|^2 \geq 0,$$

pentru o anumită constantă $M > 0$ şi, prin urmare, egalitatea $f(x_k) = f(x_{k+1})$ poate avea loc numai atunci când $\nabla f(x_k) = 0$.

b) Deoarece şirul $\{f(x_k)\}_{k\geq 0}$ este descrescător şi mărginit inferior, deci converge. Astfel, în particular $f(x_k) - f(x_{k+1}) \to 0$ când $k \to \infty$, ceea ce, combinat cu inegalitatea de mai sus, implică faptul că $\|\nabla f(x_k)\| \to 0$ pe măsură ce $k \to \infty$.

□

Rata de convergenţă a normelor de gradient (Opţional)

Propoziţie

În condiţiile propoziţiei anterioare, fie f^* limita şirului $\{f(x_k)\}_{k\geq 0}$. Atunci, pentru orice $n = 0, 1, 2, \ldots$.

$$\min_{k=0,1,\ldots,n} \|\nabla f(x_k)\| \leq \sqrt{\frac{f(x_0) - f^*}{M(n+1)}},$$

unde

$$M = \begin{cases} \bar{t}\left(1 - \frac{\bar{t}L}{2}\right) & \text{dimensiunea pasului constant,} \\ \frac{1}{2\,L} & \text{pas exact,} \\ \alpha \min\{s, \frac{2(1-\alpha)\beta}{L}\} & \text{backtracking.} \end{cases}$$

Demonstraţie. Prin adunarea inegalităţilor

$$f(x_k) - f(x_{k+1}) \geq M\|\nabla f(x_k)\|^2,$$

pentru $k = 0, 1, ..., n$, obţinem

$$f(x_0) - f(x_{n+1}) \geq M \sum_{k=0}^{n} \|\nabla f(x_k)\|^2,$$

Deoarece $f(x_{n+1}) \geq f^*$, putem astfel concluziona că

$$f(x_0) - f^* \geq M \sum_{k=0}^{n} \|\nabla f(x_k)\|^2.$$

În final, folosind această ultimă inegalitate împreună cu faptul că pentru fiecare $k = 0, 1, ..., n$ avem inegalitatea evidentă $\|\nabla f(x_k)\|^2 \geq \min_{k=0,1,...,n} \|\nabla f(x_k)\|^2$, rezultă că

$$f(x_0) - f^* \geq M(n+1) \min_{k=0,1,...,n} \|\nabla f(x_k)\|^2.$$

□

Completare didactică: rescalarea ca precondiţionare elementară

Dacă variabilele au scări foarte diferite, curbele de nivel pot fi alungite în coordonatele originale chiar şi atunci când problema are o structură simplă. Introducând $x = Sy$, cu S inversabilă, funcţia pătratică devine $\widetilde{f}(y) = \frac{1}{2} y^T (S^T A S) y - (S^T b)^T y$. Alegerea lui S urmăreşte ca matricea transformată $S^T A S$ să aibă un spectru mai grupat.

O alegere diagonală simplă poate normaliza aproximativ scările variabilelor. În termeni de sisteme liniare, această idee este un caz elementar de precondiţionare. În metode avansate, căutăm o matrice $M \approx A$ pentru care aplicarea lui M^{-1} este ieftină şi sistemul precondiţionat are o condiţionare mai bună.

Exemplu

Pentru $A = \mathrm{diag}(1, 10^6)$, alegerea $S = \mathrm{diag}(1, 10^{-3})$ produce $S^T A S = I$. În noile coordonate, o problemă extrem de anisotropă devine isotropă. Desigur, în probleme generale nu cunoaştem o transformare perfectă atât de simplă, dar exemplul arată obiectivul.

Laborator: scalare diagonală

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi matrici SPD de forma $A = DCD$, unde C are condiţionare moderată, iar elementele lui D variază pe mai multe ordine de mărime. Comparaţi metoda gradientului înainte şi după schimbarea de variabilă sugerată de diagonala lui A. Raportaţi condiţionarea şi numărul de iteraţii.

Exerciţii şi verificarea înţelegerii

1. Derivaţi gradientul funcţiei transformate $\widetilde{f}(y) = f(Sy)$.
2. Pentru $A = \mathrm{diag}(a_1, \ldots, a_n)$ cu $a_i > 0$, găsiţi o matrice diagonală S care transformă Hessiana în identitate.
3. Explicaţi de ce o rescalare bună pentru algoritm trebuie însoţită de transformarea corectă înapoi la variabilele originale.
4. Comparaţi scalarea pe linii din eliminarea gaussiană cu schimbarea de variabilă din optimizare. Sunt operaţii conceptual identice?

Mini-proiect: precondiţionare diagonală

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Pentru matricea Poisson 2D perturbată prin coeficienţi variabili, testaţi o precondiţionare Jacobi $M = \mathrm{diag}(A)$. Comparaţi o metodă iterativă înainte şi după precondiţionare. Măsuraţi nu doar numărul de iteraţii, ci şi costul total, deoarece un precondiţionator util trebuie să fie ieftin de aplicat.

Exemplu rezolvat: o schimbare de scară care elimină anisotropia

Pentru funcţia cu $A = \mathrm{diag}(1, 100)$ introducem schimbarea de variabile $x = Dy$, unde $D = \mathrm{diag}(1, 0.1)$. Atunci

$$\frac{1}{2}x^T Ax = \frac{1}{2}y^T (D^T AD)y,$$

iar $D^T AD = \mathrm{diag}(1, 1) = I$. În coordonatele y, curbele de nivel nu mai sunt elipse foarte alungite, ci cercuri. Condiţionarea părţii pătratice a scăzut de la 100 la 1.

În probleme reale nu cunoaştem în general transformarea perfectă. Rescalarea pe diagonale, normalizarea coloanelor sau precondiţionarea cu o aproximaţie a lui A^{-1} urmăresc aceeaşi idee: alegerea unor coordonate în care problema este mai uniformă.

Atenţie la interpretare

Schimbarea de variabile modifică coordonatele, nu problema fizică. După rezolvarea în y, trebuie revenit la $x = Dy$. De asemenea, scalarea poate îmbunătăţi algoritmul fără a elimina complet sensibilitatea intrinsecă faţă de perturbaţiile modelului iniţial.

Sarcină practică

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru o matrice SPD cu elemente diagonale foarte diferite, comparaţi metoda gradientului înainte şi după scalarea $D = \operatorname{diag}(a_{11}^{-1/2}, \dots, a_{nn}^{-1/2})$. Măsuraţi condiţionarea matricei transformate şi numărul de iteraţii. Construiţi şi un exemplu în care această scalare diagonală ajută puţin, arătând că nu toate problemele pot fi rezolvate numai prin normalizarea diagonalei.

Exerciţii suplimentare

1. Verificaţi calculul $D^T A D = I$ din exemplu.
2. Pentru $A = \operatorname{diag}(10^{-4}, 1, 10^{6})$ găsiţi o scalare diagonală care transformă diagonala în unităţi.
3. Formulaţi sistemul transformat atunci când $Ax = b$ şi $x = Dy$.
4. Ce diferenţă conceptuală există între scalarea necunoscutelor şi scalarea ecuaţiilor?
5. Daţi un exemplu de matrice cu diagonală unitară, dar condiţionare mare. Ce arată acesta despre limitele scalării diagonale?

Fişă de seminar: metode de descreştere şi gradient

Problema rezolvată 1: pas fix versus pas exact

Pentru $f(x) = \frac{1}{2} x^T A x$ cu $A = \operatorname{diag}(1, 10)$, metoda gradientului cu pas fix este $x_{k+1} = (I - \alpha A) x_k$. Convergenţa pentru orice punct iniţial cere $|1 - \alpha \lambda_i| < 1$ pentru toate valorile proprii, deci $0 < \alpha < 2/\lambda_{\max} = 0.2$. Dacă alegem $\alpha = 0.1$, componentele se multiplică cu 0.9 şi 0, deci a doua componentă dispare într-un pas, iar prima scade lent. Dacă alegem $\alpha = 0.25$, componenta asociată lui $\lambda = 10$ se multiplică cu -1.5 şi metoda diverge.

Pasul exact se adaptează punctului curent şi evită alegerea manuală a unei valori globale, dar cere produse suplimentare cu A. Backtracking-ul este o alternativă robustă pentru funcţii generale când pasul exact nu este disponibil în formă simplă.

Problema rezolvată 2: precondiţionarea ca schimbare de metrică

În loc de direcţia $-g_k$, putem folosi $-M^{-1} g_k$ cu M SPD. Aceasta este direcţia de gradient în produsul scalar $\langle u, v \rangle_M = u^T M v$. Dacă $M \approx A$, curburile problemei sunt echilibrate şi convergenţa poate fi accelerată. Rescalarea diagonală studiată în capitol este cel mai simplu exemplu de această idee.

Laborator integrator 9: trei strategii de pas

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru aceeaşi funcţie pătratică şi acelaşi punct iniţial, comparaţi: pas fix ales conservativ, pas exact şi backtracking Armijo. Raportaţi iteraţiile, evaluările funcţiei, produsele

matrice–vector şi timpul. Repetaţi pentru condiţionări $10, 10^2, 10^4$. Nu declaraţi o metodă „mai bună” fără a preciza costul măsurat.

Set de probleme

1. Derivaţi intervalul de convergenţă al pasului fix pentru o matrice SPD.
2. Găsiţi valoarea de pas fix care minimizează $\max_i |1 - \alpha\lambda_i|$ când spectrul este în $[m, M]$.
3. De ce direcţia $-g$ este direcţia de scădere maximă numai relativ la norma euclidiană?
4. Arătaţi că $-M^{-1}g$ este direcţie de descreştere dacă M este SPD şi $g \neq 0$.
5. Construiţi un exemplu în care backtracking porneşte cu $t_0 = 1$ şi acceptă imediat aproape de soluţie.
6. De ce un criteriu $\|g_k\| \le \varepsilon$ depinde de scalarea funcţiei?
7. Comparaţi eroarea în energie $\|e\|_A$ cu eroarea euclidiană.
8. Explicaţi legătura dintre condiţionarea lui A şi forma elipselor de nivel.

Proiect de capitol: optimizare vizuală

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi un program MATLAB care permite alegerea valorilor proprii şi a rotaţiei unei funcţii pătratice în două variabile. Afişaţi curbele de nivel şi traseele pentru pas fix, pas exact şi gradient precondiţionat. Utilizatorul trebuie să poată modifica condiţionarea şi să observe imediat efectul asupra zigzagului şi numărului de iteraţii.

Extensie: metoda gradientului conjugat pentru sisteme SPD

Pentru $A = A^T \succ 0$, sistemul $Ax = b$ poate fi rezolvat prin minimizarea funcţiei

$$\phi(x) = \frac{1}{2}x^T Ax - b^T x, \qquad \nabla\phi(x) = Ax - b.$$

Metoda gradientului foloseşte direcţia $r_k = b - Ax_k$. În metoda gradientului conjugat (CG), direcţiile p_k sunt alese astfel încât

$$p_i^T A p_j = 0, \qquad i \neq j.$$

Algoritmul CG

Alegem x_0 şi punem $r_0 = b - Ax_0$, $p_0 = r_0$. Pentru $k = 0, 1, \dots$ calculăm

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}, \qquad x_{k+1} = x_k + \alpha_k p_k,$$

$$r_{k+1} = r_k - \alpha_k A p_k, \qquad \beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}, \qquad p_{k+1} = r_{k+1} + \beta_k p_k.$$

Iteraţia se opreşte când reziduul relativ satisface criteriul ales.

În aritmetică exactă, CG găseşte soluţia în cel mult n paşi. Pentru analiza convergenţei se foloseşte norma $\|z\|_A = (z^T A z)^{1/2}$. Dacă $\kappa = \kappa_2(A)$, atunci

$$\|x_k - x_*\|_A \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|x_0 - x_*\|_A.$$

Faţă de estimarea corespunzătoare metodei gradientului, dependenţa este de $\sqrt{\kappa}$ şi nu de κ. Pentru sisteme SPD mari şi rare, acest fapt face metoda CG mult mai eficientă în multe situaţii.

Precondiţionare

Fie M o matrice SPD pentru care sistemele $Mz = r$ se rezolvă uşor. Precondiţionarea urmăreşte ca valorile proprii relevante să fie mai bine grupate pentru problema transformată. În calcule nu se formează M^{-1}; la fiecare pas se rezolvă un sistem cu matricea M.

Gradient şi gradient conjugat

Pentru o matrice diagonală SPD cu două valori proprii foarte diferite, comparaţi metoda gradientului cu pas exact şi metoda CG. Urmăriţi eroarea şi reziduul la fiecare pas şi explicaţi diferenţa dintre cele două metode.

Sarcini de aprofundare: metode de descreştere şi gradient

Probleme de aprofundare: geometrie, pas şi condiţionare

1. Pentru $f(x) = \frac{1}{2}x^T A x - b^T x$ cu A SPD, demonstraţi că minimizatorul unic este soluţia lui $Ax = b$ şi că $\nabla f(x) = Ax - b$ este opusul reziduului liniar.
2. Pentru aceeaşi funcţie, deduceţi pasul exact al metodei gradientului $\alpha_k = (r_k^T r_k)/(r_k^T A r_k)$, cu o convenţie coerentă pentru reziduu.
3. Demonstraţi că direcţia $-\nabla f(x)$ este direcţie de descreştere atunci când gradientul este nenul. Ce ipoteză de regularitate este folosită?
4. Pentru $A = \mathrm{diag}(1, \kappa)$, reprezentaţi geometric curbele de nivel şi explicaţi de ce traiectoria gradientului devine în zig-zag când κ este mare.
5. Comparaţi un pas fix α, pasul exact şi backtracking pe o funcţie pătratică. Ce informaţie suplimentară necesită fiecare variantă?
6. Determinaţi intervalul de valori ale pasului fix pentru care metoda converge pe o funcţie pătratică SPD în funcţie de $\lambda_{\max}(A)$.
7. Arătaţi că pentru pasul optim constant $\alpha = 2/(\lambda_{\min} + \lambda_{\max})$, factorul spectral este $(\kappa - 1)/(\kappa + 1)$. Interpretaţi dependenţa de condiţionare.
8. Explicaţi cum o schimbare de variabile $x = Dy$ poate fi văzută ca o precondiţionare a geometriei problemei.
9. Pentru o funcţie neliniară convexă, formulaţi condiţia Armijo şi explicaţi de ce ea asigură o descreştere suficientă, nu neapărat minimizarea pe direcţie.
10. Construiţi o funcţie simplă pentru care un pas fix prea mare produce oscilaţii sau

divergenţă. Verificaţi analitic şi numeric.

11. Discutaţi diferenţa dintre criteriile de oprire $\|\nabla f(x_k)\|$, $\|x_{k+1} - x_k\|$ şi reducerea relativă a lui f. De ce este utilă combinarea lor?
12. Pentru $f(x) = \frac{1}{2}\|Ax - b\|_2^2$, scrieţi gradientul fără a forma A^TA. De ce este această formă potrivită pentru implementări în care nu formăm explicit matricea?

Laborator integrator: aceeaşi problemă cu condiţionări controlate

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Observaţi legătura dintre forma curbelor de nivel, spectru şi numărul de iteraţii.

1. Construiţi în $\mathbb{R}^2$ matrici SPD cu valori proprii $(1, 2)$, $(1, 10)$, $(1, 100)$, $(1, 1000)$ şi aceiaşi vectori proprii rotiţi.
2. Pentru fiecare matrice reprezentaţi curbele de nivel şi traiectoriile metodei gradientului cu pas exact din aceeaşi stare iniţială.
3. Salvaţi $f(x_k) - f(x_\star)$, $\|\nabla f(x_k)\|$ şi eroarea $\|x_k - x_\star\|$.
4. Comparaţi pasul exact cu un pas fix bazat pe $\lambda_{\max}$ şi cu backtracking Armijo.
5. Estimaţi experimental rata asimptotică şi comparaţi-o cu factorii teoretici disponibili pentru cazul pătratic.
6. Aplicaţi o scalare diagonală care reduce condiţionarea şi repetaţi. Reprezentaţi traiectoria în variabilele originale.
7. Pentru o problemă mai mare şi rară, implementaţi produsul $A^T(Ax - b)$ fără a forma A^TA şi comparaţi memoria.
8. Formulaţi o concluzie despre ce parte a performanţei vine din alegerea pasului şi ce parte din geometria problemei.

Proiect de capitol: least squares fără formarea explicită a matricei prin gradient

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Consideraţi o matrice rară $A \in \mathbb{R}^{m\times n}$ cu $m \gg n$ sau cu dimensiune suficient de mare încât formarea lui A^TA să fie nefavorabilă.

1. **Modelare.** Definiţi $f(x) = \frac{1}{2}\|Ax - b\|_2^2$ şi deduceţi gradientul.
2. **Calcul manual.** Pentru un exemplu 4×2, executaţi două iteraţii complete.
3. **Algoritm.** Implementaţi gradient cu pas fix şi backtracking folosind doar produse cu A şi A^T.
4. **Experiment.** Construiţi cazuri cu condiţionări diferite şi comparaţi numărul de iteraţii.

5. **Scalare.** Normalizaţi coloanele lui A şi studiaţi efectul asupra convergenţei.
6. **Validare.** Comparaţi soluţia finală cu QR sau `A\b` pe dimensiuni moderate.
7. **Cost.** Raportaţi memoria şi timpul per iteraţie şi explicaţi de ce metoda poate deveni atractivă pentru probleme mari.

Test de capitol

Test – Metode de descreştere şi metoda gradientului

1. Calcul explicit. Se consideră funcţia pătratică

$$f(x) = \frac{1}{2}x^T Ax - b^T x, \qquad A = \begin{pmatrix} 4 & 1 \\ 1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

a) Calculaţi $\nabla f(x)$.
b) Pornind din $x^{(0)} = (0,0)^T$, calculaţi o iteraţie a metodei gradientului cu pasul $\alpha = 1/4$.
c) Verificaţi dacă valoarea funcţiei a scăzut.

2. Analiza codului.

```
g = A'*(A*x-b);
x = x-alpha*g;
```

Pentru ce funcţie f reprezintă g gradientul? Care dintre următoarele afirmaţii sunt corecte?

A) codul realizează un pas de gradient pentru problema celor mai mici pătrate;
B) alegerea pasului poate influenţa convergenţa;
C) pentru orice $\alpha > 0$ funcţia scade la fiecare pas;
D) un criteriu de oprire poate folosi mărimea gradientului sau variaţia iteraţiilor.

Capitolul 10

Metoda Gauss–Newton pentru cele mai mici pătrate neliniare

Metoda Gauss–Newton: cele mai mici pătrate neliniare

Există numeroase situaţii în care datele conduc la un sistem de ecuaţii neliniare $f_i(x) = c_i$, $i = 1, \dots, m$, unde $f_i : \mathbb{R}^n \to \mathbb{R}$ şi $c_i \in \mathbb{R}$ sunt cunoscute, iar vectorul x trebuie determinat. În prezenţa zgomotului sau atunci când $m > n$, nu ne aşteptăm în general ca toate ecuaţiile să poată fi satisfăcute exact. În loc să cerem $f_i(x) = c_i$ pentru fiecare i, măsurăm abaterea totală prin suma pătratelor reziduurilor.

Problema celor mai mici pătrate pentru modele neliniare (NLS)

Definim

$$g(x) := \sum_{i=1}^{m} \big(f_i(x) - c_i\big)^2.$$

Problema NLS constă în determinarea unui minimizator al lui g, adică în rezolvarea problemei $\min_{x \in \mathbb{R}^n} g(x)$.

Spre deosebire de cazul liniar, nu există o formulă universală care să rezolve problema într-un număr finit de paşi. Metoda Gauss–Newton construieşte, la fiecare iteraţie, o problemă liniară de tip cele mai mici pătrate obţinută prin liniarizarea reziduurilor. Vom presupune că funcţiile f_i sunt cel puţin continuu diferenţiabile.

Introducem vectorul reziduurilor

$$F(x) = \begin{pmatrix} f_1(x) - c_1 \\ f_2(x) - c_2 \\ \vdots \\ f_m(x) - c_m \end{pmatrix} \in \mathbb{R}^m,$$

astfel încât problema devine $\min_{x \in \mathbb{R}^n} \|F(x)\|_2^2$. Dacă x_k este aproximaţia curentă, liniarizarea de ordinul întâi este $F(x) \approx F(x_k) + J(x_k)(x - x_k)$, unde

$$J(x_k) = \begin{pmatrix} \nabla f_1(x_k)^T \\ \nabla f_2(x_k)^T \\ \vdots \\ \nabla f_m(x_k)^T \end{pmatrix} \in \mathbb{R}^{m \times n}$$

este matricea Jacobiană.

Pasul Gauss–Newton alege următoarea aproximaţie minimizând suma pătratelor liniarizate:

$$x_{k+1} = \operatorname*{argmin}_{x\in\mathbb{R}^n} \sum_{i=1}^{m} \left[f_i(x_k) + \langle \nabla f_i(x_k), x - x_k \rangle - c_i \right]^2 .$$

Aceasta este problema celor mai mici pătrate pentru modelul liniarizat. Dacă o scriem sub forma $\min_x \|A_k x - b_k\|_2^2$, atunci $A_k = J(x_k)$ şi

$$b_k = J(x_k)x_k - F(x_k).$$

Dacă $J(x_k)$ are rang coloană complet, ecuaţiile normale dau

$$x_{k+1} = \Big(J(x_k)^T J(x_k)\Big)^{-1} J(x_k)^T b_k = x_k - \Big(J(x_k)^T J(x_k)\Big)^{-1} J(x_k)^T F(x_k).$$

În implementare nu vom forma explicit inversa; rezolvăm sistemul normal sau, preferabil atunci când precizia este importantă, subproblema cele mai mici pătrate prin QR.

Dacă definim

$$d_k = \Big(J(x_k)^T J(x_k)\Big)^{-1} J(x_k)^T F(x_k),$$

atunci $x_{k+1} = x_k - d_k$. Deoarece $\nabla g(x) = 2J(x)^T F(x)$, rezultă

$$d_k = \frac{1}{2}\Big(J(x_k)^T J(x_k)\Big)^{-1} \nabla g(x_k).$$

Aşadar, atunci când $J(x_k)$ are rang coloană complet, Gauss–Newton poate fi privită ca o metodă de gradient scalat, matricea de scalare fiind pozitiv definită. Această observaţie explică şi caracterul de direcţie de descreştere al pasului, atâta timp cât gradientul este nenul.

Metoda Gauss–Newton amortizată

Varianta pură foloseşte direct pasul $x_{k+1} = x_k - d_k$. Departe de soluţie, liniarizarea poate fi însă insuficient de precisă şi pasul complet poate mări funcţia obiectiv. Introducem de aceea un factor $t_k > 0$, ales printr-o procedură de căutare liniară, şi folosim $x_{k+1} = x_k - t_k d_k$.

Algoritm: Gauss–Newton amortizat

- **Date de intrare:** o toleranţă $\varepsilon > 0$ şi un punct iniţial $x_0 \in \mathbb{R}^n$.
- **Pentru** $k = 0, 1, 2, \ldots$:
  - calculaţi $F(x_k)$ şi $J(x_k)$;
  - determinaţi d_k din problema $\min_d \|J(x_k)d - F(x_k)\|_2^2$, echivalent, în cazul rangului complet, cu $(J^T J)d_k = J^T F$;
  - alegeţi t_k printr-o căutare liniară aplicată funcţiei $h(t) = g(x_k - t d_k)$;
  - puneţi $x_{k+1} = x_k - t_k d_k$;
  - opriţi dacă $\|\nabla g(x_{k+1})\|_2 \le \varepsilon$ sau dacă un alt criteriu de oprire justificat este satisfăcut.

Ce aproximăm de fapt?

Metoda Newton liniarizează condiţia de staţionaritate şi foloseşte Hessiana exactă. Gauss–Newton exploatează structura specială $g = \|F\|^2$ şi păstrează în Hessiană termenul dominant $2J^T J$, neglijând termenii care conţin reziduurile. Când reziduurile la soluţie sunt mici, această aproximaţie este adesea foarte bună; când ele sunt mari sau Jacobianul este aproape deficient de rang, metoda poate necesita amortizare ori regularizare.

Extensie didactică: Gauss–Newton ca secvenţă de probleme LS liniare

Considerăm o funcţie de reziduuri $r : \mathbb{R}^n \to \mathbb{R}^m$ şi problema $f(x) = \frac{1}{2}\|r(x)\|_2^2$. Gradientul este $\nabla f(x) = J(x)^T r(x)$, unde J este Jacobianul reziduurilor. Hessiana exactă este

$$\nabla^2 f(x) = J(x)^T J(x) + \sum_{i=1}^{m} r_i(x)\nabla^2 r_i(x).$$

Metoda Gauss–Newton neglijează al doilea termen şi foloseşte aproximaţia $J^T J$. Această aproximaţie este potrivită în apropierea unei soluţii pentru care reziduurile sunt mici sau problema este aproape liniară.

Derivarea prin liniarizarea reziduurilor

În jurul lui x_k, avem $r(x_k + p) \approx r_k + J_k p$. În loc să minimizăm problema neliniară completă, alegem p_k ca soluţie a problemei liniare LS

$$\min_p \|J_k p + r_k\|_2^2.$$

Ecuaţiile normale sunt $J_k^T J_k p = -J_k^T r_k$, dar numeric putem rezolva subproblema prin QR. Actualizarea neamortizată este $x_{k+1} = x_k + p_k$; o variantă cu control al pasului foloseşte $x_{k+1} = x_k + t_k p_k$ şi o căutare a pasului (line search).

Legătura cu Newton

Newton foloseşte Hessiana exactă a lui f. Gauss–Newton exploatează structura specială "sumă de pătrate" şi înlocuieşte Hessiana cu $J^T J$, care este pozitiv semidefinită. Astfel, problema neliniară este atacată printr-o succesiune de probleme de algebră liniară numerică – exact tema manualului.

Exemplu: ajustare exponenţială

Pentru date (t_i, y_i) şi modelul $y \approx ae^{bt}$, reziduul este $r_i(a, b) = ae^{bt_i} - y_i$. Jacobianul are coloanele $\partial r_i/\partial a = e^{bt_i}$ şi $\partial r_i/\partial b = at_i e^{bt_i}$. La fiecare iteraţie evaluăm aceste cantităţi în parametrii curenţi şi rezolvăm o problemă LS cu două necunoscute.

Laborator: Gauss–Newton cu QR

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Generaţi date din $y = 2e^{-0.7t}$ cu zgomot. Implementaţi Gauss–Newton astfel încât subproblema liniară să fie rezolvată prin `J\(-r)` sau prin QR, nu prin formarea explicită a inversei lui $J^T J$. Testaţi mai multe iniţializări. Adăugaţi o procedură de backtracking pe funcţia f dacă pasul complet creşte costul.

Exerciţii şi verificarea înţelegerii

1. Derivaţi formula gradientului $J^T r$ componentă cu componentă.
2. Derivaţi Hessiana exactă şi identificaţi termenul neglijat de Gauss–Newton.
3. Când reziduul la soluţie este exact zero, ce se întâmplă cu termenul suplimentar din Hessiană în acel punct?
4. Pentru modelul exponenţial de mai sus, scrieţi explicit Jacobianul pentru trei observaţii.
5. Explicaţi de ce rangul deficient al lui J_k poate face pasul Gauss–Newton neunic sau foarte sensibil.
6. Comparaţi conceptual Gauss–Newton amortizat cu metoda gradientului cu backtracking.

Mini-proiect: circle fitting geometric

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Folosiţi problema cercului din capitolul anterior. Implementaţi Gauss–Newton pentru reziduurile geometrice şi comparaţi: (i) iniţializare din media punctelor, (ii) iniţializare dintr-o ajustare algebrică, (iii) iniţializare deliberat slabă. Studiaţi convergenţa, reziduul final şi sensibilitatea la puncte aberante.

Exemplu scalar cu două reziduuri incompatibile

Considerăm $r_1(x) = x^2 - 1$ şi $r_2(x) = x^2 - 4$, deci $f(x) = \frac{1}{2}[(x^2-1)^2 + (x^2-4)^2]$. Nu există x care să anuleze simultan ambele reziduuri. Pentru $x \neq 0$, punctele staţionare nenule satisfac $2x^2 - 5 = 0$, iar minimizatorul pozitiv este $x^* = \sqrt{5/2} \approx 1.5811$.

Pornim din $x_0 = 1.5$. Avem $r(x_0) = (1.25, -1.75)^T$ şi Jacobianul $J(x_0) = (3, 3)^T$. Pasul Gauss–Newton p rezolvă $\min_p \|(3p + 1.25, 3p - 1.75)^T\|_2^2$. Ecuaţia normală scalară este $18p = -3(1.25 - 1.75) = 1.5$, deci $p = 1/12 \approx 0.08333$. Rezultă $x_1 \approx 1.5833$, foarte aproape de minimizatorul exact.

Exemplul este mic, dar arată mecanismul complet: liniarizăm vectorul reziduurilor, rezolvăm problema celor mai mici pătrate pentru modelul liniarizat şi actualizăm parametrul. În dimensiuni mari exact aceleaşi trei etape rămân valabile.

Rang deficient şi amortizare

Dacă J_k nu are rang coloană complet, matricea $J_k^T J_k$ este singulară şi formula bazată pe inversă nu este disponibilă. Chiar înainte de singularitate, o condiţionare mare poate produce paşi foarte mari. O remediere uzuală este rezolvarea subproblemei printr-o factorizare QR cu pivotare; o altă idee este Levenberg–Marquardt, care adaugă un termen λI la $J^T J$. În acest fel, metoda Gauss–Newton se leagă de regularizarea studiată anterior.

Sarcină de programare: Gauss–Newton fără inverse

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Implementaţi metoda astfel încât pasul să fie calculat din `p = -J\r` sau din QR, nu din `inv(J'*J)`. Adăugaţi opţional un parametru de amortizare şi backtracking. În istoric păstraţi $\|r_k\|$, $\|J_k^T r_k\|$, $\|p_k\|$ şi pasul acceptat. Testaţi problema scalară de mai sus, circle fitting şi o problemă cu Jacobian aproape deficient de rang.

Exerciţii suplimentare

1. Derivaţi minimizatorul exact pozitiv al exemplului şi comparaţi cu x_1.
2. Calculaţi încă o iteraţie Gauss–Newton pentru problema scalară.
3. Scrieţi Hessiana exactă a lui $\frac{1}{2}\|r(x)\|^2$ şi identificaţi termenul neglijat de Gauss–Newton.
4. În ce situaţie termenul neglijat este exact zero?
5. Explicaţi de ce o căutare a pasului (line search) poate salva o iteraţie pentru care liniarizarea locală este prea optimistă.

Fişă de seminar şi laborator: Gauss–Newton

Problema rezolvată 1: derivarea Hessienei exacte

Pentru $g(x) = \frac{1}{2}\sum_i r_i(x)^2$, gradientul este $\nabla g = \sum_i r_i \nabla r_i = J^T r$. Derivând încă o dată,

$$\nabla^2 g = J^T J + \sum_i r_i \nabla^2 r_i.$$

Gauss–Newton păstrează primul termen. Dacă reziduurile la soluţie sunt zero, al doilea termen dispare exact în soluţie. Dacă reziduurile sunt mici, termenul este de ordinul lor şi aproximaţia poate fi bună. Dacă reziduurile sunt mari, metoda nu mai seamănă atât de bine cu Newton şi poate avea convergenţă mai lentă.

Problema rezolvată 2: model liniar ca caz particular

Dacă $r(x) = Ax - b$, Jacobianul este constant $J = A$, iar Hessienele individuale ale reziduurilor sunt zero. Gauss–Newton rezolvă atunci într-un singur pas problema celor mai mici pătrate, independent de punctul iniţial, dacă folosim pas complet şi rangul este adecvat. Acesta este un test excelent pentru implementare: algoritmul neliniar trebuie să reducă exact la algoritmul liniar pentru cele mai mici pătrate.

Laborator integrator 10: trei probleme NLS

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă

este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Testaţi aceeaşi implementare Gauss–Newton pe: (i) o problemă liniară LS, unde trebuie să termine practic într-un pas; (ii) circle fitting cu zgomot mic; (iii) o problemă cu reziduuri nenule la soluţie. Pentru fiecare comparaţi pas pur şi pas cu Armijo. Raportaţi norma reziduului, norma gradientului şi numărul de evaluări ale funcţiei.

Set de probleme

1. Derivaţi gradientul şi Hessiana pentru $g = \frac{1}{2}\|r\|^2$.
2. Identificaţi exact condiţia în care Gauss–Newton şi Newton au aceeaşi Hessiană.
3. Pentru un model liniar, demonstraţi terminarea într-un pas în aritmetică exactă.
4. Ce problemă apare dacă J_k are coloane aproape dependente?
5. Formulaţi pasul Levenberg–Marquardt $(J^TJ + \lambda I)p = -J^Tr$ şi interpretaţi λ.
6. De ce rezolvarea subproblemei prin QR este preferabilă construirii inversei lui J^TJ?
7. Ce criterii de oprire aţi folosi dacă reziduul minim nu este zero?
8. Construiţi o iniţializare pentru circle fitting care produce o problemă în Jacobian şi explicaţi de ce.

Proiect de capitol: localizare din distanţe

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Simulaţi 6 repere în plan şi un punct necunoscut. Generaţi distanţe cu zgomot, apoi estimaţi poziţia prin Gauss–Newton. Comparaţi repere bine distribuite cu repere aproape coliniare şi porniţi algoritmul din mai multe iniţializări. Raportaţi condiţionarea Jacobianului la soluţie şi eroarea poziţiei. Adăugaţi ponderi dacă zgomotul senzorilor este diferit.

10.1 Problema celor mai mici pătrate pentru modele neliniare: ajustarea cercurilor şi problema localizării

Problema celor mai mici pătrate pentru un model neliniar: ajustarea unui cerc

Există situaţii în care ni se dă un sistem de ecuaţii neliniare

$$f_i(x) = c_i, \qquad i = 1, 2, ..., m,$$

unde $f_i : \mathbb{R}^n \to \mathbb{R}$, $c_i \in \mathbb{R}$ sunt date şi x trebuie determinat.



În acest caz, problema de aproximare este problema celor mai mici pătrate pentru un model neliniar (NLS), formulată astfel

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} (f_i(x) - c_i)^2.$$

Nu există o modalitate uşoară de a rezolva problemele NLS. Metoda Gauss-Newton este o modalitate, dar aceasta converge numai către un punct staţionar.

Circle fitting

Să presupunem că ne sunt date m puncte $a_1, a_2, ..., a_m \in \mathbb{R}^n$. Problema adaptării cercului urmăreşte să găsească un cerc

$$C(x, r) = \{y \in \mathbb{R}^n : \|y - x\| = r\}$$

care se potriveşte cel mai bine punctelor m.

Aplicaţie?

Ecuaţiile neliniare asociate cu această problemă sunt

$$\|x - a_i\| = r, \qquad i = 1, 2, ..., m.$$

Deoarece dorim să avem de-a face cu funcţii diferenţiabile, iar funcţia normă nu este diferenţiabilă, vom considera versiunea pătratică a acesteia:

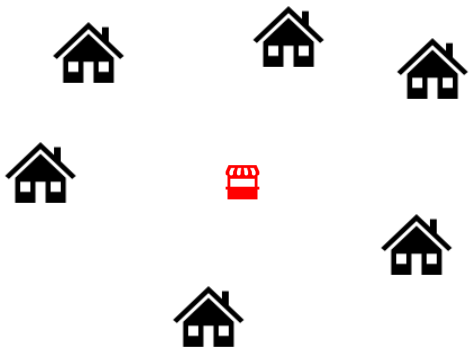
$$\|x - a_i\|^2 = r^2, \quad i = 1, 2, ..., m.$$

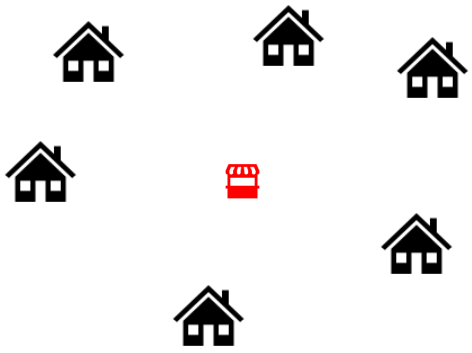
Problema locaţiei egal depărtate

$$\min_{x\in\mathbb{R}^n, r\in\mathbb{R}} \sum_{i=1}^{m} (\|x - a_i\|^2 - r^2)^2.$$

Problema locaţiei egal depărtate cu ponderi de importanţă

$$\min_{x\in\mathbb{R}^n, r\in\mathbb{R}} \sum_{i=1}^{m} \omega_i (\|x - a_i\|^2 - r^2)^2.$$





Problema NLS pentru ajustarea unui cerc

Problema NLS asociată cu aceste ecuaţii este

$$\min_{x\in\mathbb{R}^n, r\in\mathbb{R}_+} \sum_{i=1}^{m} (\|x-a_i\|^2 - r^2)^2.$$

Observaţie: În această formă nu avem o problemă de optimizare fără constrângeri!

Dar, de fapt, problema este echivalentă cu

$$\min_{x\in\mathbb{R}^n, r\in\mathbb{R}} \sum_{i=1}^{m} (-2\langle a_i, x\rangle + \|x\|^2 + \|a_i\|^2 - r^2)^2.$$

Efectuând schimbarea de variabile $R = \|x\|^2 - r^2$, problema de mai sus se reduce la

$$\min_{x\in\mathbb{R}^n, \|x\|^2\geq R} f(x,R) := \sum_{i=1}^{m} (-2\langle a_i, x\rangle + R + \|a_i\|^2)^2.$$

De fapt, orice soluţie optimă $(\widehat{x}, \widehat{R})$ a problemei

$$\min_{x\in\mathbb{R}^n, R\in\mathbb{R}} f(x,R) := \sum_{i=1}^{m} (-2\langle a_i, x\rangle + R + \|a_i\|^2)^2$$

satisface în mod automat $\|\widehat{x}\|^2 \geq \widehat{R}$, deoarece altfel

$$-2\langle a_i, \widehat{x}\rangle + \widehat{R} + \|a_i\|^2 > -2\langle a_i, \widehat{x}\rangle + \|\widehat{x}\|^2 + \|a_i\|^2 = \|\widehat{x} - a_i\|^2 \geq 0, \ \ i = 1, 2, ..., m.$$

Prin ridicarea la pătrat a ambelor părţi ale primei inegalităţi din ecuaţia de mai sus şi adunarea la i rezultă

$$\begin{aligned} f(\widehat{x}, \widehat{R}) &= \sum_{i=1}^{m} (-2\langle a_i, \widehat{x}\rangle + \widehat{R} + \|a_i\|^2)^2 \\ &> \sum_{i=1}^{m} (-2\langle a_i, \widehat{x}\rangle + \|\widehat{x}\|^2 + \|a_i\|^2)^2 = \sum_{i=1}^{m} \|\widehat{x} - a_i\|^4 = f(\widehat{x}, \|\widehat{x}\|^2), \end{aligned}$$

arătând că $(\widehat{x}, \|\widehat{x}\|^2)$ conduce la o valoare a funcţiei mai mică decât $(\widehat{x}, \widehat{R})$, în contradicţie cu optimalitatea lui $(\widehat{x}, \widehat{R})$.

În concluzie, problema NLS

$$\min_{x\in\mathbb{R}^n, \|x\|^2\geq R} f(x,R) := \sum_{i=1}^{m} (-2\langle a_i, x\rangle + R + \|a_i\|^2)^2$$

este de fapt echivalentă cu problema LS

$$\min_{(x,R)\in\mathbb{R}^{n+1}} f(x,R) := \|A \begin{pmatrix} x \\ R \end{pmatrix} - b\|^2,$$

unde $A = \begin{pmatrix} 2\,a_1^T & -1 \\ 2\,a_2^T & -1 \\ \vdots & \vdots \\ 2\,a_m^T & -1 \end{pmatrix}$ şi $b = \begin{pmatrix} \|a_1\|^2 \\ \|a_2\|^2 \\ \vdots \\ \|a_m\|^2 \end{pmatrix}$.

Dacă A este de rang maxim, atunci soluţia unică a problemei celor mai mici pătrate pentru sistemul liniar este

$$\begin{pmatrix} x \\ R \end{pmatrix} = (A^T A)^{-1} A^T b.$$

Optimul x este dat de primele n componente, iar raza r este dată de

$$r = \sqrt{\|x\|^2 - R}.$$

Completare didactică: de ce circle fitting este neliniar?

În ajustarea unei drepte, necunoscutele apar liniar. Pentru un cerc cu centru $c = (a, b)$ şi rază R, distanţa unui punct $p_i = (x_i, y_i)$ la centru este $\|p_i - c\|_2$, iar condiţia ideală este $\|p_i - c\|_2 = R$. Reziduul geometric $r_i(a, b, R) = \sqrt{(x_i - a)^2 + (y_i - b)^2} - R$ depinde neliniar de centru. Minimizarea lui $\sum_i r_i^2$ conduce deci la problema celor mai mici pătrate, în varianta neliniară.

Există şi o "liniarizare algebrică" pornind de la $(x_i - a)^2 + (y_i - b)^2 = R^2$, prin introducerea unor parametri combinaţi. Ea poate oferi o iniţializare bună, dar nu minimizează în general aceeaşi eroare geometrică. Distincţia dintre distanţa algebrică şi cea geometrică este importantă în aplicaţii.

Localizare prin distanţe

Aceeaşi structură apare când vrem să găsim poziţia necunoscută $c \in \mathbb{R}^2$ din distanţe măsurate d_i faţă de puncte de referinţă s_i: $\|c - s_i\| \approx d_i$. Cu mai mult de trei repere şi date zgomotoase, obţinem o problemă redundantă neliniară. Dacă unele măsurători sunt mai precise, folosim ponderi w_i şi minimizăm $\sum_i w_i(\|c - s_i\| - d_i)^2$.

Jacobianul reziduurilor

Pentru $r_i(c) = \|c - s_i\| - d_i$ şi $c \neq s_i$, gradientul reziduului este $(c - s_i)^T / \|c - s_i\|$. Punând aceste linii într-o matrice Jacobian $J(c)$ obţinem exact obiectul de care va avea nevoie metoda Gauss–Newton. Astfel, această secţiune pregăteşte natural algoritmul neliniar studiat mai târziu.

Laborator: localizare 2D

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Alegeţi patru puncte de reper în plan şi o poziţie adevărată $c_\star$. Calculaţi distanţele exacte, adăugaţi zgomot şi reprezentaţi cercurile corespunzătoare. Implementaţi funcţia reziduurilor şi funcţia cost. Folosiţi mai multe puncte iniţiale şi o rutină de optimizare disponibilă pentru a observa dacă soluţia depinde de iniţializare. Înregistraţi norma reziduului final.

Exerciţii şi verificarea înţelegerii

1. Derivaţi componentele Jacobianului pentru reziduul $r_i(a, b, R)$ al problemei de circle fitting.
2. Ce singularitate apare dacă centrul curent coincide cu unul dintre punctele observate în modelul bazat pe distanţă?
3. Formulaţi problema de localizare ponderată atunci când varianţa erorii senzorului i este aproximativ σ_i^2. Ce alegere naturală au ponderile?
4. Construiţi o configuraţie de repere care produce o localizare slab condiţionată geometric.
5. Explicaţi de ce mai multe măsurători nu garantează automat o localizare bună dacă toate reperele sunt aproape coliniare.

Mini-proiect: GPS în miniatură

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Simulaţi localizarea unui punct într-un pătrat folosind distanţe faţă de 3, 4, 6 şi 10 repere. Pentru fiecare număr de repere, repetaţi experimentul cu zgomot şi calculaţi eroarea poziţiei. Comparaţi o configuraţie de repere bine distribuită cu una aproape coliniară. Discutaţi efectul geometriei asupra sensibilităţii problemei.

Exemplu rezolvat: primul pas Gauss–Newton pentru un cerc

Luăm patru puncte situate pe cercul cu centrul $(1, 2)$ şi raza 2: $(3, 2)$, $(-1, 2)$, $(1, 4)$, $(1, 0)$. Pornim intenţionat din aproximaţia $x_0 = (a_0, b_0, R_0) = (0.5, 1.5, 1.5)$. Pentru fiecare punct definim $r_i(a, b, R) = \sqrt{(x_i - a)^2 + (y_i - b)^2} - R$. La x_0, vectorul reziduurilor este aproximativ

$$r(x_0) = (1.0495, 0.0811, 1.0495, 0.0811)^T.$$

Linia i a Jacobianului este

$$\left(\frac{a - x_i}{d_i}, \frac{b - y_i}{d_i}, -1 \right), \qquad d_i = \sqrt{(x_i - a)^2 + (y_i - b)^2}.$$

Rezolvând subproblema

$$\min_p \| J(x_0)p + r(x_0) \|_2,$$

obţinem aproximativ $p = (0.5353, 0.5353, 0.4197)^T$. Noua aproximaţie este $x_1 = x_0 + p \approx (1.0353, 2.0353, 1.9197)$. Norma reziduului scade de la aproximativ 1.489 la 0.176. O singură liniarizare ne-a adus deja aproape de cercul exact.

Exemplul arată şi de ce iniţializarea contează. Formula Jacobianului conţine distanţele d_i la numitor; dacă centrul curent coincide cu un punct observat, derivata nu este definită. Mai general, departe de soluţie, modelul liniar local poate fi o aproximaţie slabă şi este utilă o amortizare a pasului.

Sarcină de programare: circle fitting complet

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Scrieţi funcţii separate pentru reziduuri şi Jacobian. Implementaţi Gauss–Newton folosind QR pentru subproblema celor mai mici pătrate, nu formula explicită $(J^TJ)^{-1}J^T$. Adăugaţi o procedură de backtracking dacă pasul complet nu micşorează suma pătratelor. Testaţi pe puncte exacte, apoi pe puncte cu zgomot. Porniţi din cel puţin zece iniţializări şi reprezentaţi centrul final, raza şi istoricul funcţiei obiectiv.

Exerciţii suplimentare

1. Derivaţi cu atenţie cele trei derivate parţiale ale lui $r_i(a, b, R)$.
2. Verificaţi numeric pasul indicat în exemplu şi recalculaţi reziduul în x_1.
3. Comparaţi ajustarea geometrică cu linearizarea algebrică a ecuaţiei cercului. De ce funcţiile obiectiv nu sunt identice?
4. Introduceţi ponderi diferite pentru puncte. Cum se modifică subproblema Gauss–Newton?
5. Construiţi un set de puncte aproape coliniare şi observaţi instabilitatea estimării cercului.

Sarcini de aprofundare şi sinteză pentru Gauss–Newton

Probleme de aprofundare: linearizare, Jacobian şi robusteţe

1. Pentru $\Phi(x) = \frac{1}{2}\|r(x)\|_2^2$, demonstraţi formula $\nabla\Phi(x) = J(x)^T r(x)$.
2. Pornind de la dezvoltarea $r(x + s) \approx r(x) + J(x)s$, deduceţi subproblema LS a metodei Gauss–Newton şi ecuaţiile normale $J^TJs = -J^Tr$.
3. Explicaţi de ce rezolvarea subproblemei prin QR este preferabilă formării explicite a lui J^TJ când Jacobianul este slab condiţionat.
4. Arătaţi că pentru un reziduu liniar $r(x) = Ax - b$, Gauss–Newton se reduce la rezolvarea problemei LS liniare.
5. Comparaţi matricea Hessiană exactă a lui Φ cu aproximaţia J^TJ. Ce termen este neglijat şi când este el mic?
6. Pentru ajustarea unui cerc, scrieţi reziduurile geometrice şi derivatele lor faţă de centru şi rază. Identificaţi un caz în care derivata poate deveni problematică.
7. Pentru localizarea din distanţe, discutaţi efectul unei configuraţii de repere aproape coliniare asupra rangului şi condiţionării lui J.
8. Formulaţi o variantă amortizată: $(J^TJ + \lambda I)s = -J^Tr$. Adăugarea termenului λI, cu $\lambda > 0$, stă la baza metodei Levenberg–Marquardt; explicaţi rolul parametrului λ în regularizarea subproblemei locale.
9. Construiţi un exemplu scalar neliniar în care pasul Gauss–Newton complet creşte funcţia obiectiv. Arătaţi cum un line search poate corecta pasul.

10. Discutaţi trei criterii de oprire: normă gradient, normă pas şi reducere relativă a reziduului. De ce “reziduu mic” nu este întotdeauna posibil într-un model cu zgomot?
11. Explicaţi efectul scalării parametrilor dacă un parametru este de ordin 10^{-3} şi altul de ordin 10^{3}. Cum poate fi modificată geometria subproblemei?
12. Distingeţi între convergenţă locală rapidă şi robusteţe globală. Ce mecanisme din curs pot îmbunătăţi a doua proprietate?

Laborator integrator: localizare 2D cu iniţializări şi zgomot

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Analizaţi Gauss–Newton ca algoritm complet, nu doar ca formulă de pas.

1. Alegeţi cel puţin cinci repere în plan şi o poziţie adevărată $x_\star$. Generaţi distanţe exacte şi apoi distanţe zgomotoase.
2. Derivaţi şi implementaţi vectorul rezidual şi Jacobianul. Verificaţi Jacobianul prin diferenţe finite pe un punct de test.
3. Implementaţi Gauss–Newton cu subproblema rezolvată prin QR, fără inverse şi fără `J'*J` în versiunea principală.
4. Testaţi cel puţin şase iniţializări, inclusiv unele departe de soluţie. Salvaţi istoricul funcţiei obiectiv, normei gradientului şi pasului.
5. Adăugaţi backtracking Armijo şi comparaţi cazurile în care pasul complet este acceptat sau redus.
6. Comparaţi o geometrie bună a reperelor cu una aproape coliniară. Raportaţi `cond(J)` la iteraţii reprezentative.
7. Introduceţi un outlier într-o singură distanţă şi observaţi deplasarea soluţiei. Explicaţi efectul prin natura LS.
8. Comparaţi rezultatul final cu o rutină MATLAB de referinţă, dacă este disponibilă, doar pentru verificare.

De explicat la predare. Care este diferenţa dintre o subproblemă liniară bine rezolvată şi o linearizare locală care nu descrie încă bine funcţia neliniară?

Proiect de capitol: identificarea parametrilor unui model neliniar

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Alegeţi un model cu doi-trei parametri, de exemplu $y(t) = ae^{-bt} + c$, şi generaţi sau folosiţi date experimentale.

1. **Modelare.** Definiţi vectorul rezidual şi parametrii fizici.

2. **Derivare.** Calculaţi Jacobianul analitic şi verificaţi-l numeric.
3. **Iniţializare.** Propuneţi o strategie rezonabilă pentru alegerea parametrilor iniţiali.
4. **Algoritm.** Implementaţi Gauss–Newton cu QR şi backtracking.
5. **Experimente.** Variaţi zgomotul şi iniţializarea; raportaţi rata de succes şi numărul de iteraţii.
6. **Sensibilitate.** Studiaţi condiţionarea Jacobianului la soluţie şi variaţia parametrilor la perturbarea datelor.
7. **Validare.** Reprezentaţi datele şi curba ajustată şi raportaţi reziduurile, nu doar parametrii finali.
8. **Interpretare.** Separaţi erorile datorate modelului, datelor şi algoritmului.

Test de capitol

Test – Metoda Gauss–Newton

1. Calcul explicit. Se consideră problema celor mai mici pătrate în varianta neliniară

$$\min_{x\in\mathbb{R}} \frac{1}{2}\Big(r_1(x)^2 + r_2(x)^2\Big), \qquad r_1(x) = x^2 - 1, \quad r_2(x) = x - 1.$$

Pornind din $x^{(0)} = 2$, calculaţi matricea Jacobiană $J(x^{(0)})$, vectorul reziduurilor şi pasul Gauss–Newton $p^{(0)}$ obţinut din ecuaţiile normale ale problemei liniarizate. Determinaţi $x^{(1)}$.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi.

A) Gauss–Newton înlocuieşte local problema neliniară prin problema celor mai mici pătrate pentru modelul liniarizat;

B) Jacobiana reziduurilor trebuie recalculată în general la fiecare iteraţie;

C) metoda coincide întotdeauna cu metoda Newton;

D) alegerea punctului iniţial poate influenţa comportarea metodei.

Capitolul 11

Descompunerea în valori singulare şi pseudoinversa

Descompunerea în valori singulare (SVD)

Descompunerea în valori singulare (SVD) este una dintre factorizările centrale ale calculului ştiinţific. Ea descrie simultan geometria aplicaţiei liniare $x \mapsto Ax$, rangul numeric al matricei, amplificarea erorilor, problema celor mai mici pătrate şi aproximaţiile de rang mic.

Teoremă 11.0.1 (Existenţa descompunerii în valori singulare)**.** *Fie $A \in \mathbb{R}^{m\times n}$ şi $q = \min(m, n)$. Atunci există matrice ortogonale $U \in \mathbb{R}^{m\times m}$ şi $V \in \mathbb{R}^{n\times n}$ şi o matrice diagonală dreptunghiulară $\Sigma \in \mathbb{R}^{m\times n}$, cu elemente diagonale $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q \geq 0$, astfel încât*

$$A = U\Sigma V^T.$$

Numerele σ_i se numesc valorile singulare ale lui A.

Demonstraţie. Considerăm $B = A^TA \in \mathbb{R}^{n\times n}$. Matricea B este simetrică şi pozitiv semidefinită, deoarece, pentru orice $x \in \mathbb{R}^n$,

$$x^TBx = x^TA^TAx = \|Ax\|_2^2 \geq 0.$$

Prin teorema spectrală există o bază ortonormală $v_1, \ldots, v_n$ de vectori proprii ai lui B, cu valori proprii ordonate

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0.$$

Dacă $n > m$, cel puţin $n-m$ dintre aceste valori sunt zero, deoarece $\operatorname{rank}(A^TA) = \operatorname{rank}(A) \leq m$. Punem $\sigma_i = \sqrt{\lambda_i}$ pentru $i = 1, \ldots, q$ şi notăm cu $r = \operatorname{rank} A$ numărul valorilor strict pozitive. Astfel $\sigma_1 \geq \cdots \geq \sigma_r > 0$, iar celelalte valori singulare sunt zero.

Pentru $i = 1, \ldots, r$ definim $u_i = Av_i/\sigma_i$. Atunci

$$\|u_i\|_2^2 = \frac{v_i^TA^TAv_i}{\sigma_i^2} = \frac{\lambda_i\|v_i\|_2^2}{\sigma_i^2} = 1,$$

iar pentru $i \neq j$,

$$u_i^Tu_j = \frac{v_i^TA^TAv_j}{\sigma_i\sigma_j} = \frac{\lambda_jv_i^Tv_j}{\sigma_i\sigma_j} = 0.$$

Prin urmare $u_1, \ldots, u_r$ sunt ortonormali în $\mathbb{R}^m$. Îi completăm la o bază ortonormală $u_1, \ldots, u_m$ şi definim

$$U = (u_1|\cdots|u_m), \qquad V = (v_1|\cdots|v_n).$$

Ambele matrice sunt ortogonale.

Pentru $i \leq r$, prin definiţie, $Av_i = \sigma_i u_i$. Pentru $i > r$, valoarea proprie corespunzătoare lui A^TA este zero şi

$$\|Av_i\|_2^2 = v_i^T A^T A v_i = 0,$$

deci $Av_i = 0$. Dacă $\Sigma \in \mathbb{R}^{m\times n}$ are pe diagonala dreptunghiulară valorile $\sigma_1, \ldots, \sigma_r$ şi zero în rest, relaţiile precedente se scriu simultan $AV = U\Sigma$. Înmulţind la dreapta cu $V^T = V^{-1}$ obţinem $A = U\Sigma V^T$. □

Neunicitatea vectorilor singulari

Valorile singulare σ_i sunt determinate în mod unic de A, dar matricile U şi V nu sunt în general unice. Chiar pentru o valoare singulară simplă, perechea (u_i, v_i) poate fi înlocuită cu $(-u_i, -v_i)$. Pentru valori singulare multiple, orice schimbare ortogonală de bază în subspaţiul singular corespunzător produce o altă SVD validă. De aceea MATLAB şi un calcul manual pot furniza vectori singulari diferiţi, deşi descompunerile sunt echivalente.

Definiţie 11.0.2. *Coloanele $u_1, \ldots, u_m$ ale lui U se numesc vectori singulari stângi, iar coloanele $v_1, \ldots, v_n$ ale lui V se numesc vectori singulari drepţi. Pentru $i \leq r$ avem $Av_i = \sigma_i u_i$ şi $A^Tu_i = \sigma_i v_i$.*

Propoziţie 11.0.3 (Proprietăţi fundamentale ale SVD)**.** *Fie $A = U\Sigma V^T$ şi $r = \operatorname{rank} A$. Atunci:*

1. *pentru $i = 1, \ldots, r$, $Av_i = \sigma_i u_i$ şi $A^Tu_i = \sigma_i v_i$;*
2. *$A^TAv_i = \sigma_i^2 v_i$ pentru $i = 1, \ldots, n$, unde convenim $\sigma_i = 0$ pentru indicii suplimentari când $n > q$; analog, $AA^Tu_i = \sigma_i^2 u_i$ pentru $i = 1, \ldots, m$, cu aceeaşi convenţie;*
3. *$A = \sum_{i=1}^r \sigma_i u_i v_i^T$;*
4. *pentru orice $x \in \mathbb{R}^n$, $\|Ax\|_2 \leq \sigma_1\|x\|_2$; dacă $m \geq n$ şi A are rang coloană complet, atunci*

$$\sigma_n\|x\|_2 \leq \|Ax\|_2 \leq \sigma_1\|x\|_2;$$

5. *$\ker A = \operatorname{span}\{v_{r+1}, \ldots, v_n\}$ şi $\operatorname{range} A = \operatorname{span}\{u_1, \ldots, u_r\}$.*

Demonstraţie. Prima relaţie de la punctul 1 a fost obţinută în demonstraţia teoremei. Pentru a doua,

$$A^Tu_i = \frac{1}{\sigma_i}A^TAv_i = \frac{\sigma_i^2}{\sigma_i}v_i = \sigma_i v_i.$$

Pentru $i \leq r$, aplicarea succesivă a lui A^T şi A dă relaţiile de la punctul 2. Pentru $i > r$, avem deja $Av_i = 0$, deci $A^TAv_i = 0$. În plus, vectorii $u_1, \ldots, u_r$ sunt r vectori ortonormali din $\mathcal{R}(A)$; deoarece $\dim \mathcal{R}(A) = r$, ei formează o bază ortonormală a imaginii. Prin construcţie, vectorii $u_{r+1}, \ldots, u_m$ sunt ortogonali pe $\mathcal{R}(A)$, deci aparţin lui $\ker A^T$. Astfel $AA^Tu_i = 0$ pentru $i > r$, ceea ce completează demonstraţia punctului 2 pentru toţi indicii. Din $A = U\Sigma V^T$ şi din faptul că numai primele r elemente diagonale ale lui Σ sunt nenule rezultă direct dezvoltarea de la punctul 3.

Pentru punctul 4 scriem $x = Vy$, astfel încât $\|x\|_2 = \|y\|_2$. Atunci

$$\|Ax\|_2^2 = \|\Sigma y\|_2^2 = \sum_{i=1}^q \sigma_i^2 y_i^2 \leq \sigma_1^2 \sum_{i=1}^n y_i^2 = \sigma_1^2\|x\|_2^2.$$

Dacă $m \geq n$ şi A are rang coloană complet, atunci $r = n$ şi $\sigma_n > 0$, deci

$$\|Ax\|_2^2 = \sum_{i=1}^n \sigma_i^2 y_i^2 \geq \sigma_n^2\|y\|_2^2.$$

În fine, $Ax = 0$ este echivalent cu $\Sigma V^T x = 0$, deci $V^T x$ poate avea componente numai pe direcţiile corespunzătoare lui $v_{r+1}, \dots, v_n$; aceasta dă formula pentru nucleu. Formula pentru imagine rezultă din dezvoltarea de rang r de la punctul 3. □

Interpretarea geometrică

Matricea V^T schimbă baza în direcţiile singulare drepte, Σ întinde sau contractă fiecare direcţie cu factorul σ_i, iar U roteşte sau reflectă rezultatul în spaţiul de sosire. Valorile singulare mici indică direcţii pe care A le comprimă puternic şi care devin sensibile la perturbaţii.

Două exemple numerice de matrici dreptunghiulare

Păstrăm exemplele numerice ale cursului pentru a vedea forma factorilor în cele două orientări ale aceleiaşi matrici.

Fie

$$A = \begin{pmatrix} 1 & 3 \\ 5 & 7 \\ 9 & 1 \end{pmatrix} \in \mathbb{R}^{3\times 2}.$$

O descompunere SVD, rotunjită la şase zecimale, este

$$A = \underbrace{\begin{pmatrix} 0.207621 & 0.370412 & 0.905366 \\ 0.679634 & 0.611049 & -0.405854 \\ 0.703556 & -0.699581 & 0.124878 \end{pmatrix}}_{U\in\mathbb{R}^{3\times 3}} \underbrace{\begin{pmatrix} 11.6522 & 0 \\ 0 & 5.4979 \\ 0 & 0 \end{pmatrix}}_{\Sigma\in\mathbb{R}^{3\times 2}} \underbrace{\begin{pmatrix} 0.852871 & 0.522122 \\ -0.522122 & 0.852871 \end{pmatrix}}_{V^T\in\mathbb{R}^{2\times 2}}.$$

Valorile singulare sunt $\sigma_1 \approx 11.6522$ şi $\sigma_2 \approx 5.4979$, deci rangul este 2.

Pentru transpusa sa,

$$A^T = \begin{pmatrix} 1 & 5 & 9 \\ 3 & 7 & 1 \end{pmatrix} \in \mathbb{R}^{2\times 3},$$

rolurile vectorilor singulari stângi şi drepţi se inversează. O SVD compatibilă este

$$A^T = \underbrace{\begin{pmatrix} 0.852871 & -0.522122 \\ 0.522122 & 0.852871 \end{pmatrix}}_{U\in\mathbb{R}^{2\times 2}} \underbrace{\begin{pmatrix} 11.6522 & 0 & 0 \\ 0 & 5.4979 & 0 \end{pmatrix}}_{\Sigma\in\mathbb{R}^{2\times 3}} \underbrace{\begin{pmatrix} 0.207621 & 0.370412 & 0.905366 \\ 0.679634 & 0.611049 & -0.405854 \\ 0.703556 & -0.699581 & 0.124878 \end{pmatrix}}_{V^T\in\mathbb{R}^{3\times 3}}.$$

Aceste două exemple arată şi faptul că matricile A şi A^T au aceleaşi valori singulare nenule.

Exemplul 1: matrice cu mai multe linii decât coloane ($m > n$)

Luăm

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{3\times 2}.$$

Calculăm

$$A^T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}, \quad A^T A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}.$$

În ordinea descrescătoare, valorile proprii ale lui $A^T A$ sunt $\lambda_1 = 4$ şi $\lambda_2 = 1$, deci valorile singulare sunt

$$\sigma_1 = 2, \qquad \sigma_2 = 1.$$

Pentru $\lambda_1 = 4$ luăm $v_1 = (0,1)^T$, iar pentru $\lambda_2 = 1$ luăm $v_2 = (1,0)^T$. Astfel

$$V = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Vectorii singulari stângi sunt

$$u_1 = \frac{1}{2} A v_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \qquad u_2 = A v_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Completăm cu $u_3 = (0,0,1)^T$, deci

$$U = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Verificarea este imediată:

$$U \Sigma V^T = \begin{pmatrix} 1 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix} = A.$$

Observaţi că ordonarea descrescătoare a valorilor singulare determină şi ordinea coloanelor lui U şi V.

Exemplul 2: matrice cu mai puţine linii decât coloane ($m < n$)

Considerăm

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix} \in \mathbb{R}^{2\times 3}.$$

Atunci $A^T A = \operatorname{diag}(1,4,0)$. În ordinea descrescătoare avem $\lambda_1 = 4$, $\lambda_2 = 1$, $\lambda_3 = 0$ şi, cum $q = \min(2,3) = 2$, valorile singulare sunt $\sigma_1 = 2$, $\sigma_2 = 1$.

Putem lua $v_1 = e_2$, $v_2 = e_1$, $v_3 = e_3$, deci

$$V = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Mai departe,

$$u_1 = \frac{1}{2} A v_1 = e_2, \qquad u_2 = A v_2 = e_1,$$

şi prin urmare

$$U = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \Sigma = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Se verifică direct că $A = U \Sigma V^T$. Vectorul $v_3 = e_3$ generează nucleul lui A şi corespunde valorii proprii zero a lui $A^T A$; el nu produce o valoare singulară suplimentară în diagonala dreptunghiulară, deoarece $q = 2$.

Aceste două exemple simple arată explicit cum dimensiunile factorilor depind de raportul dintre m şi n.

Pentru MATLAB. Comanda

```
[U,S,V] = svd(A);
```

calculează SVD completă. Pentru forma economică se recomandă

```
[U,S,V] = svd(A,'econ');
```

(forma mai veche `svd(A,0)` are acelaşi rol în situaţiile uzuale). Dacă $r_0 = \min(m,n)$, forma economică are $U \in \mathbb{R}^{m\times r_0}$; dimensiunile lui S şi V depind de cazul $m \geq n$ sau $m < n$ şi sunt alese astfel încât $USV^T = A$. În laborator este preferabil să verificăm dimensiunile cu `size(U)`, `size(S)`, `size(V)` în loc să memorăm o formulă unică, incorectă pentru unul dintre cazurile dreptunghiulare.

În ceea ce priveşte rangul, dacă $q = \min(m,n)$ şi

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_q = 0,$$

atunci $\operatorname{rank} A = r$, $\ker A = \operatorname{span}\{v_{r+1}, \ldots, v_n\}$, iar $\operatorname{range} A = \operatorname{span}\{u_1, \ldots, u_r\}$.

Definiţie 11.0.4. *Fie $A \in \mathbb{R}^{m\times n}$ de rang r şi $A = U\Sigma V^T$ o SVD. Pseudoinversa Moore–Penrose este*

$$A^\dagger = V\Sigma^\dagger U^T,$$

unde $\Sigma^\dagger \in \mathbb{R}^{n\times m}$ se obţine transpunând forma diagonală dreptunghiulară şi înlocuind fiecare valoare singulară nenulă σ_i cu $1/\sigma_i$; toate celelalte elemente sunt zero.

Matricea $A^\dagger$ este numită şi inversa generalizată a lui A. Dacă $\operatorname{rank} A = n \leq m$, atunci $A^\dagger = (A^TA)^{-1}A^T$, iar în cazul pătratic şi inversabil $A^\dagger = A^{-1}$.

Revenind la sisteme

Dacă A nu are rang complet, tehnicile de soluţionare prin descompunerea QR de mai sus nu funcţionează şi avem nevoie de o altă tehnică

Teoremă 11.0.5. *Să considerăm $A \in \mathbb{R}^{m\times n}$ cu SVD dat de $A = U\Sigma V^T$. Atunci soluţia unică a problemei de minimizare*

$$\begin{aligned} &\textit{găseşte } x^* \in \mathbb{R}^n \textit{ cu normă euclidiană minimă printre toţi minimizatorii lui } \Phi, \\ &\textit{unde } \Phi(x) = \|Ax - b\|_2^2, \qquad \Phi(x^*) = \min_{x\in\mathbb{R}^n} \Phi(x) \end{aligned} \tag{11.1}$$

este

$$x^* = A^\dagger b, \tag{11.2}$$

unde $A^\dagger$ este pseudoinversa lui A.

Demonstraţie. Folosind SVD-ul lui A, introducem schimbarea de variabilă $w = V^Tx$. Deoarece V este ortogonală, avem $\|w\|_2 = \|x\|_2$. Problema devine astfel determinarea unui vector w cu normă euclidiană minimă pentru care

$$\|\Sigma w - U^Tb\|_2^2 \leq \|\Sigma y - U^Tb\|_2^2 \qquad \forall\ y \in \mathbb{R}^n. \tag{11.3}$$

Dacă r este numărul de valori singulare nenule σ_i ale lui A, atunci

$$\|\Sigma w - U^Tb\|_2^2 = \sum_{i=1}^{r} \left(\sigma_i w_i - (U^Tb)_i\right)^2 + \sum_{i=r+1}^{m} \left((U^Tb)_i\right)^2, \tag{11.4}$$

ceea ce este minim dacă $w_i = (U^Tb)_i/\sigma_i$ pentru $i = 1, \ldots, r$.

În plus, este clar că printre vectorii w ai lui $\mathbb{R}^n$ care au primele r componente fixe, vectorul cu normă euclidiană minimă are celelalte $n - r$ componente egale cu zero.

Aşadar, vectorul soluţie este $w^* = \Sigma^\dagger U^Tb$, adică $x^* = V\Sigma^\dagger U^Tb = A^\dagger b$, unde $\Sigma^\dagger$ este matricea diagonală definită în definiţia pseudo-inversei. □

Extensie didactică: geometria şi puterea descompunerii SVD

Pentru orice matrice $A \in \mathbb{R}^{m\times n}$ există o descompunere $A = U\Sigma V^T$, unde U şi V sunt ortogonale, iar Σ conţine valorile singulare $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$. Spre deosebire de descompunerea spectrală, SVD există pentru orice matrice dreptunghiulară şi nu cere simetrie.

Imagine geometrică

Transformarea $x \mapsto Ax$ poate fi privită în trei etape: V^T roteşte sau reflectă domeniul, Σ întinde sau contractă pe direcţii ortogonale, iar U roteşte sau reflectă rezultatul. Sfera unitate este transformată într-un elipsoid ale cărui semiaxe au lungimile σ_i. Valorile singulare mici identifică direcţii aproape pierdute de transformare.

Rangul este numărul valorilor singulare nenule. Pentru o matrice cu rang complet relevant, condiţionarea spectrală este $\kappa_2(A) = \sigma_{\max}/\sigma_{\min}$. O valoare singulară foarte mică înseamnă că există o direcţie v de normă unu pentru care Av este foarte mic; de aceea recuperarea componentei lui x în acea direcţie din date zgomotoase este sensibilă.

Pseudoinversa

Dacă $A = U\Sigma V^T$, pseudoinversa Moore–Penrose este $A^+ = V\Sigma^+U^T$, unde fiecare valoare singulară nenulă este înlocuită cu inversa sa. Pentru un sistem incompatibil, $x^+ = A^+b$ este o soluţie LS; dintre toate soluţiile LS, este cea de normă euclidiană minimă. Pentru un sistem compatibil subdeterminat, selectează soluţia exactă de normă minimă.

Această formulă trebuie citită şi ca avertisment: componentele datelor pe direcţiile u_i sunt împărţite la σ_i. Dacă σ_i este foarte mic, zgomotul pe acea componentă este amplificat. Regularizarea poate fi interpretată spectral tocmai ca o limitare a acestei amplificări.

Exemplu diagonal

Pentru $A = \operatorname{diag}(3, 10^{-4})$, SVD este deja vizibilă: valorile singulare sunt 3 şi 10^{-4}. Dacă $b = (3, 10^{-4})^T$, soluţia este $(1, 1)^T$. O perturbare de numai 10^{-4} pe a doua componentă a lui b poate modifica a doua componentă a soluţiei cu ordinul unu. Geometria elipsei şi sensibilitatea algebrică spun acelaşi lucru.

Cea mai bună aproximaţie de rang mic

SVD oferă şi o aproximare optimă. Dacă $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$, atunci A_k este cea mai bună aproximaţie de rang cel mult k atât în norma spectrală, cât şi în norma Frobenius. Erorile sunt $\|A - A_k\|_2 = \sigma_{k+1}$ şi

$$\|A - A_k\|_F^2 = \sum_{i>k} \sigma_i^2.$$

Acesta este fundamentul matematic al compresiei imaginilor discutate în capitolul următor.

Laborator: SVD ca instrument de diagnostic

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru matrici cu rang complet, aproape deficient şi exact deficient, reprezentaţi valorile singulare pe scară semilogaritmică. Calculaţi `rank` cu toleranţe diferite şi observaţi că “rangul numeric” depinde de scara valorilor singulare şi de precizia aritmetică. Comparaţi `pinv(A)*b` cu o soluţie obţinută prin ecuaţii normale.

Exerciţii şi verificarea înţelegerii

1. Arătaţi că valorile singulare pătrate sunt valorile proprii ale lui A^TA.
2. Demonstraţi că $\|A\|_2 = \sigma_1$.
3. Pentru $A = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}$, calculaţi pseudoinversa şi soluţiile lui $Ax = b$ pentru mai multe alegeri ale lui b.
4. Arătaţi că A^+b este ortogonală pe nucleul lui A.
5. Explicaţi de ce SVD este mai costisitoare decât unele factorizări specializate, dar oferă mai multă informaţie despre rang şi sensibilitate.
6. Derivaţi formula numărului de parametri necesari pentru stocarea unei aproximaţii de rang k: aproximativ $k(m+n+1)$ în loc de mn.

Mini-proiect: rang numeric şi zgomot

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi o matrice de rang exact r ca produs BC^T, apoi adăugaţi zgomot de nivel ε. Reprezentaţi spectrul singular pentru mai multe valori ale lui ε şi propuneţi o regulă de separare între semnal şi zgomot. Reconstruiţi matricea prin truncarea SVD şi comparaţi cu matricea exactă iniţială.

Exemplu rezolvat: SVD, pseudoinversă şi truncare

Considerăm

$$A = \begin{pmatrix} 3 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Coloanele sunt deja ortogonale, astfel încât o SVD este $A = U\Sigma V^T$ cu $V = I_2$, primele două coloane ale lui U egale cu e_1, e_2 şi valori singulare $\sigma_1 = 3$, $\sigma_2 = 1$. Pseudoinversa este

$$A^\dagger = \begin{pmatrix} 1/3 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Pentru $b = (6, 2, 5)^T$, soluţia LS de normă minimă este $x = A^\dagger b = (2, 2)^T$. Componenta $5e_3$ a lui b este ortogonală pe imaginea lui A şi nu poate fi reprodusă; ea devine exact reziduul.

Dacă păstrăm numai prima valoare singulară, cea mai bună aproximaţie de rang 1 este $A_1 = 3e_1^{(3)}(e_1^{(2)})^T$, unde indicii superiori precizează dimensiunile vectorilor canonici; adică păstrăm numai prima coloană efectivă. Eroarea spectrală este $\sigma_2 = 1$, iar eroarea Frobenius este de asemenea 1 în acest exemplu. Aceasta ilustrează teorema Eckart–Young.

SVD separă geometria problemei

Vectorii v_i sunt direcţii în spaţiul necunoscutelor, valorile σ_i măsoară cât sunt întinse acele direcţii de A, iar u_i indică direcţiile rezultate în spaţiul datelor. O valoare singulară foarte mică înseamnă o direcţie a necunoscutelor aproape invizibilă în date; inversarea ei este tocmai sursa sensibilităţii.

Sarcină practică: spectru singular prescris

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Generaţi două matrici ortogonale U, V şi alegeţi valori singulare $\sigma_i = 10^{-i/2}$. Formaţi $A = U\Sigma V^T$. Verificaţi că `svd(A)` recuperează spectrul şi studiaţi soluţia $A^\dagger b$ când adăugaţi zgomot pe direcţiile u_i asociate valorilor singulare mici. Repetaţi după truncarea valorilor sub un prag şi comparaţi bias-ul cu reducerea zgomotului.

Exerciţii suplimentare

1. Calculaţi proiecţia lui $b = (6, 2, 5)^T$ pe $\mathcal{R}(A)$ şi reziduul.
2. Verificaţi una dintre condiţiile Moore–Penrose, $AA^\dagger A = A$, pentru exemplu.
3. Arătaţi că $\|A\|_2 = \sigma_1$ şi, pentru rang complet, $\|A^\dagger\|_2 = 1/\sigma_r$.
4. Deduceţi $\kappa_2(A) = \sigma_1/\sigma_r$.
5. Pentru o matrice cu rang deficient, explicaţi de ce pseudoinversa selectează soluţia de normă minimă.

Fişă de seminar: SVD şi pseudoinversa

Problema rezolvată 1: soluţia de normă minimă într-un sistem subdeterminat

Considerăm $A = (1\ 1) \in \mathbb{R}^{1\times 2}$ şi ecuaţia $x_1 + x_2 = 2$. Există infinit de multe soluţii $(t, 2 - t)$. Norma pătratică este $t^2 + (2 - t)^2 = 2(t - 1)^2 + 2$, minimă pentru $t = 1$. Soluţia de normă minimă este deci $(1, 1)^T$.

SVD oferă acelaşi rezultat sistematic. Valoarea singulară este $\sqrt{2}$, vectorul singular drept asociat este $(1,1)^T/\sqrt{2}$, iar pseudoinversa este $A^\dagger = (1/2, 1/2)^T$. Prin urmare $A^\dagger 2 = (1,1)^T$. Componenta unei soluţii din nucleul lui A, generat de $(1,-1)^T$, măreşte norma fără a schimba datele.

Problema rezolvată 2: rang numeric

În aritmetică exactă, o valoare singulară este fie zero, fie nenulă. În date măsurate şi calcule finite, valori precum 10^{-14} lângă o valoare dominantă 1 pot fi indistincte de zgomot. Rangul numeric depinde de scară, precizie şi toleranţa relevantă pentru aplicaţie. De aceea rutinele folosesc praguri de forma $\tau \approx \max(m,n)u_{mach}\sigma_1$ sau praguri definite de utilizator.

Laborator integrator 11: pseudoinversă şi rang

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi matrici cu spectru singular $1, 10^{-2}, 10^{-4}, \dots$ şi adăugaţi perturbări de nivel controlat. Comparaţi `rank(A)` pentru toleranţe diferite, soluţia pseudoinversă completă şi soluţia cu truncare. Reprezentaţi norma soluţiei şi reziduul în funcţie de prag. Identificaţi zona în care păstrarea valorilor foarte mici creşte puternic norma fără a reduce semnificativ reziduul.

Set de probleme

1. Calculaţi pseudoinversa matricei $A = (1\ 1)$ şi verificaţi cele patru condiţii Moore–Penrose.
2. Pentru $A = \mathrm{diag}(3,1,0)$ calculaţi pseudoinversa şi proiecţiile $AA^\dagger$, $A^\dagger A$.
3. Demonstraţi că $AA^\dagger$ este proiecţia ortogonală pe imaginea lui A.
4. Demonstraţi că $A^\dagger A$ este proiecţia ortogonală pe $\mathcal{R}(A^T)$.
5. Cum se citeşte rangul din SVD?
6. De ce condiţionarea unei matrici dreptunghiulare cu rang complet poate fi definită prin raportul valorilor singulare extreme?
7. Formulaţi truncated SVD ca filtru spectral.
8. Comparaţi conceptual pseudoinversa, Tikhonov şi QR cu pivotare pentru o problemă aproape deficientă de rang.

Proiect de capitol: identificarea rangului latent

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Generaţi date $A_0 = UV^T$ de rang r şi adăugaţi zgomot de amplitudini diferite. Calculaţi spectrul singular şi încercaţi să recuperaţi r printr-un prag. Repetaţi pentru mai multe raporturi semnal/zgomot. Prezentaţi grafic când ruptura spectrală este clară şi când noţiunea de rang devine ambiguă.

Sarcini de aprofundare şi sinteză pentru SVD şi pseudoinversă

Probleme de aprofundare: rang numeric, aproximare şi sensibilitate

1. Calculaţi manual o SVD pentru o matrice 2×2 simplă şi verificaţi relaţiile $A^T A v_i = \sigma_i^2 v_i$ şi $u_i = Av_i/\sigma_i$ pentru valorile singulare nenule.
2. Demonstraţi că $\|A\|_2 = \sigma_1$ şi, dacă A este inversabilă, $\|A^{-1}\|_2 = 1/\sigma_n$. Deduceţi $\kappa_2(A) = \sigma_1/\sigma_n$.
3. Arătaţi că multiplicarea lui A la stânga sau la dreapta cu o matrice ortogonală nu modifică valorile singulare.
4. Pentru o matrice simetrică reală, demonstraţi că valorile singulare sunt modulele valorilor proprii. Construiţi un exemplu nesimetric unde această relaţie eşuează.
5. Demonstraţi că soluţia de normă minimă a unei probleme LS se obţine prin pseudoinversă. Explicaţi ce se întâmplă pe nucleul lui A.
6. Pentru o matrice de rang r perturbată cu zgomot mic, explicaţi de ce rangul algebric devine adesea complet, dar rangul numeric poate rămâne r. Propuneţi un criteriu bazat pe spectrul singular.
7. Folosind teorema Eckart–Young–Mirsky, calculaţi eroarea optimă Frobenius a aproximaţiei de rang k şi explicaţi de ce nu este necesar să căutăm printre toate matricile de rang k.
8. Construiţi două matrici cu acelaşi rang, dar spectre singulare foarte diferite. Care este mai sensibilă la perturbări şi de ce?
9. Pentru $A = U\Sigma V^T$, descrieţi geometric efectul asupra sferei unitate. Ce rol au V, Σ şi U?
10. Analizaţi efectul unui prag τ în pseudoinversa trunchiată: ce se câştigă şi ce bias se introduce când valorile singulare sub τ sunt eliminate?
11. Demonstraţi relaţia $A^+ = V\Sigma^+ U^T$ şi verificaţi două dintre condiţiile Penrose pe un exemplu numeric.
12. Explicaţi diferenţa dintre compresie, denoising şi regularizare atunci când toate sunt implementate prin eliminarea sau atenuarea direcţiilor singulare mici.

Laborator integrator: rang latent sub zgomot

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Reconstruiţi o matrice de rang mic din observaţii zgomotoase şi decideţi rangul numeric din date.

1. Construiţi $A_0 = BC^T$ cu $B \in \mathbb{R}^{m\times r}$ şi $C \in \mathbb{R}^{n\times r}$ pentru $r \ll \min(m, n)$.
2. Adăugaţi zgomot gaussian cu mai multe amplitudini şi calculaţi spectrul singular al lui $A = A_0 + E$.
3. Reprezentaţi valorile singulare pe scară logaritmică şi identificaţi vizual tranziţia dintre semnal şi zgomot.
4. Pentru $k = 1, \ldots, 2r+5$, construiţi aproximaţia trunchiată A_k şi calculaţi $\|A - A_k\|_F$ şi $\|A_0 - A_k\|_F$.
5. Comparaţi rangul care minimizează eroarea faţă de A_0 cu rangul sugerat de spectrul singular. Explicaţi de ce în date reale prima mărime nu este disponibilă.
6. Verificaţi numeric formula teoretică a erorii Frobenius prin suma pătratelor valorilor singulare eliminate.
7. Introduceţi un prag pentru pseudoinversă şi rezolvaţi o problemă LS construită cu aceeaşi matrice. Studiaţi compromisul dintre reziduu şi norma soluţiei.
8. Formulaţi o regulă practică de alegere a rangului sau pragului şi precizaţi limitele ei.

Proiect de capitol: descoperirea unei structuri de rang mic

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Construiţi sau alegeţi un set de date matricial în care este plauzibilă existenţa câtorva factori latenţi: măsurători repetate, preferinţe sintetice, semnale sau cadre de imagine.

1. **Modelare.** Explicaţi de ce o aproximare de rang mic este rezonabilă în contextul ales.
2. **Preprocesare.** Decideţi dacă datele trebuie centrate sau scalate şi justificaţi alegerea.
3. **SVD.** Calculaţi spectrul singular şi procentul de energie captat de primele k componente.
4. **Alegerea rangului.** Comparaţi cel puţin două criterii: prag relativ şi energie cumulată.
5. **Reconstrucţie.** Comparaţi erorile pentru mai multe valori ale lui k şi vizualizaţi sau interpretaţi reconstrucţia.
6. **Zgomot.** Perturbaţi datele şi studiaţi stabilitatea subspaţiilor dominante sau a reconstrucţiei.
7. **Cost.** Comparaţi numărul de valori necesare pentru stocarea matricei complete cu stocarea factorilor trunchiaţi.
8. **Interpretare.** Explicaţi ce par să reprezinte primele direcţii singulare în aplicaţia aleasă.

Test de capitol

Test – SVD şi pseudoinversa

1. Calcul explicit. Pentru

$$A = \begin{pmatrix} 3 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

construiţi o descompunere SVD redusă. Determinaţi rangul, valorile singulare şi pseudoinversa A^+. Folosiţi A^+ pentru a determina soluţia de normă minimă a problemei celor mai mici pătrate pentru $b = (3, 2, 1)^T$.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi.

A) valorile singulare sunt nenegative;

B) pseudoinversa este definită şi pentru matrici dreptunghiulare sau de rang redus;

C) A^+b furnizează soluţia de normă minimă dintre soluţiile de cele mai mici pătrate;

D) valorile singulare ale unei matrice sunt întotdeauna identice cu valorile sale proprii.

Capitolul 12

Aplicaţii SVD în compresia imaginilor

12.1 Descompunerea în valori singulare (SVD) şi tensorii de ordinul al doilea

Descompunerea în valori singulare (SVD)

Descompunerea în valori singulare (SVD) a unei matrice A nu este utilă doar pentru problema celor mai mici pătrate, ci are numeroase aplicaţii practice.

În continuare vom indica câteva metode simple şi directe de aplicare în comprimarea imaginilor.

Ne aducem aminte.

Reamintire: descompunerea în valori singulare

Pentru orice $A \in \mathbb{R}^{m\times n}$, cu $q = \min(m, n)$, există matrice ortogonale U, V şi o matrice diagonală dreptunghiulară Σ cu valori singulare $\sigma_1 \geq \cdots \geq \sigma_q \geq 0$ astfel încât $A = U\Sigma V^T$. Demonstraţia completă a fost dată în capitolul precedent.

$$A = U\Sigma V^\top$$

Definiţie 12.1.1. *Vectorii coloană ai lui $U = [u_1, \ldots, u_m]$ sunt numiţi vectori singulari stângi şi, similar, $V = [v_1, \ldots, v_n]$ sunt vectorii singulari drepţi. Valorile $\sigma_i = \sqrt{\lambda_i(A^\top A)}$ sunt numite valorile singulare ale lui A (unde $\lambda_i(A^\top A)$ sunt valorile proprii ale lui $A^\top A$).*

În ceea ce priveşte rangul, dacă $q = \min(m, n)$ şi

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_q = 0,$$

atunci $\operatorname{rank} A = r$, $\ker A = \operatorname{span}\{v_{r+1}, \ldots, v_n\}$ şi $\operatorname{range} A = \operatorname{span}\{u_1, \ldots, u_r\}$.

Matricea A se poate scrie ca o sumă de matrice

$$A = \sum_{i=1}^{r} \sigma_i u_i\, v_i^T = \sigma_1 u_1\, v_1^T + \sigma_2 u_2\, v_2^T + \ldots + \sigma_r u_r\, v_r^T + 0 + 0 + \ldots + 0.$$

Tensor de ordin 2

Vom folosi în continuare noţiunea de tensor de ordin 2.

Prin produsul tensorial al doi vectori $\overline{\xi} \in \mathbb{R}^n$, $\overline{\eta} \in \mathbb{R}^m$, notat $\overline{\xi} \otimes \overline{\eta} \in \mathbb{R}^{n\times m}$, înţelegem un operator liniar

$$\overline{\xi} \otimes \overline{\eta} : \mathbb{R}^m \to \mathbb{R}^n, \quad (\overline{\xi} \otimes \overline{\eta})\,.\overline{v} = \langle \overline{\eta}, \overline{v} \rangle\, \overline{\xi}\,. \tag{12.1}$$

Fiind un operator liniar, un tensor este un obiect matematic care poate fi caracterizat prin elementele unei baze din domeniu şi a unei baze din codomeniu.

Prin urmare în baze diferite, vom avea caracterizări diferite.

Fie $(\overline{e}_i)_{i=1}^n$ o bază ortonormată a lui $\mathbb{R}^n$ şi $(\overline{f}_j)_{j=1}^m$ o bază ortonormată a lui $\mathbb{R}^m$. Pentru vectorii $\overline{\xi} = \sum_{i=1}^n \xi_i \overline{e}_i$ şi $\overline{\eta} = \sum_{j=1}^m \eta_j \overline{f}_j$ avem

$$\overline{\xi} \otimes \overline{\eta} = \sum_{i=1}^n \sum_{j=1}^m \xi_i \eta_j\, \overline{e}_i \otimes \overline{f}_j.$$

Prin urmare familia

$$\{\overline{e}_i \otimes \overline{f}_j : \ 1 \le i \le n,\ 1 \le j \le m\}$$

este o bază a spaţiului tensorial $\mathbb{R}^n \otimes \mathbb{R}^m$, pe care în dimensiune finită îl identificăm cu spaţiul operatorilor liniari $\mathcal{L}(\mathbb{R}^m, \mathbb{R}^n)$ şi, după alegerea bazelor, cu $\mathbb{R}^{n\times m}$.

O distincţie utilă

Mulţimea produselor tensoriale simple $\{\xi \otimes \eta\}$ nu este, în general, un subspaţiu vectorial: suma a două matrice de rang unu poate avea rang doi. Spaţiul tensorial este format din combinaţii liniare finite de astfel de produse. Această distincţie va fi folosită când interpretăm SVD ca sumă de termeni de rang unu.

Tensor de ordin 2

Fie o bază ortonormată $(\overline{e}_i)_{i=1}^n$ în $\mathbb{R}^n$ şi o bază ortonormată $(\overline{f}_j)_{j=1}^m$ în $\mathbb{R}^m$.

Pentru cele necesare nouă, prin tensor de ordin 2 vom înţelege un obiect matematic I definit prin

$$I = \sum_{i=1}^n \sum_{j=1}^m I_{ij}\, (\overline{e}_i \otimes \overline{f}_j),$$

iar I_{ij} se vor numi componentele tensorului I relativ la bazele $\overline{e}_i$ şi $\overline{f}_j$.

Tensorul de ordin 2 se poate identifica cu matricea sa într-o pereche de baze. ATENŢIE! Dacă schimbăm una dintre baze, atunci matricea cu care se identifică tensorul se SCHIMBĂ după regulile pentru operatori liniari!

Considerând doi vectori $\overline{\xi} = \sum_{i=1}^n \xi_i \overline{e}_i \in \mathbb{R}^n$, $\overline{\eta} = \sum_{j=1}^m \eta_j \overline{f}_j \in \mathbb{R}^m$ *produsul tensorial* al lor este complet definit de matricea operatorului liniar.

Deoarece

$$\overline{\xi} \otimes \overline{\eta} = \sum_{i=1}^n \sum_{j=1}^m \xi_i \eta_j \overline{e}_i \otimes \overline{f}_j$$

şi

$$(\overline{\xi} \otimes \overline{\eta}).\overline{f}_k = \sum_{i=1}^n \sum_{j=1}^m \xi_i \eta_j (\overline{e}_i \otimes \overline{f}_j).\overline{f}_k = \sum_{i=1}^n \sum_{j=1}^m \xi_i \eta_j \langle \overline{f}_j, \overline{f}_k \rangle \overline{e}_i = \sum_{i=1}^n \xi_i \eta_k \overline{e}_i$$

matricea operatorului în bazele considerate are componentele

$$(\overline{\xi} \otimes \overline{\eta})_{ij} = \xi_i\, \eta_j.$$

Dar $\xi_i\,\eta_j$ sunt chiar componentele matricei produs $\overline{\xi}\cdot\overline{\eta}^T$.

Avem

$$(\overline{\xi}\otimes\overline{\eta})_{ij}=\xi_i\,\eta_j.$$

Dar $\xi_i\,\eta_j$ sunt chiar componentele matricei produs $\overline{\xi}\cdot\overline{\eta}^T$.

Să ne întoarcem acum la descompunerea în valori singulare şi la scrierea matricei A ca

$$A=\sum_{i=1}^{r}\sigma_i u_i\,v_i^T=\sigma_1u_1\,v_1^T+\sigma_2u_2\,v_2^T+\ldots+\sigma_ru_r\,v_r^T+0+0+\ldots+0.$$

Folosind terminologia de la tensori, putem scrie

Scrierea matricei ca sumă de tensori

Prin urmare, identificând produsul tensorial cu matricea sa în baza canonică, avem

$$A=\sum_{i=1}^{r}\sigma_iu_i\otimes v_i=\sigma_1u_1\otimes v_1+\sigma_2u_2\otimes v_2+\ldots+\sigma_ru_r\otimes v_r.$$

Mai mult, deoarece u_i şi v_i sunt vectori ortonormaţi în $\mathbb{R}^m$, respectiv $\mathbb{R}^n$, putem spune că scriere

$$A=\sum_{i=1}^{r}\sigma_iu_i\otimes v_i$$

ne oferă o descompunere a matricei A în spaţiul tensorilor de ordin 2.

Dacă $\overline{\xi}\neq 0$ şi $\overline{\eta}\neq 0$, matricea asociată produsului tensorial $\overline{\xi}\otimes\overline{\eta}$ are rangul 1; dacă unul dintre vectori este zero, rangul este 0.

Deci, în scrierea

$$A=\sum_{i=1}^{r}\sigma_iu_i\otimes v_i=\sigma_1u_1\otimes v_1+\sigma_2u_2\otimes v_2+\ldots+\sigma_ru_r\otimes v_r$$

avem o descompunere a lui A ca sumă de termeni de rang unu, adaptaţi matricii prin vectorii săi singulari.

Mai mult, pentru orice $1\leq k\leq r$ avem

$$\operatorname{rank}(\sigma_1u_1\otimes v_1+\sigma_2u_2\otimes v_2+\ldots+\sigma_ku_k\otimes v_k)=k.$$

Ne întrebăm dacă nu putem **COMPRIMA** matricea A folosind doar o parte din termenii sumei. POATE DOAR PÂNĂ CÂND σ_i începe să nu mai conteze. Ce înseamnă oare asta?

Deoarece am scris valorile singulare ordonate descrescător, aceasta ar însemna că se aproximează matricea A cu o matrice de rang k.

Dar care o fi cea mai bună aproximare a sa?

Teoremă 12.1.2 (Eckart–Young–Mirsky). *Fie $A\in\mathbb{R}^{m\times n}$ de rang r, cu SVD*

$$A=\sum_{i=1}^{r}\sigma_iu_iv_i^T,\qquad \sigma_1\geq\cdots\geq\sigma_r>0.$$

Pentru $0\leq k<r$ definim

$$A_k=\sum_{i=1}^{k}\sigma_iu_iv_i^T.$$

Atunci $\operatorname{rank} A_k = k$ *şi, pentru orice matrice* B *cu* $\operatorname{rank} B \le k$,

$$\|A - B\|_2 \ge \|A - A_k\|_2 = \sigma_{k+1},$$

iar pentru norma Frobenius

$$\|A - B\|_F^2 \ge \|A - A_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2.$$

Prin urmare A_k *este o cea mai bună aproximaţie de rang cel mult* k *atât în norma spectrală, cât şi în norma Frobenius.*

Demonstraţie. Deoarece u_i şi v_i sunt familii ortonormale, A_k are exact valorile singulare $\sigma_1, \ldots, \sigma_k$, deci are rangul k. În plus,

$$A - A_k = \sum_{i=k+1}^{r} \sigma_i u_i v_i^T,$$

de unde $\|A - A_k\|_2 = \sigma_{k+1}$ şi, prin ortogonalitatea termenilor în produsul scalar Frobenius,

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2.$$

Demonstrăm mai întâi optimalitatea în norma spectrală. Fie B de rang cel mult k. Atunci $\dim \ker B \ge n - k$. Subspaţiul $E = \operatorname{span}\{v_1, \ldots, v_{k+1}\}$ are dimensiunea $k + 1$, deci

$$\dim(\ker B) + \dim E \ge n + 1,$$

şi există un vector unitar $h \in \ker B \cap E$. Scriem $h = \sum_{i=1}^{k+1} c_i v_i$, cu $\sum |c_i|^2 = 1$. Deoarece $Bh = 0$,

$$\begin{aligned} \|A - B\|_2^2 &\ge \|(A - B)h\|_2^2 = \|Ah\|_2^2 \\ &= \sum_{i=1}^{k+1} \sigma_i^2 |c_i|^2 \ge \sigma_{k+1}^2 \sum_{i=1}^{k+1} |c_i|^2 = \sigma_{k+1}^2. \end{aligned}$$

Pentru norma Frobenius, notăm cu $S = \operatorname{range} B$ şi cu P_S proiecţia ortogonală pe S. Deoarece coloanele lui B aparţin lui S,

$$(I - P_S)B = 0,$$

iar contractivitatea proiecţiei ortogonale dă

$$\|A - B\|_F^2 \ge \|(I - P_S)(A - B)\|_F^2 = \|(I - P_S)A\|_F^2.$$

Putem presupune $\dim S = k$, completând subspaţiul dacă este necesar. Fie $w_1, \ldots, w_k$ o bază ortonormală a lui S. Atunci

$$\begin{aligned} \|P_S A\|_F^2 &= \sum_{j=1}^{k} \|A^T w_j\|_2^2 = \sum_{j=1}^{k} \sum_{i=1}^{r} \sigma_i^2 |u_i^T w_j|^2 \\ &= \sum_{i=1}^{r} \sigma_i^2 p_i, \qquad p_i := \sum_{j=1}^{k} |u_i^T w_j|^2. \end{aligned}$$

Avem $0 \le p_i \le 1$ şi $\sum_i p_i \le k$. Deoarece σ_i^2 sunt ordonate descrescător, suma ponderată este cel mult $\sum_{i=1}^{k} \sigma_i^2$. Prin urmare

$$\|(I - P_S)A\|_F^2 = \|A\|_F^2 - \|P_S A\|_F^2 \ge \sum_{i=k+1}^{r} \sigma_i^2.$$

Această limită este atinsă pentru $S = \operatorname{span}\{u_1, \ldots, u_k\}$ şi $B = A_k$, ceea ce încheie demonstraţia. □

Pentru a stoca reprezentarea trunchiată A_k sub forma valorilor şi vectorilor singulari sunt necesari idealizat $k(m+n+1)$ coeficienţi, în loc de mn. Reconstrucţia A_k are în continuare dimensiunea $m \times n$; ceea ce se comprimă este reprezentarea, nu dimensiunea imaginii reconstruite. Când $k \ll m, n$, economia de stocare poate fi substanţială.



Figura 12.1: Considerăm această fotografie şi dorim să o comprimăm.

Salvarea acestei fotografii în MATLAB se face sub forma unei matrice.

A comprima imaginea în acest model înseamnă a stoca o reprezentare de rang mic, din care reconstruim ulterior o matrice de aceeaşi dimensiune cu imaginea iniţială.

Completare didactică: o imagine în tonuri de gri ca matrice

O imagine în tonuri de gri cu m linii şi n coloane poate fi reprezentată printr-o matrice $A \in \mathbb{R}^{m \times n}$, fiecare element codificând intensitatea unui pixel. Din perspectiva algebrei liniare, o imagine este deci un tensor de ordin doi, iar SVD poate identifica direcţii dominante ale variaţiei globale.

Aproximaţia de rang k este $A_k = U_k \Sigma_k V_k^T$. În loc să stocăm mn valori, stocăm mk valori pentru U_k, k valori singulare şi nk pentru V_k, adică $k(m+n+1)$. Compresia are sens numai dacă această cantitate este semnificativ mai mică decât mn şi dacă spectrul singular scade suficient de repede.

Energie şi eroare

Norma Frobenius satisface $\|A\|_F^2 = \sum_i \sigma_i^2$. De aceea raportul $E_k = \sum_{i=1}^{k} \sigma_i^2 / \sum_i \sigma_i^2$ poate fi interpretat ca fracţiune de "energie" matricială reţinută. Este însă o măsură algebrică, nu o măsură completă a calităţii perceptuale; două imagini cu aceeaşi eroare Frobenius pot fi percepute diferit.

Laborator: primul spectru singular al unei imagini

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Încărcaţi o imagine în tonuri de gri şi calculaţi `svd(A,'econ')`. Reprezentaţi valorile singulare şi energia cumulativă. Reconstruiţi pentru $k = 1, 5, 10, 20, 50, 100$ şi puneţi imaginile alături de raportul de stocare $k(m+n+1)/(mn)$ şi eroarea relativă Frobenius.

Exerciţii şi verificarea înţelegerii

1. Dacă $m = n = 1000$ şi $k = 40$, estimaţi raportul dintre numărul de valori stocate în SVD trunchiată şi numărul de pixeli originali.
2. Arătaţi că $E_k = 1 - \|A - A_k\|_F^2 / \|A\|_F^2$.
3. Ce imagine produce o matrice de rang 1 de forma uv^T? Descrieţi dependenţa liniilor şi coloanelor.
4. De ce o imagine complet aleatoare tinde să fie mai greu de comprimat prin rang mic decât una cu regiuni netede şi structură repetitivă?
5. Explicaţi diferenţa dintre "rang matematic" şi "rang util pentru compresie".

Mini-proiect: alegerea automată a rangului

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Implementaţi trei reguli de alegere a lui k: prag pe energia cumulativă (de exemplu 95%, 99%), prag pe eroarea relativă Frobenius şi buget maxim de stocare. Aplicaţi-le pe imagini foarte diferite şi comparaţi rangurile rezultate şi calitatea vizuală.

Exemplu matricial: o imagine cu rang exact 1

Matricea

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

poate fi scrisă ca produs exterior $A = uv^T$, cu $u = (1, 1, 0)^T$ şi $v = (1, 1, 0)^T$. Prin urmare are rang 1. Singura valoare singulară nenulă este $\|u\|\|v\| = 2$. În limbaj de imagine, toate liniile nenule sunt multipli ale aceleiaşi forme; există o redundanţă perfectă şi reprezentarea de rang 1 este exactă.

Imaginile reale nu au de obicei rang mic exact, dar valorile singulare pot scădea repede. Atunci o aproximaţie de rang k păstrează structurile dominante şi elimină componentele asociate valorilor singulare mici. Din punct de vedere numeric, compresia se bazează pe rang aproximativ, nu pe rang exact.

Sarcină practică: de la matrice sintetică la imagine

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina

studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Construiţi matrici ca sume de r produse exterioare aleatoare şi verificaţi că rangul lor este cel mult r. Adăugaţi zgomot gaussian şi observaţi că rangul numeric devine complet, dar spectrul singular păstrează o ruptură între primele valori şi nivelul zgomotului. Folosiţi această ruptură pentru a propune un rang de reconstrucţie.

Exerciţii suplimentare

1. Calculaţi SVD pentru matricea de mai sus folosind normalizarea lui u şi v.
2. Ce rang are o imagine în care toate coloanele sunt identice?
3. Daţi un exemplu de matrice vizual simplă, dar cu rang mare.
4. Explicaţi diferenţa dintre rang algebric şi rang numeric.
5. Dacă o imagine $m \times n$ este păstrată la rang k, câte numere sunt necesare pentru factorii SVD truncaţi?

12.2 Compresia imaginilor în tonuri de gri

```
A = im2double(imread('cameraman.tif'));   % exemplu grayscale
[U,S,V] = svd(A,'econ');
size(A)
figure(1)
imshow(A), title(sprintf('Initial'));
k = 20;
Ak = U(:,1:k) * S(1:k,1:k) * V(:,1:k)';
Ak = min(max(Ak,0),1);
figure(2)
imshow(Ak), title(sprintf('SVD rank-%d', k));
```

```
k = 40;
Ak = U(:,1:k) * S(1:k,1:k) * V(:,1:k)';
Ak = min(max(Ak,0),1);
figure(3)
imshow(Ak), title(sprintf('SVD rank-%d', k));
k = 100;
Ak = U(:,1:k) * S(1:k,1:k) * V(:,1:k)';
Ak = min(max(Ak,0),1);
figure(4)
imshow(Ak), title(sprintf('SVD rank-%d', k));
k = 200;
Ak = U(:,1:k) * S(1:k,1:k) * V(:,1:k)';
Ak = min(max(Ak,0),1);
figure(5)
```

```
imshow(Ak), title(sprintf('SVD rank-%d', k));
s = svd(A);
cs = cumsum(s.^2) / sum(s.^2);
figure
plot(cs, "LineWidth", 2)
xlabel("k")
ylabel("Energie cumulativa")
title("Energie cumulativa")
```



Figura 12.2: Fotografie iniţială.



Figura 12.3: Fotografie reconstruită



Figura 12.4: Fotografie iniţială.



Figura 12.5: Fotografie reconstruită



Figura 12.6: Fotografie iniţială.

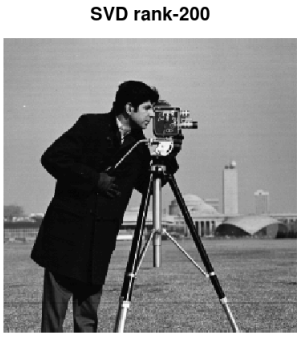


Figura 12.7: Fotografie reconstruită



Figura 12.8: Fotografie iniţială.

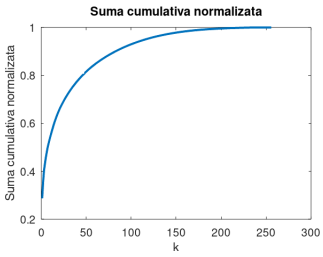


Figura 12.9: Se observă că în jur de valoarea 200 valoarea stagnează, ceea ce însemnă că nu mai există informaţie spectrală semnificativă la care să merite să renunţăm.

Completare didactică: compresia imaginilor în tonuri de gri pas cu pas

Într-o implementare corectă, compresia SVD are patru etape distincte: citirea şi conversia datelor într-un tip numeric potrivit, calculul descompunerii, alegerea rangului şi reconstrucţia. Este util să separăm aceste etape pentru a nu confunda erorile de reprezentare a imaginii cu eroarea aproximării de rang mic.

Alegerea lui k

Un k fix este simplu, dar nu se adaptează imaginii. O alegere bazată pe energia cumulativă selectează cel mai mic k pentru care $E_k \geq \eta$, cu η de exemplu 0.95 sau 0.99. O altă alegere porneşte de la un buget de memorie. Pentru o imagine $m \times n$, compresia strictă cere aproximativ $k(m+n+1) < mn$; de aici putem deduce limita utilă pentru k.

Metrici

Pe lângă eroarea Frobenius, putem calcula MSE, eroarea medie pătratică pe pixeli, şi PSNR. Aceste metrici sunt utile pentru comparaţii reproducibile, dar nu înlocuiesc inspecţia vizuală. În special, degradările concentrate pe contururi pot fi perceptual importante chiar dacă eroarea globală este moderată.

Laborator: curbă calitate–cost

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru imaginea `cameraman.png` sau alta în tonuri de gri, calculaţi pentru o gamă de valori k: raportul de stocare, eroarea Frobenius, MSE şi timpul de reconstrucţie. Reprezentaţi calitatea în funcţie de cost. Identificaţi un “cot” al curbei unde creşterea lui k produce îmbunătăţiri vizuale tot mai mici.

Exerciţii şi verificarea înţelegerii

1. Derivaţi cel mai mare k pentru care stocarea factorilor trunchiaţi este mai mică decât stocarea brută a unei matrice $m \times n$.
2. Pentru valori singulare $10, 5, 2, 1$, calculaţi energia cumulativă pentru $k = 1, 2, 3$.
3. Dacă înmulţim toate intensităţile cu 2, cum se modifică valorile singulare şi energia cumulativă normalizată?
4. De ce truncarea SVD este globală? Explicaţi de ce o modificare locală a imaginii poate influenţa mai mulţi vectori singulari.
5. Comparaţi conceptual SVD globală cu o metodă pe blocuri.

Mini-proiect: compresie pe blocuri

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Împărţiţi imaginea în blocuri, de exemplu 32×32 sau 64×64, şi aplicaţi SVD cu acelaşi prag energetic în fiecare bloc. Comparaţi cu SVD globală la un buget total de stocare asemănător. Analizaţi artefactele locale, timpul şi avantajele/dezavantajele celor două strategii.

Calcul de stocare: ce înseamnă concret rangul k?

O imagine în tonuri de gri 512×512 conţine 262144 valori. O reprezentare SVD trunchiată la rang k păstrează $U_k \in \mathbb{R}^{512\times k}$, cele k valori singulare şi $V_k \in \mathbb{R}^{512\times k}$, deci aproximativ

$$k(512 + 512 + 1) = 1025k$$

numere. Pentru $k = 40$ sunt 41000 numere, adică aproximativ 15.6% din numărul iniţial, un factor idealizat de compresie de circa 6.4. Calculul ignoră formatele de fişier, cuantizarea şi metadatele, dar este util pentru înţelegerea structurii.

Alegerea lui k poate fi făcută şi energetic. Dacă vrem să păstrăm cel puţin 99% din energia Frobenius, alegem cel mai mic k cu

$$\frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_i \sigma_i^2} \geq 0.99.$$

Această regulă este obiectivă, dar nu garantează automat aceeaşi calitate perceptuală pentru imagini diferite.

Sarcină practică: curba rată–distorsiune

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al

cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru imaginea `cameraman.png` şi încă două imagini, calculaţi reconstrucţiile pentru $k = 5, 10, 20, 40, 80, 160$. Pentru fiecare, raportaţi numărul de coeficienţi, eroarea Frobenius relativă şi PSNR. Reprezentaţi eroarea în funcţie de raportul de stocare. Comparaţi imaginile la acelaşi k: de ce unele sunt mai bine aproximabile prin rang mic?

Exerciţii suplimentare

1. Calculaţi numărul de coeficienţi pentru $k = 10, 50, 100$ într-o imagine 1000×1500.
2. Derivaţi eroarea Frobenius a aproximării de rang k în termenii valorilor singulare eliminate.
3. Ce se întâmplă dacă $k = \min(m, n)$?
4. De ce pixelii reconstruiţi pot ieşi în afara intervalului admis şi cum îi trataţi la salvarea imaginii?
5. Comparaţi alegerea lui k prin energie cu alegerea printr-o eroare perceptuală sau PSNR.

12.3 Compresia imaginilor color în spaţiul RGB

Fotografii color

O imagine color nu este o matrice, ci un tensor de ordin 3:

$\mathcal{I} \in \mathbb{R}^{m \times n \times 3}$

- dimensiunea 1=m: înălţime
- dimensiunea 2=n: lăţime
- dimensiunea 3=3: canal de culoare (R, G, B) (roşu, verde, albastru)

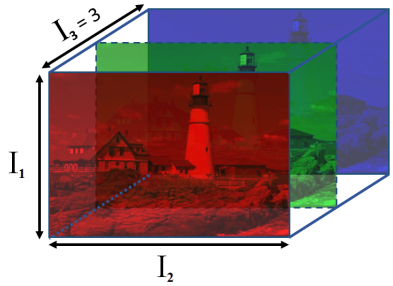


Figura 12.10: Fiecare "felie" este o imagine în tonuri de gri.

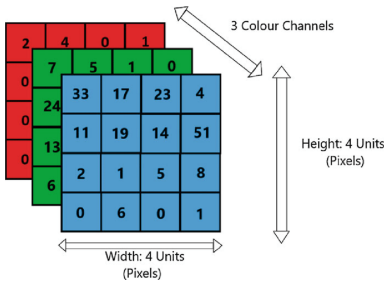


Figura 12.11: Avem "un vector de matrice", adică un tensor de ordin 3.

Putem să aplicăm SVD pentru fiecare canal

```
I = im2double(imread("peppers.png"));   % imagine RGB
size(I)   % m x n x 3
figure(1)
imshow(I)
k = 100;   % rangul de aproximare
Ic = zeros(size(I));
```

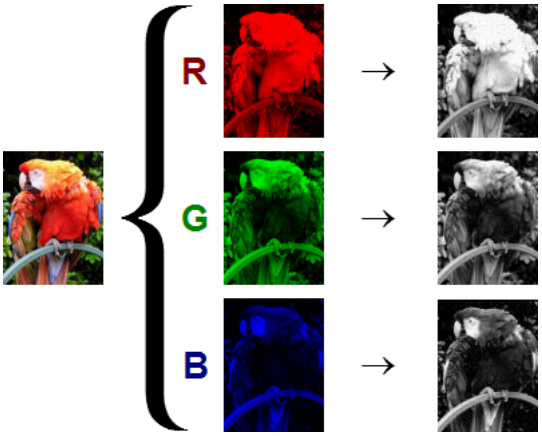


Figura 12.12: Fiecare “felie" este o imagine în tonuri de gri.

```
for c = 1:3
A = I(:,:,c);
[U,S,V] = svd(A, "econ");
Ic(:,:,c) = U(:,1:k) * S(1:k,1:k) * V(:,1:k)';
Ic(:,:,c) = min(max(Ic(:,:,c),0),1);
end
figure(2)
imshow(Ic); title(sprintf("SVD RGB -- rank %d", k))
for c = 1:3
s = svd(I(:,:,c)); cs = cumsum(s.^2) / sum(s.^2);
figure(c+2); plot(cs, "LineWidth", 2); xlabel("k");
ylabel("Energie cumulativa"); title(sprintf("Canal %d (R=1, G=2, B=3)", c))
end
```



Figura 12.13: Fotografie iniţială.

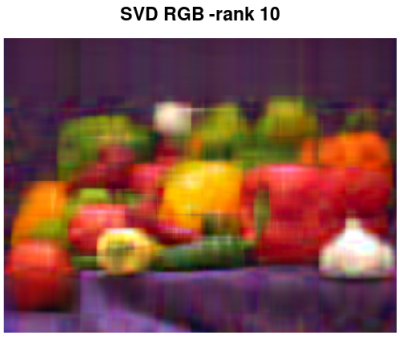


Figura 12.14: Fotografie reconstruită

O abordare care să nu separe canalele

O altă strategie este concatenarea matricelor asociate celor trei canale, urmată de aplicarea SVD asupra matricei rezultate. În acest fel, reducerea se realizează în mod corelat pe cele trei canale, iar aproximaţia ţine seama de structura întregului tensor.

```
I = im2double(imread("peppers.png"));
[m,n,~] = size(I);
A = [I(:,:,1), I(:,:,2), I(:,:,3)];      % m x (3n)
```

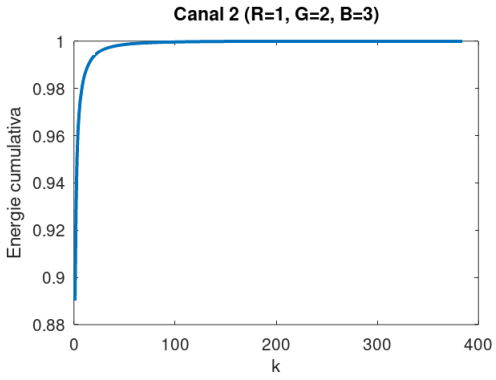


Figura 12.15: Suma cumulativă canal 1.

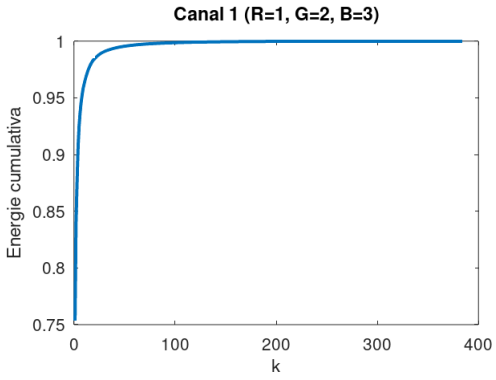


Figura 12.16: Suma cumulativă canal 2.



Figura 12.17: Fotografie iniţială.



Figura 12.18: Fotografie reconstruită

```
[U,S,V] = svd(A,"econ");
              figure(1);  imshow(I); title(sprintf("Initial"))
k = 60;
Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
Ak = min(max(Ak,0),1);
R = Ak(:, 1:n);
G = Ak(:, n+1:2*n);
B = Ak(:, 2*n+1:3*n);
Ic = cat(3, R, G, B);
imshow(Ic)
   figure(2);  imshow(Ic); title(sprintf("SVD RGB -rank %d", k))
 s = svd(A);
 cs = cumsum(s.^2) / sum(s.^2);
 figure(3); clf;  plot(cs, "LineWidth", 2); xlabel("k")
 ylabel("Energie cumulativa");
title(sprintf("Suma cumulativ-canale reunite"))
```

Completare didactică: imagini RGB şi trei moduri de a folosi SVD

O imagine RGB este un tablou $m \times n \times 3$. Fiecare canal R, G, B este o matrice, dar cele trei canale nu sunt independente: muchiile şi structurile geometrice apar adesea simultan în toate. Aplicarea SVD separat pe fiecare canal este simplă şi păstrează exact modelul matricial, însă poate folosi ranguri diferite sau poate pierde corelaţii între culori.

Strategia 1: SVD pe fiecare canal

Calculăm R_k, G_k, B_k separat şi reconstruim tensorul color. Putem alege acelaşi k sau praguri energetice separate. Avantajul este simplitatea; dezavantajul este că tratăm trei probleme independente.

Strategia 2: concatenarea canalelor

Putem forma $A = [R\ G\ B] \in \mathbb{R}^{m\times 3n}$ şi aplica o singură SVD. Vectorii singulari din stânga sunt comuni tuturor canalelor, iar factorul din dreapta poate codifica relaţii între ele. După aproximare, matricea se desparte în cele trei blocuri. Aceasta este o modalitate elementară de a introduce corelaţia între canale fără a folosi încă o descompunere tensorială propriu-zisă.

Cost şi evaluare

Pentru SVD separată, costul şi stocarea sunt suma celor trei probleme. Pentru matricea concatenată, una dintre dimensiuni devine $3n$, ceea ce poate creşte costul descompunerii. Nu există o strategie universal superioară: rezultatul depinde de dimensiuni, conţinut şi criteriul de calitate.

Laborator: canal separat versus concatenat

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Folosiţi `peppers.png`. Pentru un buget de stocare aproximativ comparabil, reconstruiţi imaginea prin cele două strategii. Calculaţi eroarea Frobenius pe tensor şi pe fiecare canal, energia cumulativă şi timpul. Examinaţi zone cu culori saturate şi muchii fine pentru artefacte.

Exerciţii şi verificarea înţelegerii

1. Scrieţi dimensiunile factorilor SVD pentru fiecare canal şi pentru matricea concatenată.
2. Dacă toate cele trei canale sunt identice, ce structură suplimentară are tensorul? De ce o metodă corelată ar trebui să profite de ea?
3. Dacă un canal este zgomot aleator independent de celelalte, cum vă aşteptaţi să se schimbe spectrul singular al concatenării?
4. Formulaţi un criteriu de alegere a rangurilor k_R, k_G, k_B sub un buget total fix.
5. De ce cliparea valorilor reconstruite în intervalul valid al imaginii poate fi necesară după aproximare?

Mini-proiect: alocarea adaptivă a rangului pe canale

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi

executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Alegeţi un buget total $K = k_R + k_G + k_B$. Căutaţi automat combinaţia de ranguri care maximizează energia totală reţinută sau minimizează eroarea Frobenius. Comparaţi cu alegerea simplă $k_R = k_G = k_B$. Discutaţi dacă îmbunătăţirea numerică este şi perceptual vizibilă.

Exemplu de buget de stocare pe canale

Pentru o imagine RGB 512×512, reprezentarea brută are trei matrici, deci $3 \cdot 512^2 = 786432$ valori. Dacă folosim acelaşi rang $k = 40$ pe toate canalele, reprezentarea SVD necesită aproximativ $3 \cdot 40 \cdot 1025 = 123000$ valori, adică aproape acelaşi raport procentual ca în cazul în tonuri de gri.

Nu este obligatoriu însă ca cele trei canale să aibă aceeaşi complexitate. Dacă alegem rangurile $(50, 30, 30)$, numărul idealizat de coeficienţi este $(50+30+30) \cdot 1025 = 112750$, adică aproximativ 14.3% din reprezentarea brută. Alegerea diferenţiată trebuie justificată prin spectrele singulare şi prin efectul vizual, nu doar printr-o preferinţă arbitrară.

De ce tratarea separată a canalelor este simplă, dar nu exploatează totul?

SVD pe fiecare canal exploatează corelaţiile spaţiale din interiorul canalului, dar nu modelează explicit corelaţia dintre roşu, verde şi albastru. Spaţii de culoare precum YCbCr separă mai bine luminanţa de crominanţă şi permit alocarea unui buget diferit informaţiei la care percepţia este mai sensibilă.

Sarcină practică: ranguri adaptive

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Calculaţi spectrele singulare pentru fiecare canal RGB. Alegeţi pentru fiecare canal cel mai mic rang care păstrează 99% din energia sa şi comparaţi cu folosirea unui singur rang comun ales după cel mai dificil canal. Raportaţi stocarea şi calitatea. Reprezentaţi şi erorile pe canale, nu doar eroarea imaginii concatenate.

Exerciţii suplimentare

1. Derivaţi formula de stocare pentru ranguri k_R, k_G, k_B.
2. Dacă un canal este aproape constant, ce vă aşteptaţi să observaţi în spectrul său singular?
3. Explicaţi de ce o eroare numerică egală pe cele trei canale nu produce neapărat acelaşi efect perceptual.
4. Propuneţi o regulă de alocare a unui buget total $K = k_R + k_G + k_B$ între canale.

5. Comparaţi SVD pe canale cu concatenarea matricelor canalelor. Ce structură diferită exploatează fiecare variantă?

12.4 Compresia imaginilor color în spaţiul YCbCr

YCbCr în loc de RGB

YCbCr este folosit în loc de RGB pentru că permite o comprimare mult mai eficientă perceptual, adică se pierde mai puţin din informaţia percepută de ochiul uman, chiar dacă, din punct de vedere numeric, pierderile sunt mai mari.

Limitările spaţiului de culoare RGB

În spaţiul de culoare RGB:

- canalele R, G, B sunt puternic corelate;
- fiecare canal amestecă informaţia de:
  - luminanţă (luminozitate);
  - culoare.

Dacă se aplică metode de comprimare direct în RGB:

- se pierd simultan luminanţa şi culoarea;
- degradarea imaginii devine rapid vizibilă.

Componentele spaţiului de culoare YCbCr

Y, Cb şi Cr sunt cele trei componente ale spaţiului de culoare YCbCr, fiecare având un rol distinct în reprezentarea şi comprimarea imaginilor.

Componenta Y – Luminanţă (brightness). Componenta Y reprezintă luminozitatea imaginii:

- indică cât de deschis sau închis este un pixel;
- conţine structura, contururile şi detaliile imaginii;
- este componenta la care ochiul uman este cel mai sensibil.

Dacă se păstrează doar componenta Y, se obţine o imagine în tonuri de gri, foarte apropiată de original.

Componentele spaţiului de culoare YCbCr

Componenta Cb – Crominanţă albastră (blue-difference).

Componenta Cb măsoară diferenţa dintre componenta albastră şi luminanţă:

$$Cb = B - Y \quad \text{(scalat)}$$

Aceasta:

- indică cât de albastru este pixelul faţă de luminozitatea sa;
- conţine informaţie de culoare, nu de structură;
- este mai puţin importantă din punct de vedere perceptual.

Componentele spaţiului de culoare YCbCr

Componenta Cr – Crominanţă roşie (red-difference).

Componenta Cr măsoară diferenţa dintre componenta roşie şi luminanţă:

$$Cr = R - Y \quad \text{(scalat)}$$

Aceasta:

- indică cât de roşu este pixelul faţă de luminozitatea sa;
- conţine informaţie de culoare;
- este, la fel ca Cb, mai puţin importantă perceptual.

Relaţia dintre RGB şi YCbCr

Componenta Y poate fi aproximată ca o combinaţie ponderată a canalelor RGB, iar Cb şi Cr reprezintă deviaţii de culoare faţă de Y. Conversia standard (simplificată) este:

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B, \\ Cb &= B - Y, \\ Cr &= R - Y. \end{aligned}$$

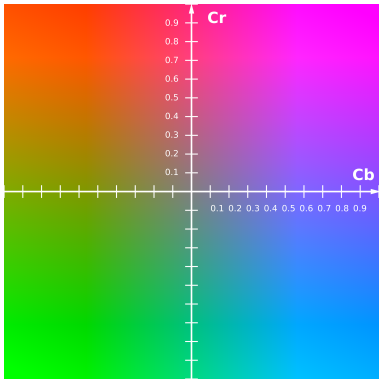


Figura 12.19: Valori pentru Cb şi Cr.



Figura 12.20: Ce extrage Y, Cb, Cr?

Convenţia MATLAB pentru YCbCr

Formulele $Y = 0.299R + 0.587G + 0.114B$, $Cb = B - Y$, $Cr = R - Y$ explică ideea transformării, dar rutina MATLAB `rgb2ycbcr` foloseşte convenţia video ITU-R BT.601, cu scalări şi deplasări ale crominanţelor. Pentru imagini de tip `double` în $[0, 1]$, luminanţa este mapată aproximativ în $[16/255, 235/255]$, iar crominanţele sunt centrate în jurul lui $128/255$. De aceea valorile produse de rutină nu trebuie comparate direct cu formulele simplificate fără a ţine seama de aceste scalări.

Implementare cu YCbCr

```
I = im2double(imread("peppers.png"));          % m x n x 3
YCC = rgb2ycbcr(I);
kY = 80;   kC = 25;
J = zeros(size(YCC));
for c = 1:3
A = YCC(:,:,c);
[U,S,V] = svd(A,"econ");
k = (c==1)*kY + (c~=1)*kC;
J(:,:,c) = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
end
Ic = ycbcr2rgb(J);
Ic = min(max(Ic,0),1);
figure(1)
imshow(Ic)
```

Alte abordări

- În loc să se aplice SVD pentru toată imaginea, se aplică pe blocuri, încât ce se pierde să fie local.
- Se generalizează descompunerea SVD pentru tensori de ordin superior.
- Se poate lucra cu optimizare pe spaţii de tensori.

Intermezzo: utilizarea critică a instrumentelor de calcul şi AI

Observaţia 1: sunt necesare cunoştinţe de algebră numerică şi calcul ştiinţific



Figura 12.21: Imagine de sinteză folosită ca punct de plecare pentru discuţia critică asupra metodelor numerice.

Observaţia 2: o imagine generată de AI poate conţine greşeli. Le puteţi identifica?



Figura 12.22: Exerciţiu de analiză critică: identificaţi eventualele erori matematice sau de reprezentare.

Soluţiile pot fi sugerate de AI, dar noi trebuie să avem capacitatea de a le analiza critic!

Completare didactică: separarea luminanţei de crominanţă

Spaţiul YCbCr este util deoarece separă aproximativ informaţia de luminanţă, la care sistemul vizual uman este foarte sensibil, de două componente de crominanţă. În consecinţă, păstrăm mai multă informaţie în canalul Y şi putem comprima mai agresiv Cb şi Cr.

Formulele prezentate în curs exprimă ideea structurală a transformării. Implementările standard folosesc convenţii precise de scalare şi offset, dependente de standard şi de intervalul numeric. De aceea, în MATLAB este preferabil să folosim `rgb2ycbcr` şi `ycbcr2rgb`, în loc să reimplementăm conversia fără a fixa explicit standardul.

Ranguri diferite pe canale

Dacă alegem $k_Y > k_C$, unde acelaşi rang k_C este folosit pentru Cb şi Cr, stocarea aproximativă este $(m + n + 1)(k_Y + 2k_C)$. Pentru acelaşi buget ca în RGB, putem distribui rangurile în acord cu importanţa perceptuală. Aceasta este o primă întâlnire cu ideea de *compresie perceptuală*: eroarea matematică nu este ponderată uniform după ceea ce percepe utilizatorul.

Exemplu de experiment

Păstraţi, de exemplu, 99% din energia canalului Y, dar numai 90%–95% din energia crominanţei. Comparaţi cu o strategie RGB care păstrează aceeaşi energie în fiecare canal. Este posibil ca eroarea Frobenius totală să fie mai mare în YCbCr, dar imaginea să pară la fel de bună sau chiar mai bună.

Laborator: hartă buget–calitate

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi

cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Pentru mai multe perechi (k_Y, k_C), salvaţi eroarea totală, eroarea pe luminanţă, raportul de stocare şi o imagine reconstruită. Construiţi o grilă de rezultate şi identificaţi configuraţii în care reducerea puternică a crominanţei este aproape invizibilă. Evitaţi concluzii universale bazate pe o singură fotografie.

Exerciţii şi verificarea înţelegerii

1. Explicaţi conceptual de ce luminanţa conţine o mare parte din informaţia de contur.
2. Pentru dimensiuni m, n date, comparaţi stocarea unei strategii RGB cu rang k pe fiecare canal şi YCbCr cu ranguri k_Y, k_C, k_C.
3. De ce o transformare liniară invertibilă între spaţii de culoare nu pierde informaţie înainte de truncarea SVD?
4. Unde apare de fapt pierderea de informaţie în algoritmul YCbCr+SVD?
5. Propuneţi un criteriu de calitate care să acorde o pondere mai mare erorii pe Y decât erorilor pe crominanţă.

Mini-proiect final: compresor SVD comparativ

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi un mic program care primeşte o imagine şi un buget de stocare şi oferă trei opţiuni: tonuri de gri/SVD, RGB pe canale şi YCbCr cu ranguri diferenţiate. Programul trebuie să raporteze rangurile alese, raportul estimat de compresie, eroarea şi timpul. Testaţi pe cel puţin cinci imagini cu conţinut diferit şi redactaţi o concluzie despre limitele SVD ca metodă de compresie a imaginilor.

Exemplu de alocare a rangului în YCbCr

Pentru aceeaşi imagine 512×512, să presupunem că alegem rangurile $k_Y = 60$, $k_{Cb} = 20$, $k_{Cr} = 20$. Reprezentarea SVD foloseşte aproximativ

$$(60 + 20 + 20)(512 + 512 + 1) = 102500$$

coeficienţi, adică circa 13.0% din cele 786432 valori RGB brute. Bugetul mai mare pentru luminanţă reflectă faptul că detaliile geometrice fine sunt în mare măsură codificate în Y, în timp ce variaţiile de crominanţă pot fi adesea aproximate mai agresiv.

Transformarea RGB–YCbCr, în forma sa liniară idealizată, nu produce singură compresie şi este invertibilă atunci când nu cuantizăm sau trunchiem. Pierderea de informaţie apare atunci când reducem rangurile, subsamplăm crominanţa ori cuantizăm coeficienţii. Trebuie precizată etapa la care este introdusă eroarea.

Sarcină practică: aceeaşi rată, două strategii

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Alegeţi un buget total de coeficienţi şi construiţi două reconstrucţii: una cu ranguri egale în RGB şi una cu ranguri diferenţiate în YCbCr. Ajustaţi rangurile astfel încât numărul total de coeficienţi să fie aproximativ egal. Comparaţi PSNR, eroarea pe luminanţă şi inspecţia vizuală. Repetaţi pe fotografii şi pe imagini sintetice cu culori saturate.

Exerciţii suplimentare

1. Calculaţi raportul idealizat de stocare pentru rangurile $(60, 20, 20)$.
2. Explicaţi de ce păstrarea numai a canalului Y produce o imagine monocromă, dar cu structură vizuală recognoscibilă.
3. Ce etapă trebuie aplicată înainte de salvarea unei imagini reconstruite dacă valorile ies din intervalul admis?
4. Propuneţi un criteriu de alegere automată a rangurilor folosind energia singulară pe fiecare canal.
5. Discutaţi limitele comparaţiei cu JPEG: SVD global şi transformările blocurilor folosite în codecuri urmăresc structuri diferite.

Test de capitol

Test – Aplicaţii SVD în compresia imaginilor

1. Compresie de rang mic. O imagine în tonuri de gri este reprezentată de o matrice $A \in \mathbb{R}^{100\times 80}$ cu valori singulare

$$\sigma_1 = 40, \quad \sigma_2 = 15, \quad \sigma_3 = 5, \quad \sigma_4 = 2,$$

iar restul valorilor singulare au suma pătratelor egală cu 1.

a) Scrieţi aproximaţia SVD de rang 2 în forma $A_2 = U_2\Sigma_2 V_2^T$.

b) Folosind rezultatul de aproximaţie optimă din capitol, determinaţi $\|A - A_2\|_F$.

c) Comparaţi numărul de valori care trebuie stocate pentru matricea originală cu numărul necesar reprezentării de rang 2 şi interpretaţi raportul.

2. Analiza unui fragment MATLAB.

```
[U,S,V] = svd(A,'econ');
k = 20;
Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
```

Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi:

A) A_k are rang cel mult k;

B) creşterea lui k păstrează mai multă informaţie din imagine;

C) eroarea Frobenius a aproximaţiei creşte obligatoriu când k creşte;

D) pentru o imagine color, aceeaşi idee poate fi aplicată separat canalelor sau după o transformare de culoare descrisă în capitol.

Capitolul 13

Analiza componentelor principale (PCA)

13.1 Motivaţie: cum descriem date de dimensiune mare prin puţine direcţii?

În multe aplicaţii, fiecare observaţie este descrisă printr-un număr mare de caracteristici. O imagine de 28×28 pixeli poate fi privită ca un vector din $\mathbb{R}^{784}$; un experiment poate produce simultan zeci sau sute de măsurători; un utilizator poate fi descris printr-un vector de interacţiuni cu foarte multe obiecte. Dimensiunea spaţiului ambiant poate fi mare, deşi datele se concentrează adesea în apropierea unui subspaţiu de dimensiune mult mai mică.

Analiza componentelor principale (Principal Component Analysis, PCA) caută tocmai aceste direcţii dominante. În forma sa clasică, PCA răspunde simultan la două întrebări:

- care este subspaţiul afin de dimensiune k care aproximează cel mai bine un nor de puncte în sensul celor mai mici pătrate?
- pe ce direcţii ortogonale variază cel mai mult datele?

Cele două probleme conduc la acelaşi răspuns şi, după capitolul precedent, nu este surprinzător că acest răspuns poate fi calculat prin SVD.

Ideea geometrică

Dacă un nor de puncte din $\mathbb{R}^3$ este foarte apropiat de un plan, atunci trei coordonate sunt în mare parte redundante: două coordonate alese în acel plan descriu aproape toată variaţia datelor. PCA găseşte planul, măsoară câtă informaţie se pierde la proiecţie şi furnizează coordonatele reduse ale fiecărui punct.

Fie $m \geq 2$ şi $x_1, \ldots, x_m \in \mathbb{R}^d$ observaţiile, scrise ca vectori coloană. În implementare vom folosi matricea de date $X \in \mathbb{R}^{m \times d}$, în care linia i este x_i^T. Numărul m reprezintă numărul observaţiilor, iar d numărul caracteristicilor.

13.2 Centrare, varianţă şi covarianţă

Definiţie 13.2.1 (Centroidul şi datele centrate)**.** *Centroidul datelor este*

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x_i \in \mathbb{R}^d.$$

Matricea datelor centrate este

$$X_c = X - \mathbf{1}_m \mu^T,$$

unde $\mathbf{1}_m = (1, \dots, 1)^T \in \mathbb{R}^m$. *Linia* i *a lui* X_c *este* $(x_i - \mu)^T$.

Centrarea nu este un artificiu cosmetic. Ea mută originea în centrul norului de date. Într-adevăr, suma liniilor matricei centrate este zero, adică $\mathbf{1}_m^T X_c = 0$.

Definiţie 13.2.2 (Varianţă, covarianţă şi matrice de covarianţă)**.** *Pentru două caracteristici centrate* $a, b \in \mathbb{R}^m$, *definim covarianţa eşantionului prin* $\operatorname{cov}(a, b) = a^T b/(m-1)$. *În particular,* $\operatorname{var}(a) = \|a\|_2^2/(m-1)$.

Matricea de covarianţă a datelor este

$$C = \frac{1}{m-1} X_c^T X_c \in \mathbb{R}^{d \times d}.$$

Matricea $M = X_c^T X_c = (m-1)C$ *va fi numită, în acest capitol, matricea de dispersie. Factorul* $1/(m-1)$ *modifică valorile proprii printr-un factor constant, dar nu modifică vectorii proprii şi, deci, nici direcţiile PCA.*

Propoziţie 13.2.3. *Matricea de covarianţă* C *este simetrică şi pozitiv semidefinită. Mai mult, pentru orice direcţie unitară* $v \in \mathbb{R}^d$, *varianţa coordonatelor datelor centrate proiectate pe direcţia* v *este* $v^T C v$.

Demonstraţie. Simetria rezultă imediat din $C^T = C$. Pentru orice $v \in \mathbb{R}^d$,

$$v^T C v = \frac{1}{m-1} v^T X_c^T X_c v = \frac{1}{m-1} \|X_c v\|_2^2 \geq 0,$$

deci C este pozitiv semidefinită.

Vectorul $z = X_c v \in \mathbb{R}^m$ conţine coordonatele scalare ale punctelor centrate pe direcţia v: $z_i = (x_i - \mu)^T v$. Deoarece $\mathbf{1}_m^T X_c = 0$, media componentelor lui z este zero. Prin urmare

$$\operatorname{var}(z) = \frac{1}{m-1} \|z\|_2^2 = \frac{1}{m-1} \|X_c v\|_2^2 = v^T C v.$$

□

Exemplu scurt: de ce covarianţa măsoară variaţia comună

Considerăm două caracteristici centrate $a = (1, 0, -1)^T$ şi $b = (2, 0, -2)^T$. Atunci $\operatorname{cov}(a, b) = (1 \cdot 2 + 0 + (-1)(-2))/2 = 2 > 0$: cele două caracteristici cresc şi descresc împreună. Dacă înlocuim b cu $-b$, covarianţa devine -2. Dacă produsul scalar este zero, cele două caracteristici sunt necorelate în sens liniar.

13.3 Cel mai bun subspaţiu afin în sensul celor mai mici pătrate

Dorim să aproximăm punctele x_i printr-un subspaţiu afin $\mathcal{A} = x_0 + L$, unde $L \subset \mathbb{R}^d$ este un subspaţiu liniar de dimensiune k. Măsurăm eroarea totală prin

$$E(x_0, L) = \sum_{i=1}^{m} \operatorname{dist}(x_i, x_0 + L)^2.$$

Dacă P_L este proiecţia ortogonală pe L şi $P_L^\perp = I - P_L$, atunci $\operatorname{dist}(x_i, x_0 + L) = \|P_L^\perp (x_i - x_0)\|_2$.

Teoremă 13.3.1 (Subspaţiul afin optim trece prin centroid)**.** *Pentru un subspaţiu liniar fix L, mulţimea afină care minimizează $E(x_0, L)$ poate fi aleasă sub forma $\mu + L$. Mai precis, toate translaţiile optime x_0 satisfac $P_L^{\perp} x_0 = P_L^{\perp} \mu$, iar ele definesc aceeaşi mulţime afină $\mu + L$.*

Demonstraţie. Punem $P = P_L^{\perp}$. Deoarece P este o proiecţie ortogonală, $P^T = P$ şi $P^2 = P$. Avem

$$E(x_0, L) = \sum_{i=1}^{m} \|P(x_i - x_0)\|_2^2 = \sum_{i=1}^{m} \|Px_i - Px_0\|_2^2.$$

Variabila x_0 apare numai prin $y = Px_0 \in L^{\perp}$. Pentru vectorii $y_i = Px_i$ folosim identitatea standard a centrării:

$$\sum_{i=1}^{m} \|y_i - y\|_2^2 = \sum_{i=1}^{m} \|y_i - \bar{y}\|_2^2 + m\|y - \bar{y}\|_2^2, \qquad \bar{y} = \frac{1}{m} \sum_{i=1}^{m} y_i.$$

O demonstrăm prin scrierea $y_i - y = (y_i - \bar{y}) + (\bar{y} - y)$ şi observarea faptului că $\sum_i (y_i - \bar{y}) = 0$, deci termenii încrucişaţi se anulează. Rezultă că minimul se obţine exact pentru $y = \bar{y}$. Dar

$$\bar{y} = \frac{1}{m} \sum_{i=1}^{m} Px_i = P\mu.$$

Prin urmare $Px_0 = P\mu$, adică $P(x_0 - \mu) = 0$ şi, deci, $x_0 - \mu \in L$. Cum $x_0 + L = \mu + L$ atunci când $x_0 - \mu \in L$, toate alegerile optime descriu aceeaşi mulţime afină. În particular putem lua $x_0 = \mu$. □

Teorema arată că problema afină se reduce la o problemă liniară pentru datele centrate $y_i = x_i - \mu$. De acum înainte notăm

$$E(L) = \sum_{i=1}^{m} \|y_i - P_L y_i\|_2^2, \qquad y_i = x_i - \mu.$$

Lemă 13.3.2 (Forma spectrală a erorii)**.** *Fie $v_1, \ldots, v_k$ o bază ortonormală a lui L şi $V_k = (v_1 | \cdots | v_k)$. Pentru $M = X_c^T X_c$ avem*

$$E(L) = \operatorname{tr}(M) - \sum_{j=1}^{k} v_j^T M v_j.$$

Demonstraţie. Deoarece $P_L = V_k V_k^T$ este proiecţie ortogonală, teorema lui Pitagora dă

$$\|y_i - P_L y_i\|_2^2 = \|y_i\|_2^2 - \|P_L y_i\|_2^2.$$

Mai mult, $\|P_L y_i\|_2^2 = \sum_{j=1}^{k} (v_j^T y_i)^2$. Prin urmare

$$\begin{aligned} E(L) &= \sum_{i=1}^{m} \|y_i\|_2^2 - \sum_{j=1}^{k} \sum_{i=1}^{m} (v_j^T y_i)^2 \\ &= \sum_{i=1}^{m} \|y_i\|_2^2 - \sum_{j=1}^{k} v_j^T \left(\sum_{i=1}^{m} y_i y_i^T \right) v_j. \end{aligned}$$

Dar $\sum_i y_i y_i^T = X_c^T X_c = M$. În plus,

$$\operatorname{tr}(M) = \sum_{i=1}^{m} \operatorname{tr}(y_i y_i^T) = \sum_{i=1}^{m} y_i^T y_i = \sum_{i=1}^{m} \|y_i\|_2^2.$$

Înlocuirea acestor două identităţi în formula precedentă încheie demonstraţia. □

13.4 Principiul Rayleigh–Ritz şi alegerea direcţiilor principale

Rezultatul următor este elementul spectral de bază al PCA.

Teoremă 13.4.1 (Rayleigh–Ritz). *Fie $A \in \mathbb{R}^{d\times d}$ simetrică, cu valori proprii $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ şi o bază ortonormală de vectori proprii $p_1, \ldots, p_d$. Atunci*

$$\max_{\|v\|_2=1} v^T A v = \lambda_1,$$

iar maximul este atins pentru $v = p_1$. Mai general, pentru $1 \leq k < d$,

$$\max_{\substack{\|v\|_2=1 \\ v\perp p_1,\ldots,p_k}} v^T A v = \lambda_{k+1},$$

iar maximul este atins pentru $v = p_{k+1}$.

Demonstraţie. Orice vector $v \in \mathbb{R}^d$ poate fi scris unic în baza ortonormală a vectorilor proprii, $v = \sum_{i=1}^d \alpha_i p_i$. Dacă $\|v\|_2 = 1$, atunci $\sum_i \alpha_i^2 = 1$. Cum $Ap_i = \lambda_i p_i$,

$$v^T A v = \sum_{i=1}^d \lambda_i \alpha_i^2 \leq \lambda_1 \sum_{i=1}^d \alpha_i^2 = \lambda_1.$$

Pentru $v = p_1$ avem egalitate, deci primul maxim este λ_1.

Dacă în plus $v \perp p_1, \ldots, p_k$, atunci $\alpha_1 = \cdots = \alpha_k = 0$. Astfel

$$v^T A v = \sum_{i=k+1}^d \lambda_i \alpha_i^2 \leq \lambda_{k+1} \sum_{i=k+1}^d \alpha_i^2 = \lambda_{k+1},$$

iar egalitatea se obţine pentru $v = p_{k+1}$. □

Pentru a determina simultan cel mai bun subspaţiu de dimensiune k avem nevoie de o observaţie elementară.

Lemă 13.4.2 (Selectarea celor mai mari valori proprii). *Fie $\lambda_1 \geq \cdots \geq \lambda_d$ şi fie $a_i \in [0,1]$ astfel încât $\sum_{i=1}^d a_i = k$, unde $k \in \{1, \ldots, d\}$. Atunci*

$$\sum_{i=1}^d \lambda_i a_i \leq \sum_{i=1}^k \lambda_i.$$

Demonstraţie. Pentru $k = d$ afirmaţia este evidentă. Presupunem $k < d$. Deoarece

$$\sum_{i=1}^k (1 - a_i) = k - \sum_{i=1}^k a_i = \sum_{i=k+1}^d a_i,$$

şi $\lambda_i \geq \lambda_k$ pentru $i \leq k$, în timp ce $\lambda_i \leq \lambda_k$ pentru $i > k$, obţinem

$$\begin{aligned}\sum_{i=1}^k \lambda_i - \sum_{i=1}^d \lambda_i a_i &= \sum_{i=1}^k \lambda_i (1 - a_i) - \sum_{i=k+1}^d \lambda_i a_i \\ &\geq \lambda_k \sum_{i=1}^k (1 - a_i) - \lambda_k \sum_{i=k+1}^d a_i = 0.\end{aligned}$$

□

Teoremă 13.4.3 (PCA ca cea mai bună aproximaţie afină)**.** *Fie C matricea de covarianţă a datelor centrate şi fie*

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d \geq 0$$

valorile sale proprii, cu vectori proprii ortonormali $p_1, \ldots, p_d$. Pentru $1 \leq k \leq d$, subspaţiul

$$L_k = \operatorname{span}\{p_1, \ldots, p_k\}$$

minimizează suma pătratelor distanţelor datelor la un subspaţiu afin de dimensiune k. Un subspaţiu afin optim este $\mu + L_k$, iar eroarea minimă este

$$E_{\min} = (m-1) \sum_{j=k+1}^{d} \lambda_j.$$

Echivalent, dacă $\gamma_j = (m-1)\lambda_j$ sunt valorile proprii ale lui $M = X_c^T X_c$, atunci $E_{\min} = \sum_{j=k+1}^{d} \gamma_j$.

Demonstraţie. Prin teorema despre centroid este suficient să minimizăm $E(L)$ pentru datele centrate. Conform lemei precedente despre eroare,

$$E(L) = \operatorname{tr}(M) - \sum_{j=1}^{k} v_j^T M v_j,$$

unde $v_1, \ldots, v_k$ este o bază ortonormală a lui L. Aşadar trebuie să maximizăm suma $\sum_j v_j^T M v_j$.

Fie $q_1, \ldots, q_d$ o bază ortonormală de vectori proprii ai lui M, în aceeaşi ordine descrescătoare a valorilor proprii $\gamma_i = (m-1)\lambda_i$. Scriem

$$a_i = \|P_L q_i\|_2^2 = \sum_{j=1}^{k} (q_i^T v_j)^2.$$

Atunci $0 \leq a_i \leq 1$, iar

$$\sum_{i=1}^{d} a_i = \sum_{j=1}^{k} \sum_{i=1}^{d} (q_i^T v_j)^2 = \sum_{j=1}^{k} \|v_j\|_2^2 = k.$$

În plus,

$$\begin{aligned} \sum_{j=1}^{k} v_j^T M v_j &= \sum_{j=1}^{k} \sum_{i=1}^{d} \gamma_i (q_i^T v_j)^2 = \sum_{i=1}^{d} \gamma_i a_i \\ &\leq \sum_{i=1}^{k} \gamma_i, \end{aligned}$$

ultima inegalitate fiind exact lema de selectare a celor mai mari valori proprii. Egalitatea se obţine pentru $L = L_k = \operatorname{span}\{q_1, \ldots, q_k\}$, adică pentru primele k direcţii proprii.

În acest caz,

$$E_{\min} = \operatorname{tr}(M) - \sum_{i=1}^{k} \gamma_i.$$

Cum M este simetrică, $\operatorname{tr}(M) = \sum_{i=1}^{d} \gamma_i$, deci

$$E_{\min} = \sum_{i=k+1}^{d} \gamma_i = (m-1) \sum_{i=k+1}^{d} \lambda_i.$$

□

Unicitate şi valori proprii multiple

Dacă $k < d$ şi $\lambda_k > \lambda_{k+1}$, subspaţiul principal L_k este determinat unic. Dacă $\lambda_k = \lambda_{k+1}$, direcţiile proprii individuale din spaţiul propriu multiplu nu sunt unice; pot exista mai multe subspaţii optime. PCA determină atunci un spaţiu invariant, nu neapărat o bază unică a acestuia. În plus, semnul fiecărui vector propriu este arbitrar: p_i şi $-p_i$ reprezintă aceeaşi direcţie.

13.5 PCA ca maximizare a varianţei

Interpretarea anterioară este echivalentă cu formularea statistică obişnuită. Prima direcţie principală este direcţia unitară pe care proiecţiile au varianţă maximă. A doua direcţie principală maximizează varianţa printre direcţiile ortogonale pe prima, şi aşa mai departe.

Definiţie 13.5.1 (Direcţii principale şi scoruri)**.** *Vectorii proprii $p_1, \dots, p_d$ ai lui C, ordonaţi după valori proprii descrescătoare, se numesc* direcţii principale. *Pentru $P_k = (p_1 | \cdots | p_k)$, matricea*

$$Z_k = X_c P_k \in \mathbb{R}^{m \times k}$$

conţine scorurile PCA*: linia i este reprezentarea redusă a observaţiei x_i, iar coloana j conţine coordonatele tuturor observaţiilor pe direcţia principală p_j.*

Propoziţie 13.5.2. *Coloanele lui $Z_d = X_c P_d$ sunt necorelate. Mai exact,*

$$\frac{1}{m-1} Z_d^T Z_d = \operatorname{diag}(\lambda_1, \dots, \lambda_d).$$

În particular, varianţa celei de-a j-a componente principale este λ_j.

Demonstraţie. Dacă $P_d = (p_1 | \cdots | p_d)$ este ortogonală şi $CP_d = P_d \Lambda$, cu $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_d)$, atunci

$$\frac{1}{m-1} Z_d^T Z_d = P_d^T \left(\frac{1}{m-1} X_c^T X_c \right) P_d = P_d^T C P_d = P_d^T P_d \Lambda = \Lambda.$$

Elementele nediagonale sunt covarianţele dintre componente şi sunt zero. □

Două formulări, aceeaşi geometrie

Formularea "minimizăm distanţa la un subspaţiu" şi formularea "maximizăm varianţa păstrată" sunt echivalente deoarece energia totală a datelor centrate este fixă. Ce nu este explicat de subspaţiul ales devine eroare de reconstrucţie; ce este explicat apare ca variaţie pe direcţiile păstrate.

13.6 Legătura exactă dintre PCA şi SVD

După capitolul precedent, PCA poate fi calculată direct prin SVD a matricei centrate.

Teoremă 13.6.1 (SVD furnizează PCA)**.** *Fie $X_c \in \mathbb{R}^{m \times d}$ matricea datelor centrate şi fie*

$$X_c = U \Sigma V^T$$

o descompunere SVD, cu valori singulare $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$. Atunci:

1. *vectorii singulari drepţi v_j (coloanele lui V) sunt direcţiile principale PCA;*
2. *valorile proprii ale matricei de covarianţă sunt*

$$\lambda_j = \frac{\sigma_j^2}{m-1};$$

3. *matricea scorurilor complete este*

$$Z = X_c V = U\Sigma;$$

4. *componentele distincte sunt necorelate, iar varianţa coloanei j a lui Z este $\sigma_j^2/(m-1)$.*

Demonstraţie. Din SVD obţinem

$$X_c^T X_c = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T U^T U \Sigma V^T = V\Sigma^T \Sigma V^T.$$

Prin urmare

$$C = \frac{1}{m-1} V \Sigma^T \Sigma V^T.$$

Matricea $\Sigma^T\Sigma$ este diagonală, cu elementele σ_j^2 urmate eventual de zerouri. Astfel coloanele lui V sunt vectori proprii ai lui C, iar valorile proprii corespunzătoare sunt $\lambda_j = \sigma_j^2/(m-1)$. Deoarece valorile singulare sunt ordonate descrescător, şi valorile proprii sunt în ordinea cerută de PCA. Rezultă punctele 1 şi 2.

Pentru scoruri,

$$Z = X_c V = U\Sigma V^T V = U\Sigma,$$

ceea ce demonstrează punctul 3. În fine,

$$\frac{1}{m-1} Z^T Z = \frac{1}{m-1} \Sigma^T U^T U \Sigma = \frac{1}{m-1} \Sigma^T \Sigma,$$

care este diagonală. Aceasta dovedeşte lipsa corelaţiei între coloane şi formula varianţelor. □

De ce calculăm PCA prin SVD şi nu formând explicit $X_c^T X_c$?

Matematic, diagonalizarea lui $C = (m-1)^{-1} X_c^T X_c$ şi descompunerea SVD a lui X_c sunt echivalente. Numeric, formarea explicită a lui $X_c^T X_c$ poate fi mai puţin favorabilă, deoarece numărul de condiţionare spectral se pătrăţeşte pe subspaţiul de rang complet: $\kappa_2(X_c^T X_c) = \kappa_2(X_c)^2$. Algoritmii SVD bine implementaţi evită formarea explicită a produsului normal şi sunt de regulă alegerea recomandată pentru PCA densă de dimensiune moderată.

13.7 Reducerea dimensiunii şi reconstrucţia

Fie $V_k = (v_1 | \cdots | v_k)$. Reprezentarea redusă este

$$Z_k = X_c V_k \in \mathbb{R}^{m \times k}.$$

Observaţia $x_i \in \mathbb{R}^d$ este astfel înlocuită printr-un vector de numai k coordonate. Pentru a reveni în spaţiul iniţial folosim

$$\widehat{X}_k = Z_k V_k^T + \mathbf{1}_m \mu^T = X_c V_k V_k^T + \mathbf{1}_m \mu^T.$$

Linia i a lui $\widehat{X}_k$ este proiecţia ortogonală a lui x_i pe subspaţiul afin $\mu + L_k$.

Propoziţie 13.7.1 (Eroarea de reconstrucţie)**.** *Dacă $\sigma_1 \geq \cdots \geq \sigma_r > 0$ sunt valorile singulare nenule ale lui X_c, atunci*

$$\|X - \widehat{X}_k\|_F^2 = \|X_c - X_c V_k V_k^T\|_F^2 = \sum_{j=k+1}^{r} \sigma_j^2 = (m-1) \sum_{j=k+1}^{d} \lambda_j.$$

Demonstraţie. Din SVD economică $X_c = U_r \Sigma_r V_r^T$ rezultă

$$X_c V_k V_k^T = \sum_{j=1}^{k} \sigma_j u_j v_j^T.$$

Prin urmare

$$X_c - X_c V_k V_k^T = \sum_{j=k+1}^{r} \sigma_j u_j v_j^T.$$

Matricele $u_j v_j^T$ sunt ortonormale în produsul scalar Frobenius, deoarece

$$\langle u_i v_i^T, u_j v_j^T \rangle_F = \operatorname{tr}(v_i u_i^T u_j v_j^T) = (u_i^T u_j)(v_i^T v_j) = \delta_{ij}.$$

Teorema lui Pitagora în spaţiul matricelor dă

$$\|X_c - X_c V_k V_k^T\|_F^2 = \sum_{j=k+1}^{r} \sigma_j^2.$$

Cum $\lambda_j = \sigma_j^2/(m-1)$, rezultă ultima egalitate. Adăugarea şi scăderea centroidului nu modifică reziduul dintre X şi reconstrucţia sa, deci prima egalitate este evidentă. □

Acest rezultat este aceeaşi geometrie ca teorema Eckart–Young–Mirsky, dar interpretată pe linii ca observaţii şi după centrare. PCA nu este o procedură independentă de SVD; ea oferă o interpretare a SVD pentru date centrate.

13.8 Câte componente păstrăm? Varianţa explicată

Dacă $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$, variaţia totală este $\operatorname{tr}(C) = \sum_j \lambda_j$. Proporţia de varianţă explicată de primele k direcţii este

$$R_k = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{d} \lambda_j} = \frac{\sum_{j=1}^{k} \sigma_j^2}{\sum_{j=1}^{r} \sigma_j^2}.$$

O regulă practică este alegerea celui mai mic k pentru care $R_k \geq \alpha$, de exemplu $\alpha = 0.90$ sau 0.95. Pragul nu este însă o lege universală: un k potrivit depinde de obiectivul aplicaţiei.

Varianţă explicată versus utilitate

PCA păstrează direcţiile cu variaţie mare, nu neapărat direcţiile cele mai utile pentru o sarcină ulterioară. O caracteristică cu variaţie mică poate fi foarte importantă pentru clasificare. De aceea procentul de varianţă explicată este un criteriu geometric, nu o garanţie privind performanţa într-o altă problemă.

Graficul valorilor proprii / valorilor singulare

În practică se reprezintă adesea λ_j sau σ_j^2 în funcţie de j (un *scree plot*) şi, separat, suma cumulativă R_k. O schimbare pronunţată de pantă sugerează o separare între direcţiile dominante şi cele secundare. Pentru date fără o astfel de separare, alegerea lui k trebuie justificată prin obiectivul aplicaţiei şi prin eroarea de reconstrucţie.

13.9 Exemplu complet calculat

Considerăm patru observaţii în $\mathbb{R}^2$:

$$x_1 = (2,0)^T, \quad x_2 = (0,2)^T, \quad x_3 = (3,1)^T, \quad x_4 = (1,3)^T.$$

Centroidul este $\mu = (3/2, 3/2)^T$. Datele centrate sunt

$$y_1 = \begin{pmatrix} 1/2 \\ -3/2 \end{pmatrix}, \quad y_2 = \begin{pmatrix} -3/2 \\ 1/2 \end{pmatrix}, \quad y_3 = \begin{pmatrix} 3/2 \\ -1/2 \end{pmatrix}, \quad y_4 = \begin{pmatrix} -1/2 \\ 3/2 \end{pmatrix}.$$

Matricea de dispersie este

$$M = X_c^T X_c = \sum_{i=1}^{4} y_i y_i^T = \begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix},$$

iar matricea de covarianţă este $C = M/3$.

Polinomul caracteristic al lui M este

$$\det(M - \gamma I) = (5-\gamma)^2 - 9 = (\gamma - 8)(\gamma - 2),$$

deci valorile proprii sunt $\gamma_1 = 8$ şi $\gamma_2 = 2$. Putem alege

$$p_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \qquad p_2 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Valorile proprii ale matricei de covarianţă sunt $\lambda_1 = 8/3$ şi $\lambda_2 = 2/3$.

Pentru $k = 1$, dreapta PCA este $\mu + \operatorname{span}\{p_1\}$. Ecuaţia ei este $x_1 + x_2 = 3$. Scorurile sunt

$$z_i = p_1^T y_i, \qquad (z_1, z_2, z_3, z_4) = (\sqrt{2}, -\sqrt{2}, \sqrt{2}, -\sqrt{2}).$$

Reconstrucţiile sunt $\widehat{x}_i = \mu + z_i p_1$; astfel

$$\widehat{x}_1 = \widehat{x}_3 = (5/2, 1/2)^T, \qquad \widehat{x}_2 = \widehat{x}_4 = (1/2, 5/2)^T.$$

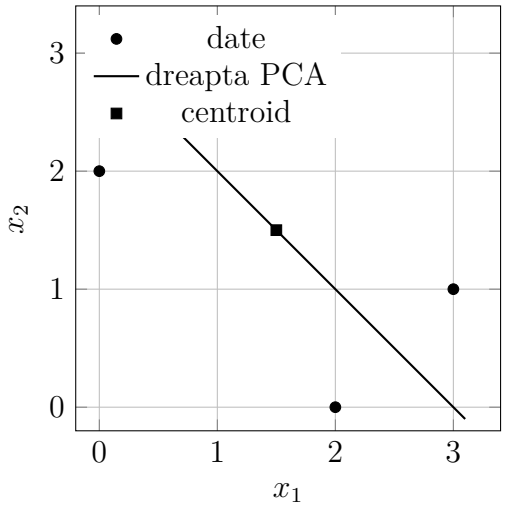
Pentru fiecare punct eroarea pătratică este $1/2$, deci eroarea totală este 2. Teorema PCA prezice exact

$$E_{\min} = \gamma_2 = 2 = (m-1)\lambda_2.$$

Prima componentă explică

$$R_1 = \frac{\lambda_1}{\lambda_1 + \lambda_2} = \frac{8}{10} = 0.8,$$

adică 80% din variaţia totală.



Ce trebuie observat în exemplu

1. Refaceţi manual calculul lui M şi diagonalizarea sa.
2. Verificaţi direct că $p_1^T C p_2 = 0$ şi că varianţele scorurilor sunt $8/3$ şi $2/3$.
3. Calculaţi proiecţia fiecărui punct pe dreapta $x_1 + x_2 = 3$ fără a folosi formula PCA, doar formula proiecţiei ortogonale, şi comparaţi rezultatele.
4. Explicaţi de ce schimbarea lui p_1 cu $-p_1$ modifică semnul scorurilor, dar nu reconstrucţia.

13.10 Centrare sau standardizare?

Centrarea face parte din formularea PCA pentru o aproximaţie afină. Standardizarea este o decizie suplimentară. Dacă variabila j are abaterea standard $s_j > 0$, putem înlocui x_{ij} cu $(x_{ij} - \mu_j)/s_j$. PCA aplicată datelor standardizate este echivalentă cu o analiză bazată pe matricea de corelaţie, nu pe covarianţa în unităţile originale.

Scara caracteristicilor contează

Dacă o coloană este măsurată în metri şi alta în milimetri, a doua poate avea numeric o varianţă mult mai mare numai din cauza unităţii de măsură şi poate domina PCA. Standardizarea este potrivită când caracteristicile trebuie tratate comparabil. Nu trebuie însă aplicată automat: dacă mărimea absolută a variaţiei are semnificaţie fizică, standardizarea poate elimina tocmai informaţia relevantă.

13.11 Algoritmul PCA în MATLAB

Pentru un set de date $X \in \mathbb{R}^{m \times d}$ şi un număr k de componente, algoritmul recomandat pentru acest curs este:

1. calculaţi centroidul μ pe coloane;
2. centraţi: $X_c = X - \mathbf{1}_m \mu^T$;
3. calculaţi SVD economică $X_c = U \Sigma V^T$;
4. păstraţi $V_k = V(:, 1:k)$;
5. calculaţi scorurile $Z_k = X_c V_k$;
6. reconstruiţi, dacă este necesar, $\widehat{X}_k = Z_k V_k^T + \mathbf{1}_m \mu^T$;
7. calculaţi proporţiile de varianţă explicată din σ_j^2.

O implementare elementară, fără apel la funcţia MATLAB `pca`, este:

```
function model = myPCA(X,k)
    [m,d] = size(X);
    if k < 1 || k > min(m-1,d)
        error('k trebuie sa fie intre 1 si min(m-1,d).');
    end

    mu = mean(X,1);
    Xc = X - mu; % implicit expansion
    [U,S,V] = svd(Xc,'econ');
```

```
    s = diag(S);
    Vk = V(:,1:k);
    Z = Xc*Vk;
    Xhat = Z*Vk' + mu;

    energy = s.^2;
    totalEnergy = sum(energy);
    if totalEnergy > 0
        explained = energy/totalEnergy;
    else
        explained = zeros(size(energy));
    end

    model.mu = mu;
    model.V = V;
    model.singularValues = s;
    model.Vk = Vk;
    model.scores = Z;
    model.reconstruction = Xhat;
    model.explained = explained;
    model.cumulativeExplained = cumsum(explained);
end
```

Observaţi că U nu este necesară pentru codarea directă $Z = X_c V_k$, dar SVD o produce şi permite verificarea identităţii $Z = U(:, 1:k)S(1:k, 1:k)$.

Laborator PCA: de la calcul manual la reducerea dimensiunii

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Obiectiv. Implementaţi PCA şi verificaţi atât geometria, cât şi relaţiile spectrale.

1. **Calcul manual.** Folosiţi cele patru puncte din exemplul precedent. Determinaţi μ, X_c, C, valorile proprii, direcţiile principale, scorurile pentru $k = 1$ şi reconstrucţiile. Aceste valori vor fi datele de referinţă.
2. **Pseudocod.** Scrieţi algoritmul fără sintaxă MATLAB. Precizaţi dimensiunile tuturor matricelor la fiecare pas.
3. **Implementare proprie.** Scrieţi funcţia `myPCA`. Nu folosiţi `pca` în această etapă; este permis `svd`, deoarece SVD este algoritmul numeric studiat în capitolul precedent.
4. **Verificări.** Pentru exemplul manual verificaţi `norm(V'*V-eye(size(V,2)))` şi `norm(X-Xhat,'fro')`. Comparaţi eroarea pătratică cu suma valorilor singulare eliminate la pătrat.
5. **Nor sintetic.** Generaţi $t = \text{linspace}(-2, 2, 200)^T$ şi date de forma

$$X = \begin{pmatrix} t & 2t \end{pmatrix} + \eta,$$

unde η este zgomot gaussian mic. Rotiţi apoi norul cu o matrice ortogonală 2×2. Estimaţi prima direcţie principală şi reprezentaţi grafic datele, centroidul şi direcţia PCA.

6. **Varianţă explicată.** Pentru un set de date cu cel puţin cinci caracteristici, reprezentaţi `cumsum(explained)` şi determinaţi cel mai mic k pentru pragurile 90%, 95% şi 99%.
7. **Control MATLAB.** Dacă este disponibilă funcţia `pca`, folosiţi-o numai la final pentru verificare. Ţineţi seama că semnele vectorilor principali pot fi opuse fără ca rezultatul să fie diferit.

De explicat la predare. De ce se centrează datele? De ce vectorii singulari drepţi, şi nu cei stângi, sunt direcţiile din spaţiul caracteristicilor? De ce σ_j^2 măsoară variaţia? De ce formarea explicită a lui $X_c^T X_c$ poate fi numeric mai puţin avantajoasă?

13.12 Aplicaţie: o colecţie de imagini ca nor de puncte

În capitolul precedent am comprimat o singură imagine prin aproximaţie de rang mic. PCA devine diferită şi mai interesantă atunci când avem o *colecţie* de imagini similare. Dacă fiecare imagine are $p \times q$ pixeli, o vectorizăm într-un punct din $\mathbb{R}^{pq}$. Pentru m imagini obţinem o matrice $X \in \mathbb{R}^{m \times pq}$.

Centroidul μ este imaginea medie. Direcţiile principale $v_j \in \mathbb{R}^{pq}$ pot fi remodelate în matrici $p \times q$ şi vizualizate ca *imagini principale*. Scorurile arată cât din fiecare imagine principală este necesar pentru reconstruirea unei observaţii. Ideea apare, în variante consacrate, în recunoaşterea feţelor (eigenfaces) şi în recunoaşterea cifrelor scrise de mână.

Pentru o imagine nouă x, aproximarea cu k componente este

$$\widehat{x} = \mu + V_k V_k^T (x - \mu).$$

În loc să stocăm cei pq pixeli ai fiecărei imagini, putem stoca k scoruri, împreună cu modelul comun (μ, V_k).

Mini-proiect: PCA pe o colecţie de imagini

Etapele proiectului. **1. Modelare:** definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Nivel: mediu. Folosiţi un set de cel puţin 30 de imagini în tonuri de gri cu aceeaşi dimensiune. Imaginile pot proveni dintr-un set furnizat la curs sau pot fi construite din fotografii proprii, redimensionate în prealabil la o rezoluţie modestă.

1. **Modelare.** Dacă imaginile au $p \times q$ pixeli, definiţi matricea $X \in \mathbb{R}^{m \times pq}$ prin vectorizarea fiecărei imagini. Explicaţi ce reprezintă o linie şi o coloană a lui X.
2. **Calcul manual redus.** Luaţi mai întâi trei "imagini" 2×2 alese de voi, vectorizaţi-le şi executaţi manual centrarea şi cel puţin un pas al calculului matricei de covarianţă. Nu începeţi proiectul direct cu date mari.
3. **Algoritm.** Scrieţi pseudocodul pentru antrenarea modelului PCA şi pentru reconstrucţia unei imagini noi. Precizaţi separat etapa de antrenare şi etapa de utilizare a modelului.

4. **MATLAB.** Implementaţi PCA cu `svd(...,'econ')`, fără `pca`. Afişaţi imaginea medie şi primele 6 direcţii principale remodelate la dimensiunea $p \times q$; pentru afişare puteţi rescala valorile unei direcţii la intervalul $[0, 1]$, fără a modifica vectorul folosit în calcule.
5. **Alegerea lui k.** Determinaţi cel mai mic k care explică cel puţin 90% şi apoi 95% din variaţie. Comparaţi vizual reconstrucţiile pentru $k \in \{2, 5, 10, k_{90}, k_{95}\}$, ajustând lista dacă dimensiunile sunt mai mici.
6. **Validare.** Pentru fiecare k raportaţi eroarea relativă Frobenius $\|X - \widehat{X}_k\|_F / \|X_c\|_F$ şi verificaţi relaţia cu valorile singulare eliminate.
7. **Compresie.** Comparaţi numărul de valori necesare pentru stocarea brută a setului X cu numărul necesar pentru (μ, V_k, Z_k). Discutaţi în ce regim apare un câştig real.
8. **Interpretare.** Alegeţi două direcţii principale şi explicaţi ce tip de variaţie vizuală par să codifice. Nu este suficientă afişarea imaginilor: formulaţi o interpretare argumentată.

Extensie opţională. Dacă imaginile sunt împărţite în câteva clase, construiţi câte un subspaţiu PCA pentru fiecare clasă. Pentru o imagine nouă x, calculaţi reziduul

$$d_c(x) = \|(x - \mu_c) - V_{c,k} V_{c,k}^T (x - \mu_c)\|_2$$

şi atribuiţi imaginea clasei cu reziduul minim. Explicaţi de ce acesta este un clasificator prin distanţa la subspaţii afine, nu o consecinţă automată a PCA.

13.13 Limitări şi situaţii în care PCA trebuie folosită cu atenţie

PCA este simplă, exactă pentru criteriul pătratic şi foarte eficientă, dar tocmai criteriul său explică limitele metodei.

1. **Sensibilitate la outlieri.** Distanţele sunt ridicate la pătrat, astfel încât un punct foarte îndepărtat poate roti semnificativ direcţiile principale.
2. **Dependenţă de scară.** Fără standardizare, caracteristicile cu varianţă numerică mare domină rezultatul.
3. **Structură liniară.** PCA caută subspaţii afine. Un nor situat aproape de o curbă neliniară poate necesita multe componente, chiar dacă geometric are puţine grade de libertate.
4. **Direcţiile nu sunt neapărat interpretabile.** O componentă principală este o combinaţie liniară a caracteristicilor originale. Ea poate fi matematic optimă şi totuşi dificil de interpretat fizic.
5. **Valori proprii multiple.** În prezenţa simetriilor, direcţiile principale individuale pot fi neidentificabile; numai subspaţiul asociat este determinat.

Despre PCA robustă

O strategie pentru reducerea sensibilităţii la outlieri este înlocuirea sumei pătratelor distanţelor cu o penalizare care creşte mai lent. Această modificare rupe însă legătura simplă cu valorile proprii şi SVD şi conduce la probleme de optimizare iterative. Într-un curs introductiv ne limităm la înţelegerea acestei cauze, fără a trata toate variantele robuste.

13.14 Exerciţii

Exerciţii teoretice şi de calcul

1. Demonstraţi direct că $\mathbf{1}_m^T X_c = 0$ şi explicaţi de ce fiecare coloană a lui X_c are medie zero.
2. Demonstraţi că $\operatorname{rank}(C) = \operatorname{rank}(X_c)$ şi deduceţi că, dacă $m < d$, matricea de covarianţă are cel puţin $d - m + 1$ valori proprii nule.
3. Fie v un vector unitar. Arătaţi că suma pătratelor distanţelor datelor centrate la dreapta $\operatorname{span}\{v\}$ este $\operatorname{tr}(M) - v^T M v$. Deduceţi teorema PCA pentru $k = 1$ direct din Rayleigh–Ritz.
4. Refaceţi demonstraţia caracterizării PCA folosind teorema Eckart–Young–Mirsky pentru X_c şi explicaţi de ce cele două demonstraţii sunt echivalente.
5. Dacă $X_c = U \Sigma V^T$, demonstraţi că scorurile $Z = X_c V$ au coloane ortogonale în $\mathbb{R}^m$ şi calculaţi norma coloanei j.
6. Arătaţi că schimbarea semnului unei coloane v_j din V schimbă semnul componentei corespunzătoare din Z, dar lasă neschimbată reconstrucţia $Z_k V_k^T$.
7. Presupuneţi că toate datele se află exact într-un subspaţiu afin de dimensiune k. Demonstraţi că $\lambda_{k+1} = \cdots = \lambda_d = 0$ şi că PCA cu k componente reconstruieşte exact datele.
8. Pentru
$$X = \begin{pmatrix} 0 & 0 \\ 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{pmatrix},$$
calculaţi manual centroidul, matricea de covarianţă, direcţiile principale, scorurile şi procentul de varianţă explicată de prima componentă.
9. Construiţi un exemplu în $\mathbb{R}^2$ pentru care cele două valori proprii ale lui C sunt egale. Explicaţi geometric de ce prima direcţie principală nu este unică.
10. Consideraţi o matrice de date cu două coloane, una exprimată în metri şi cealaltă în milimetri. Arătaţi printr-un exemplu numeric cum schimbarea unităţii poate modifica drastic PCA pe covarianţă, deşi datele fizice sunt aceleaşi. Repetaţi după standardizare.
11. Demonstraţi că alegerea celui mai mic k cu $R_k \geq \alpha$ este echivalentă cu impunerea
$$\|X - \widehat{X}_k\|_F^2 \leq (1 - \alpha)\|X_c\|_F^2.$$
12. **Exerciţiu avansat.** Pentru ponderi $w_i \geq 0$, nu toate nule, consideraţi energia $E_w(L) = \sum_i w_i \|y_i - P_L y_i\|_2^2$. Arătaţi că subspaţiul optim este generat de vectorii proprii dominanţi ai matricei $M_w = \sum_i w_i y_i y_i^T$. Dacă se optimizează şi translaţia, determinaţi centroidul ponderat.

Exerciţiu MATLAB: efectul unui outlier

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi

cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Generaţi un nor de 100 de puncte aproape de dreapta $y = 2x$, cu zgomot gaussian mic. Calculaţi prima direcţie PCA. Adăugaţi apoi un singur punct la distanţă mare de nor şi recalculaţi direcţia. Reprezentaţi cele două drepte principale şi măsuraţi unghiul dintre ele. Repetaţi pentru distanţe crescătoare ale outlierului. Explicaţi rezultatul prin faptul că PCA minimizează o sumă de pătrate şi nu doar printr-o observaţie vizuală.

13.15 Rezumatul capitolului

PCA este o aplicaţie directă şi profundă a ideilor dezvoltate anterior în curs. Datele sunt centrate în jurul centroidului, iar matricea de covarianţă $C = (m-1)^{-1} X_c^T X_c$ descrie variaţia lor liniară. Direcţiile principale sunt vectorii proprii corespunzători celor mai mari valori proprii ale lui C. Aceleaşi direcţii sunt vectorii singulari drepţi ai lui X_c, iar valorile singulare determină varianţa explicată.

Geometric, primele k direcţii definesc cel mai bun subspaţiu afin de dimensiune k în sensul sumei pătratelor distanţelor. Algoritmic, PCA reduce observaţiile de la d coordonate la k scoruri prin $Z_k = X_c V_k$ şi reconstruieşte prin $\widehat{X}_k = Z_k V_k^T + \mathbf{1}_m \mu^T$. Numeric, SVD este preferabilă formării explicite a matricei normale $X_c^T X_c$. În aplicaţii, alegerea lui k, scara caracteristicilor, outlierii şi interpretarea direcţiilor trebuie analizate, nu tratate automat.

Întrebări de autoevaluare

1. Care este diferenţa dintre o direcţie principală v_j şi scorurile componentei principale $X_c v_j$?
2. De ce PCA pe date necentrate rezolvă, în general, o altă problemă geometrică decât aproximarea printr-un subspaţiu afin?
3. Cum se citeşte numeric o valoare singulară mică a lui X_c în contextul PCA?
4. În ce situaţie are sens standardizarea caracteristicilor înainte de PCA şi când poate fi ea nedorită?
5. Ce relaţie trebuie verificată pentru a confirma că implementarea reconstrucţiei este coerentă cu teoria?

Notă bibliografică

Expunerea din acest capitol urmăreşte două perspective complementare din materialele consultate: formularea PCA ca problemă de aproximare printr-un subspaţiu afin şi reducere de dimensiune, prezentă în notele lui Jeff Calder, şi formularea prin varianţă, covarianţă, principiul Rayleigh–Ritz şi SVD, dezvoltată de Jean Gallier şi Jocelyn Quaintance. În manual am folosit o notaţie unificată şi am completat demonstraţiile astfel încât rezultatele să poată fi urmărite fără apel la fapte nedemonstrate din teoria PCA.

Test de capitol

Test – Analiza componentelor principale (PCA)

1. Calcul şi interpretare. Se consideră patru observaţii bidimensionale

$$x_1 = (1, 1), \quad x_2 = (2, 2), \quad x_3 = (3, 3), \quad x_4 = (4, 4).$$

a) Calculaţi centroidul şi matricea datelor centrate X_c.
b) Determinaţi prima direcţie principală şi explicaţi geometric rezultatul.
c) Arătaţi că o singură componentă principală reconstruieşte exact aceste date.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi.

A) PCA trebuie, în formularea afină din capitol, aplicată datelor centrate;
B) primele k direcţii principale maximizează variaţia explicată dintre toate subspaţiile de dimensiune k;
C) valorile singulare ale lui X_c sunt legate de varianţele explicate;
D) schimbarea unităţilor unei caracteristici nu poate influenţa niciodată direcţiile PCA.

Cazul $m < d$

Dacă matricea centrată $X_c \in \mathbb{R}^{m\times d}$ are $m \ll d$ (de exemplu, câteva zeci de imagini descrise prin mii de pixeli), formarea matricei $d \times d$, $X_c^T X_c$, poate fi costisitoare. Putem diagonaliza matricea mai mică $X_c X_c^T \in \mathbb{R}^{m\times m}$. Dacă $X_c X_c^T u_j = \sigma_j^2 u_j$ şi $\sigma_j > 0$, atunci

$$v_j = \frac{X_c^T u_j}{\sigma_j}$$

este un vector singular drept şi o direcţie principală. În calcule numerice, se poate folosi direct SVD-ul lui X_c, fără formarea explicită a matricilor normale.

Capitolul 14

Aplicaţii integratoare: PageRank şi sisteme de recomandare

14.1 De ce introducem aceste aplicaţii abia acum?

În capitolele precedente am construit separat mai multe instrumente: matrici rare, norme induse, rază spectrală, metode iterative, sisteme liniare, problema celor mai mici pătrate, regularizare, factorizări QR şi Cholesky, metode de optimizare şi SVD. În această parte le folosim împreună. Scopul nu este descrierea completă a unor sisteme comerciale actuale, ci înţelegerea unor modele matematice reprezentative în care noţiunile cursului apar împreună şi pot fi analizate riguros.

14.2 PageRank ca problemă de calcul ştiinţific

14.2.1 De la un graf orientat la o matrice stocastică

Considerăm n pagini şi o matrice de adiacenţă $W \in \mathbb{R}^{n\times n}$, unde $w_{ij} = 1$ dacă pagina i conţine o legătură către pagina j şi $w_{ij} = 0$ în caz contrar. Notăm cu $d_i = \sum_{j=1}^{n} w_{ij}$ numărul legăturilor care pleacă din pagina i. Atunci când $d_j > 0$, definim

$$P_{ij} = \frac{w_{ji}}{d_j}.$$

Prin această convenţie, coloana j conţine probabilităţile de trecere din pagina j către celelalte pagini şi, prin urmare, $\sum_i P_{ij} = 1$.

O pagină fără legături de ieşire are $d_j = 0$ şi se numeşte *nod suspendat*. Pentru a păstra interpretarea probabilistică, coloana corespunzătoare trebuie înlocuită conceptual printr-un vector de probabilităţi v; alegerea uniformă $v = e/n$ este cea mai simplă. În aplicaţii mari nu efectuăm însă această înlocuire în matricea stocată, deoarece am putea distruge raritatea. Notăm cu P_0 matricea rară în care coloanele nodurilor suspendate rămân nule şi cu $\mathcal{D}$ mulţimea indicilor acestor noduri. Atunci aplicarea matricei stocastice corectate P se face fără a o forma explicit:

$$Px = P_0 x + \left(\sum_{j\in\mathcal{D}} x_j \right) v.$$

Astfel costul unei aplicări rămâne proporţional cu $\mathrm{nnz}(P_0) + n$, iar structura rară este păstrată. În continuare, prin P înţelegem această matrice stocastică corectată.

Propoziţie 14.2.1. *Dacă $P \in \mathbb{R}^{n\times n}$ are elemente nenegative şi sumele pe coloane sunt egale cu 1, atunci $\|P\|_1 = 1$ şi, pentru orice $z \in \mathbb{R}^n$, $\|Pz\|_1 \le \|z\|_1$.*

Demonstraţie. Norma matricială indusă de norma 1 este maximul sumelor pe coloane. Deoarece $P_{ij} \ge 0$ şi fiecare coloană are suma 1, rezultă imediat $\|P\|_1 = 1$. Compatibilitatea normei induse dă $\|Pz\|_1 \le \|P\|_1\|z\|_1 = \|z\|_1$. Putem verifica şi direct:

$$\|Pz\|_1 = \sum_i \left|\sum_j P_{ij}z_j\right| \le \sum_i \sum_j P_{ij}|z_j| = \sum_j |z_j| \sum_i P_{ij} = \|z\|_1.$$

□

14.2.2 Modelul surferului aleator şi teleportarea

Dacă un utilizator urmează la fiecare pas o legătură aleasă aleator, distribuţia probabilităţilor evoluează prin $x^{(k+1)} = Px^{(k)}$. Această schemă poate avea dificultăţi structurale: pot exista clase închise, periodicitate sau noduri suspendate. Introducem de aceea un parametru $0 \le \alpha < 1$ şi un vector de probabilităţi $v \ge 0$, $e^T v = 1$. Cu probabilitatea α urmăm o legătură, iar cu probabilitatea $1-\alpha$ efectuăm o teleportare conform lui v. Iteraţia devine

$$x^{(k+1)} = (1-\alpha)v + \alpha P x^{(k)}. \tag{14.1}$$

În implementarea rară, aceeaşi relaţie se evaluează sub forma

$$x^{(k+1)} = (1-\alpha)v + \alpha\left[P_0 x^{(k)} + \left(\sum_{j\in\mathcal{D}} x_j^{(k)}\right) v\right],$$

fără construirea coloanelor dense asociate nodurilor suspendate. Un punct fix x^* satisface

$$(I - \alpha P)x^* = (1-\alpha)v. \tag{14.2}$$

Aşadar PageRank este simultan o problemă de punct fix, un sistem liniar şi o problemă spectrală.

Teoremă 14.2.2 (Existenţa şi unicitatea vectorului PageRank)**.** *Fie P o matrice stocastică pe coloane, $0 \le \alpha < 1$ şi $v \in \mathbb{R}^n$. Atunci ecuaţia (14.2) are o soluţie unică. Dacă v este un vector de probabilităţi, soluţia x^* este de asemenea un vector de probabilităţi.*

Demonstraţie. Pentru unicitate este suficient să arătăm că $I - \alpha P$ este inversabilă. Dacă $(I-\alpha P)z = 0$, atunci $z = \alpha P z$ şi, folosind propoziţia precedentă,

$$\|z\|_1 = \alpha\|Pz\|_1 \le \alpha\|z\|_1.$$

Cum $\alpha < 1$, rezultă $\|z\|_1 = 0$, deci $z = 0$. Nucleul este trivial, iar în dimensiune finită matricea pătratică este inversabilă. Existenţa şi unicitatea rezultă.

Presupunem acum că $v \ge 0$ şi $e^T v = 1$. Din $\|\alpha P\|_1 = \alpha < 1$ rezultă convergenţa seriei Neumann,

$$(I - \alpha P)^{-1} = \sum_{k=0}^{\infty} \alpha^k P^k.$$

Toate matricile din sumă au elemente nenegative, deci $x^* = (1-\alpha)(I-\alpha P)^{-1}v \ge 0$. Mai mult, deoarece $e^T P = e^T$, înmulţind (14.2) la stânga cu e^T obţinem

$$(1-\alpha)e^T x^* = (1-\alpha)e^T v = 1 - \alpha,$$

deci $e^T x^* = 1$. □

Teoremă 14.2.3 (Convergenţa iteraţiei PageRank)**.** *În ipotezele precedente, pentru orice $x^{(0)} \in \mathbb{R}^n$, iteraţia (14.1) converge la soluţia unică x^* şi verifică estimarea*

$$\|x^{(k)} - x^*\|_1 \leq \alpha^k \|x^{(0)} - x^*\|_1.$$

Demonstraţie. Scădem ecuaţia punctului fix $x^* = (1-\alpha)v + \alpha P x^*$ din iteraţie. Pentru eroarea $e^{(k)} = x^{(k)} - x^*$ obţinem exact

$$e^{(k+1)} = \alpha P e^{(k)}, \qquad e^{(k)} = (\alpha P)^k e^{(0)}.$$

Aplicând norma 1 şi propoziţia anterioară,

$$\|e^{(k+1)}\|_1 \leq \alpha \|e^{(k)}\|_1.$$

Prin inducţie, $\|e^{(k)}\|_1 \leq \alpha^k \|e^{(0)}\|_1$. Cum $\alpha^k \to 0$, rezultă $x^{(k)} \to x^*$. □

Legătura cu raza spectrală

Deoarece $\rho(\alpha P) \leq \|\alpha P\|_1 = \alpha < 1$, criteriul general pentru metode iterative garantează convergenţa. PageRank nu introduce o teorie nouă; este o aplicaţie directă a rezultatelor despre norme induse, rază spectrală şi iteraţii staţionare. Parametrul α controlează simultan modelul şi o limită pentru rata de convergenţă.

14.2.3 Forma spectrală

Definim matricea Google

$$G = \alpha P + (1-\alpha) v e^T.$$

Dacă x este vector de probabilităţi, $e^T x = 1$, iar $Gx = \alpha P x + (1-\alpha)v$. Prin urmare iteraţia PageRank poate fi scrisă $x^{(k+1)} = Gx^{(k)}$, iar vectorul PageRank satisface $Gx^* = x^*$. Am ajuns astfel la calculul unui vector propriu pentru valoarea proprie 1. În implementare nu este necesar să formăm matricea densă ve^T: termenul de teleportare se calculează separat, ceea ce păstrează avantajul structurii rare a lui P.

14.2.4 Exemplu complet cu patru pagini

Considerăm legăturile: pagina 1 trimite către 2 şi 3, pagina 2 trimite către 3, pagina 3 trimite către 1, iar pagina 4 trimite către 1 şi 3. Cu convenţia pe coloane,

$$P = \begin{pmatrix} 0 & 0 & 1 & 1/2 \\ 1/2 & 0 & 0 & 0 \\ 1/2 & 1 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Fie $\alpha = 0.8$, $v = (1/4, 1/4, 1/4, 1/4)^T$ şi $x^{(0)} = v$. Studentul trebuie să poată executa primele iteraţii fără calculator. Calculăm

$$Px^{(0)} = \begin{pmatrix} 3/8 \\ 1/8 \\ 1/2 \\ 0 \end{pmatrix}, \qquad x^{(1)} = 0.2v + 0.8Px^{(0)} = \begin{pmatrix} 0.35 \\ 0.15 \\ 0.45 \\ 0.05 \end{pmatrix}.$$

În continuare,

$$Px^{(1)} = \begin{pmatrix} 0.475 \\ 0.175 \\ 0.35 \\ 0 \end{pmatrix}, \qquad x^{(2)} = \begin{pmatrix} 0.43 \\ 0.19 \\ 0.33 \\ 0.05 \end{pmatrix}.$$

Soluţia sistemului (14.2) este aproximativ

$$x^* \approx (0.37264,\ 0.19906,\ 0.37830,\ 0.05)^T.$$

Observaţi că pagina 4 nu are legături de intrare, astfel încât scorul ei provine numai din teleportare.

Laborator MATLAB: PageRank de la matrice la algoritm

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Folosiţi exemplul cu patru pagini înainte de a trece la grafuri mai mari.

1. Calculaţi manual $x^{(1)}$ şi $x^{(2)}$ şi verificaţi că suma componentelor rămâne 1.
2. Scrieţi pseudocodul iteraţiei, precizând explicit unde apare înmulţirea rară $Px^{(k)}$ şi unde apare teleportarea.
3. Implementaţi funcţia MATLAB

```
function [x,k,hist] = pagerank_iter(P,alpha,v,tol,maxit)
    n = length(v);
    x = ones(n,1)/n;
    hist = zeros(maxit,1);
    for k = 1:maxit
        xnew = alpha*(P*x) + (1-alpha)*v;
        hist(k) = norm(xnew-x,1);
        x = xnew;
        if hist(k) <= tol
            hist = hist(1:k);
            return
        end
    end
    hist = hist(1:maxit);
end
```

4. Verificaţi rezultatul cu soluţia directă, calculată separat prin operatorul MATLAB de rezolvare a unui sistem liniar:

```
xdir = (speye(n)-alpha*P) \ ((1-alpha)*v);
```

5. Raportaţi $\|x - x_{dir}\|_1$, reziduul $\|(I - \alpha P)x - (1 - \alpha)v\|_1$ şi numărul de iteraţii.
6. Repetaţi pentru $\alpha = 0.5, 0.8, 0.9, 0.95$ şi reprezentaţi semilogaritmic istoricul diferenţelor. Explicaţi legătura cu factorul α din estimarea teoretică.

Exerciţii PageRank

1. Demonstraţi direct că, dacă $x^{(0)}$ şi v sunt vectori de probabilităţi, toate iteratele $x^{(k)}$ sunt vectori de probabilităţi.
2. Arătaţi că $e^T G = e^T$, deci G este stocastică pe coloane.
3. Pentru exemplul cu patru pagini calculaţi manual $x^{(3)}$ şi comparaţi cu x^*.
4. Adăugaţi un nod suspendat şi explicaţi de ce coloana nulă distruge interpretarea probabilistică înainte de corecţie.
5. Arătaţi că $\kappa_1(I - \alpha P)$ poate creşte atunci când α se apropie de 1; interpretaţi consecinţa asupra rezolvării directe.
6. De ce nu este indicat să formăm explicit G pentru un graf foarte mare, chiar dacă formula este simplă?
7. Explicaţi diferenţa dintre criteriul de oprire $\|x^{(k+1)} - x^{(k)}\|_1 \leq \texttt{tol}$ şi eroarea necunoscută $\|x^{(k)} - x^*\|_1$.
8. Formulaţi o versiune personalizată alegând un vector v concentrat pe două pagini şi anticipaţi înainte de calcul cum se va modifica ordinea de relevanţă.

Proiect de nivel mediu: PageRank pe un graf construit de student

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi un graf orientat cu 10–20 de noduri, suficient de mic pentru a fi inspectat, dar suficient de mare pentru ca iteraţia să nu fie trivială.

1. Desenaţi graful şi selectaţi un subgraf de patru noduri pe care efectuaţi manual o iteraţie completă.
2. Construiţi matricea de adiacenţă şi apoi matricea stocastică P în format `sparse`. Trataţi explicit nodurile suspendate.
3. Implementaţi PageRank fără a forma matricea densă G.
4. Comparaţi iteraţia cu rezolvarea directă a sistemului pentru dimensiunea aleasă şi explicaţi de ce această comparaţie nu ar rămâne realistă la scară foarte mare.
5. Studiaţi cel puţin patru valori ale lui α şi două alegeri pentru v, dintre care una personalizată.
6. Modificaţi o singură legătură a grafului şi măsuraţi schimbarea vectorului PageRank. Identificaţi nodurile cele mai sensibile.
7. Raportul final trebuie să conţină matricea sau graful, pseudocodul, codul MATLAB, un tabel cu numărul de iteraţii şi reziduul, două grafice şi o explicaţie matematică a rezultatelor.

14.3 Un model de recomandare prin factorizare de rang mic

Într-un model simplificat, liniile unei matrice $R \in \mathbb{R}^{m\times n}$ corespund utilizatorilor, coloanele corespund unor obiecte (filme, melodii, produse), iar numai o parte dintre valorile R_{ij} sunt observate. Presupunem că preferinţele pot fi descrise prin k factori latenţi, cu $k \ll \min(m,n)$, şi că

$$R_{ij} \approx u_i^T v_j,$$

unde $u_i, v_j \in \mathbb{R}^k$. Pentru mulţimea Ω a intrărilor observate considerăm funcţia regularizată

$$F(U,V) = \frac{1}{2}\sum_{(i,j)\in\Omega}(R_{ij} - u_i^T v_j)^2 + \frac{\lambda}{2}\left(\sum_i \|u_i\|_2^2 + \sum_j \|v_j\|_2^2\right), \qquad \lambda > 0. \tag{14.3}$$

Problema nu este liniară simultan în U şi V, dar se reduce la problema celor mai mici pătrate cu regularizare dacă una dintre cele două familii de vectori este fixată.

Propoziţie 14.3.1 (Un pas ALS)**.** *Fixăm vectorii v_j. Pentru un utilizator i, fie Ω_i mulţimea obiectelor observate, V_i matricea ale cărei linii sunt v_j^T, $j \in \Omega_i$, şi r_i vectorul scorurilor observate. Minimizatorul unic în raport cu u_i satisface*

$$(V_i^T V_i + \lambda I)u_i = V_i^T r_i.$$

Matricea sistemului este simetrică pozitiv definită.

Demonstraţie. Termenii din (14.3) care depind de u_i sunt

$$\phi_i(u) = \frac{1}{2}\|V_i u - r_i\|_2^2 + \frac{\lambda}{2}\|u\|_2^2.$$

Derivând,

$$\nabla\phi_i(u) = V_i^T(V_i u - r_i) + \lambda u.$$

Condiţia de staţionaritate dă sistemul din enunţ. Pentru orice $z \neq 0$,

$$z^T(V_i^T V_i + \lambda I)z = \|V_i z\|_2^2 + \lambda\|z\|_2^2 > 0,$$

deci matricea este SPD. Funcţia ϕ_i este strict convexă, iar punctul staţionar este minimizatorul unic. □

Propoziţie 14.3.2. *Dacă într-un algoritm ALS minimizăm exact succesiv funcţia F în raport cu U pentru V fixat şi apoi în raport cu V pentru U fixat, valorile funcţiei obiectiv nu cresc de la un semi-pas la următorul.*

Demonstraţie. La primul semi-pas alegem U^{k+1} ca minimizator al funcţiei $U \mapsto F(U, V^k)$. Prin definiţia minimului,

$$F(U^{k+1}, V^k) \le F(U^k, V^k).$$

La al doilea semi-pas alegem V^{k+1} ca minimizator al lui $V \mapsto F(U^{k+1}, V)$, deci

$$F(U^{k+1}, V^{k+1}) \le F(U^{k+1}, V^k).$$

Combinând cele două inegalităţi obţinem monotonia valorilor obiectiv. Aceasta nu afirmă, de una singură, că problema globală neconvexă are un minimizator unic sau că ALS converge la minimul global. □

Un pas calculat cu pixul pe hârtie

Presupunem $k = 2$, iar pentru un utilizator avem două observaţii $r = (5, 3)^T$ şi vectorii obiectelor $v_1 = (1, 0)^T$, $v_2 = (1, 1)^T$. Pentru $\lambda = 1$,

$$V_i = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad V_i^T V_i + I = \begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}, \quad V_i^T r = \begin{pmatrix} 8 \\ 3 \end{pmatrix}.$$

Rezolvând sistemul obţinem $u = (13/5, 1/5)^T$. Acest calcul mic conţine exact operaţiile repetate în ALS: construirea problemei celor mai mici pătrate cu regularizare şi rezolvarea unui sistem SPD, pentru care Cholesky este o alegere naturală.

Laborator MATLAB: factorizare matricială prin ALS

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Folosiţi o matrice mică de scoruri, cu valori lipsă marcate prin `NaN`.

1. Pentru un singur utilizator şi factori V fixaţi, construiţi manual V_i, r_i şi sistemul normal regularizat.
2. Scrieţi două funcţii MATLAB, una care actualizează toţi vectorii u_i şi una care actualizează toţi vectorii v_j. Pentru sistemele SPD folosiţi propria rutină Cholesky sau operatorul \ numai după ce aţi justificat forma sistemului.
3. Iniţializaţi U, V cu valori mici aleatoare şi efectuaţi 20–50 de iteraţii ALS.
4. Calculaţi după fiecare semi-pas funcţia F numai pe intrările observate şi verificaţi monotonia demonstrată mai sus.
5. Separaţi câteva observaţii înainte de antrenare şi folosiţi-le ca set de test. Raportaţi eroarea pe datele observate şi eroarea de test.
6. Repetaţi pentru două valori ale rangului k şi trei valori ale lui λ. Explicaţi compromisul dintre potrivirea datelor şi regularizare.

Proiect de nivel mediu: recomandare pe date sintetice

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Generaţi factori "adevăraţi" $U_0 \in \mathbb{R}^{30\times 3}$ şi $V_0 \in \mathbb{R}^{20\times 3}$, formaţi $R_0 = U_0 V_0^T$ şi adăugaţi zgomot moderat. Ascundeţi aleator 30% dintre intrări.

1. Explicaţi de ce matricea fără zgomot are rang cel mult 3 şi verificaţi această afirmaţie prin SVD.
2. Implementaţi ALS folosind numai intrările de antrenare.
3. Comparaţi rangurile $k = 1, 2, 3, 5$ şi cel puţin trei valori ale lui λ.

4. Reprezentaţi eroarea de antrenare şi eroarea pe intrările ascunse. Identificaţi un caz în care o potrivire mai bună pe datele observate nu produce o predicţie mai bună.
5. Comparaţi cu o aproximaţie SVD de rang mic în cazul în care toate intrările sunt cunoscute şi explicaţi de ce problema cu date lipsă este diferită.
6. Raportul trebuie să conţină derivarea pasului ALS, pseudocod, cod MATLAB, rezultatele numerice şi o discuţie despre rolul rangului şi al regularizării.

14.4 Ce noţiuni din curs au fost folosite?

În PageRank am folosit matrice rare, norme induse, raza spectrală, convergenţa puterilor unei matrice, sisteme liniare şi metode iterative. În modelul de recomandare am folosit aproximări de rang mic, cele mai mici pătrate, regularizare, sisteme SPD, Cholesky şi optimizare alternantă. Acesta este motivul pentru care cele două aplicaţii sunt plasate la final: ele nu reprezintă exemple izolate, ci puncte în care ideile dezvoltate în curs se întâlnesc.

Probleme de sinteză

1. Comparaţi conceptual rolul lui α în PageRank cu rolul lui λ în regularizarea Tikhonov. În ce sens ambii parametri schimbă simultan modelul şi comportamentul numeric?
2. Identificaţi în proiectul ALS locurile în care ar putea fi folosite Cholesky, QR sau SVD şi explicaţi avantajul fiecărei alegeri.
3. În PageRank, ce informaţie oferă reziduul sistemului liniar şi ce informaţie oferă diferenţa dintre două iterate consecutive?
4. În recomandare, de ce problema este convexă în U cu V fixat, dar nu este convexă simultan în (U, V)?
5. Formulaţi o regulă generală pentru alegerea dintre o metodă directă şi una iterativă pe baza structurii, dimensiunii şi numărului de rezolvări necesare.

Test de capitol

Test – PageRank şi sisteme de recomandare

1. PageRank pe o reţea mică. Se consideră trei pagini. Pagina 1 indică spre paginile 2 şi 3, pagina 2 indică numai spre pagina 3, iar pagina 3 indică numai spre pagina 1. Construiţi matricea de tranziţie coloană-stochastică P conform convenţiei din capitol. Pentru $\alpha = 0.8$ şi vector de teleportare uniform $v = (1/3, 1/3, 1/3)^T$, scrieţi sistemul liniar satisfăcut de vectorul PageRank p şi efectuaţi o iteraţie pornind din $p^{(0)} = v$.

2. Selectaţi afirmaţia/afirmaţiile corecte şi justificaţi.

A) parametrul $\alpha < 1$ contribuie la contractivitatea iteraţiei PageRank;

B) o implementare pentru grafuri mari ar trebui să exploateze caracterul rar al matricei;

C) în factorizarea de recomandare $R \approx UV^T$, fixarea lui V conduce, pentru fiecare factor din U, la problema celor mai mici pătrate;

D) în ALS, problema este simultan liniară în toate elementele lui U şi V luate împreună.

Noduri suspendate

Fie D mulţimea paginilor fără legături de ieşire şi v vectorul de teleportare. Nu este necesar să înlocuim explicit fiecare coloană nulă prin vectorul dens v, deoarece matricea ar deveni mai puţin rară. Dacă P conţine numai tranziţiile efective, produsul corectat se aplică sub forma

$$Px + \left(\sum_{j \in D} x_j \right) v.$$

Termenul suplimentar are rang unu şi se calculează fără a forma coloane dense în matricea de tranziţie.

În loc de încheiere

Sinteză finală şi proiect integrator

Manualul a urmărit aceeaşi idee în contexte aparent diferite: exploatarea structurii. Pentru o matrice triunghiulară folosim substituţia; pentru o matrice generală, LU cu pivotare; pentru SPD, Cholesky; pentru structură tridiagonală, Thomas; pentru probleme mari şi rare, metode iterative; pentru cele mai mici pătrate, QR; pentru rang şi sensibilitate, SVD; pentru probleme neliniare de tip sumă de pătrate, Gauss–Newton; iar în aplicaţiile integratoare, aceste instrumente se reunesc în PageRank şi în modele simple de recomandare.

Tabel de referinţă: costuri dominante

Costurile de mai jos sunt ordine de mărime sau termeni dominanţi; constantele inferioare depind de implementare şi de modul de numărare a operaţiilor.

Metodă	Ipoteze / structură	Cost dominant
Substituţie triunghiulară	diagonală nenulă	n^2
LU cu pivotare parţială	A nesingulară	$\frac{2}{3}n^3$
LDL^T	A simetrică	$\frac{1}{3}n^3$
Cholesky	A SPD	$\frac{1}{3}n^3$
Thomas	tridiagonală	$O(n)$
Bandă, semilăţimi p, q	matrice de bandă	$O(npq)$
QR Householder ($m \geq n$)	dreptunghiulară	$2mn^2 - \frac{2}{3}n^3$
Ecuaţii normale + Cholesky	rang coloană complet	$mn^2 + \frac{1}{3}n^3$
SVD densă	fără structură specială	$O(mn \min\{m, n\})$
Jacobi / Gauss–Seidel	pe iteraţie	$O(\mathrm{nnz}(A))$

Pentru $m \gg n$, ecuaţiile normale sunt mai ieftine decât QR, dar verifică $\kappa_2(A^TA) = \kappa_2(A)^2$. Din acest motiv, QR este adesea preferată atunci când condiţionarea are un rol important.

Întrebările pe care trebuie să le puneţi înainte de a alege metoda

1. Care sunt dimensiunile şi structura lui A? Este densă, rară, simetrică, SPD, de bandă, dreptunghiulară?
2. Am nevoie de o singură rezolvare sau de mulţi termeni liberi?
3. Problema este bine condiţionată? Datele conţin zgomot?
4. Este suficientă o soluţie aproximativă la o toleranţă prescrisă?
5. Pot aplica matricea unui vector fără a o stoca explicit?
6. Ce măsură de eroare are sens în aplicaţie?

Alegerea metodei numerice începe cu aceste întrebări, nu cu numele unui algoritm.

Probleme de sinteză

1. Aveţi o matrice SPD densă de ordin 5000 şi 100 de termeni liberi. Ce metodă alegeţi şi ce reutilizaţi?
2. Aveţi o matrice rară (sparse) de ordin 10^7 disponibilă doar prin produsul $v \mapsto Av$. Ce metode din manual rămân conceptual posibile şi care nu?
3. Aveţi o problemă LS cu $m = 10^5$, $n = 20$, dar coloane aproape dependente. Comparaţi ecuaţiile normale, QR şi SVD.
4. Un algoritm produce reziduu relativ 10^{-13}, dar soluţia pare greşită. Enumeraţi cel puţin trei diagnostice pe care le-aţi face.
5. Pentru o problemă de localizare neliniară, explicaţi cum se combină modelarea, Jacobianul, QR şi căutarea pasului (line search) într-un singur algoritm Gauss–Newton robust.

Proiect integrator de semestru

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Alegeţi o aplicaţie care produce fie un sistem liniar mare, fie problema celor mai mici pătrate, eventual cu regularizare. Proiectul trebuie să conţină: derivarea modelului matematic; identificarea structurii matricei; cel puţin două metode numerice comparate; analiză de cost; reziduu şi eroare atunci când aceasta poate fi estimată; studiu de condiţionare; cod reproductibil; un experiment în care una dintre metode se comportă slab şi explicaţia matematică a eşecului. Exemple potrivite sunt discretizarea unei EDP, calibrarea unui model, denoising, localizare, compresie de imagine, un graf PageRank de dimensiune moderată sau o problemă sintetică de recomandare.

Fişă finală: compresia imaginilor ca proiect de calcul ştiinţific

Problema rezolvată 1: eroarea optimă la rang fix

Dacă valorile singulare ale unei imagini-matrice sunt $\sigma_1 \geq \cdots \geq \sigma_r$, teorema Eckart–Young spune că aproximaţia trunchiată $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ minimizează atât eroarea spectrală, cât şi eroarea Frobenius între toate matricile de rang cel mult k. Mai precis,

$$\|A - A_k\|_2 = \sigma_{k+1}, \qquad \|A - A_k\|_F^2 = \sum_{i>k} \sigma_i^2.$$

Aşadar, odată ce rangul k este fixat, nu există o altă matrice de rang k care să aibă eroare Frobenius mai mică. Dificultatea practică nu este găsirea celei mai bune aproximaţii pentru k fix, ci alegerea lui k şi relaţia dintre eroarea matricială, stocare şi calitatea vizuală.

Problema rezolvată 2: prag energetic comun

Presupunem că pentru canalul de luminanţă primele 50 de valori singulare păstrează 99.5% din energia Frobenius, iar pentru crominanţe primele 15 păstrează 99%. O strategie justificată este $(k_Y, k_{Cb}, k_{Cr}) = (50, 15, 15)$, nu ranguri egale. Dacă imaginea are dimensiunea $m \times n$, costul idealizat este $(50 + 15 + 15)(m + n + 1)$ coeficienţi. Această alegere adaptează bugetul la structură.

Laborator integrator 12: flux complet de compresie

Paşi obligatorii de lucru. 1. Refaceţi mai întâi pe hârtie relaţiile şi cel puţin un pas numeric. **2.** Scrieţi algoritmul în pseudocod şi precizaţi datele de intrare, ieşire şi criteriul de oprire, dacă este cazul. **3.** Implementaţi în MATLAB fără a apela direct rutina care rezolvă exact sarcina studiată, exceptând etapa de verificare. **4.** Testaţi pe un caz cu rezultat cunoscut. **5.** Comparaţi cu o rutină MATLAB de referinţă folosind numai mărimi şi criterii definite până în acel punct al cursului. **6.** Formulaţi în câteva propoziţii ce a făcut algoritmul şi de ce rezultatul este credibil.

Realizaţi o funcţie care primeşte o imagine şi un buget sau un prag energetic. Implementaţi trei moduri: tonuri de gri, RGB pe canale şi YCbCr. Programul trebuie să raporteze rangurile, stocarea idealizată, eroarea Frobenius, PSNR şi timpul SVD. Salvaţi reconstrucţiile şi spectrele singulare. Testaţi minimum cinci imagini: portret, peisaj, text/grafică, imagine cu zgomot şi imagine cu zone uniforme.

Set final de probleme

1. Demonstraţi formula erorii Frobenius pentru A_k folosind ortogonalitatea termenilor singulari.
2. Calculaţi stocarea pentru o imagine 1920×1080 la rang $k = 100$ în tonuri de gri.
3. Pentru RGB cu ranguri diferite, formulaţi problema alocării unui buget total pentru minimizarea erorii Frobenius totale.
4. De ce o imagine cu zgomot alb poate avea valori singulare care scad mai lent?
5. Ce diferenţă este între „compresie matematică” prin numărul de coeficienţi şi dimensiunea reală a unui fişier comprimat?
6. De ce SVD global nu este în mod necesar competitiv cu codecurile specializate pentru imagini?
7. Explicaţi ce informaţie structurală oferă totuşi experimentul SVD despre imagine.
8. Propuneţi o extensie către video sau către un tensor de ordin superior şi discutaţi ce dificultăţi noi apar.

Proiect final de curs

Etapele proiectului. 1. Modelare: definiţi toate mărimile şi termenii specifici aplicaţiei înainte de a-i folosi şi formulaţi problema matematică. **2. Derivare:** justificaţi metoda şi executaţi manual un exemplu mic. **3. Algoritm:** prezentaţi pseudocodul şi analizaţi costul. **4. MATLAB:** scrieţi o implementare proprie, modulară. **5. Experimente:** variaţi cel puţin un parametru şi comparaţi cu o metodă de referinţă. **6. Validare:** folosiţi numai criterii definite până în acel punct al cursului. **7. Explicaţie:** includeţi o concluzie matematică, nu doar grafice sau tabele.

Construiţi un raport reproductibil care compară cel puţin două strategii SVD de compresie pe minimum zece imagini. Includeţi criteriu de alegere a rangului, rată de stocare, eroare, timp şi observaţii vizuale. Introduceţi o secţiune obligatorie „când metoda nu funcţionează

bine”, cu exemple şi explicaţii. Bonus: compараţi cu JPEG la dimensiuni de fişier apropiate, subliniind că obiectivele şi reprezentările algoritmilor sunt diferite.