Rezolvarea sistemelor de ecuații liniare. Factorizarea QR

Profeanu Ioana

Universitatea Politehnica din București
Facultatea de Automatică și Calculatoare
Grupa: 323CA
ioana.profeanu@gmail.com

Abstract. Subiectul documentului de față are la bază rezolvarea sistemelor de ecuații liniare utilizând metode numerice directe și analiza acestora în funcție de tipul și mărimea datelor care trebuie prelucrate. Algoritmii numerici aleși sunt Gram-Schmidt, Givens și Householder, ce fac parte din categoria metodelor de factorizare QR.

Keywords: Givens · Householder · Gram-Schmidt · Metode QR

1 Introducere

1.1 Descrierea problemei rezolvate

Un set finit de ecuații liniare simultane ale căror soluții trebuie aflate poartă numele de sistem de ecuații liniare.

Un sistem de n ecuații liniare cu m necunoscute este de forma:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1m}x_m = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2m}x_m = b_2 \\ ..... \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nm}x_m = b_n \end{cases}$$

Făcând notațiile:
A = matricea coeficienților,
x = vectorul coloană al necunoscutele sistemului,
b = vectorul coloană al termenilor liberi,
sistemul anterior devine, sub formă matriceală:

$$\mathrm{Ax} = \mathrm{b},$$

unde

$$A_{m,n} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix}, \; x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}, \; b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Problema își propune aflarea valorii necunoscutelor sistemului $x_1, x_2, x_3, ...x_{m-1}, x_m$, ce alcătuiesc vectorul coloană x. [1]

1.2 Aplicații practice ale sistemelor de ecuații liniare

Deși sistemele de ecuații liniare sunt utilizate în varii domenii, precum *matematica*, *fizica*, *domeniul financiar*, *telecomunicații* și *automatizări*, o aplicabilitate majoră este regăsită în *machine learning*. Algebra liniară și componentele sale stau la baza lucrului cu date și prepararea datelor, cum ar fi *One-Hot Encoding*, *Image Processing*, *Singular-Value Decomposition*, *Linear Regression*, unele metode și notații fiind înrădăcinate în submetode precum *Deep learning*, *Natural Language Processing* și *Recommender Systems* [2].

1.3 Specificarea soluțiilor alese

În rezolvarea ecuației liniare $Ax = b$, am ales utilizarea a trei algoritmi ce fac parte din categoria metodelor exacte, care furnizează soluția exactă a sistemului dacă se neglijează erorile de rotunjire [3] și se bazează pe factorizarea ortogonală: *Gram-Schmidt, Givens și Householder*.

Factorizarea ortogonală (QR) reprezintă scrierea matricei A ca produs dintre o matrice ortogonală (Q) și una superior triunghiulară (R), rezolvarea ulterioară a sistemului bazându-se pe proprietatea că într-o matrice ortogonală, transpusa este egală cu inversa, urmând apoi aflarea soluțiilor utilizând backward-substitution.

Gram-Schmidt

Algoritmul *Gram-Schmidt* este utilizat pentru a găsi o bază ortogonală dintr-o bază neortogonală. Procesul se bazează pe găsirea proiecției ortogonale q_i a fiecărui vector coloană a_i, scăzând apoi suma proiecțiilor vectorilor deja calculați. Vectorul rezultat este apoi împărțit la lungimea sa, obținându-se vectorul unitar e_i (procesul de normare), ce reprezintă coloanele matricei ortogonale Q.

Deși algoritmul Gram-Schmidt este cel mai ușor de implementat dintre cei trei algoritmi prezentați, este și cel mai instabil numeric, împărțirea la normă generând erori. În consecință, este eficient pe matrice dense, cu cât mai puține zero-uri, și de dimensiuni cât mai reduse.

Givens

Metoda numerică *Givens* utilizează matrice de rotație ortogonale pentru a elimina elementele de sub diagonala principală, fiecare rotație formând câte un zero pentru a forma matricea R. Concatenarea tuturor rotațiilor Givens vor forma ulterior matricea ortogonală Q. Matricea A va fi descompusă astfel:

$$A = G^T R,$$
$$\text{unde } G = G_{n-1,m} G_{n-2,m} G_{n-2,m-1} ... G_{1n} ... G_{13} G_{12}, \quad R = GA$$

Deoarece scopul algoritmului este acela de a forma zero-uri sub diagonala principală, acesta este cel mai eficient atunci când A este matrice rară.

Householder

Metoda de triunghiularizare ortogonală *Householder* are la bază utilizarea reflectorilor Householder, al căror produs reprezintă matricea ortogonală H:

$$H = H_{min(m-1,n)} \cdot ... \cdot H_p \cdot ... \cdot H_2 \cdot H_1, H_p = I_m - 2\frac{v_p v_p^T}{v_p^T v_p} \text{ [6]}$$

Astfel, sistemul liniar de ecuații $Ax = b, A \in R^{mxn}$ se transformă într-un sistem ortogonal echivalent, $HAx = HB$, unde matricea sistemului HA este superior triunghiulară, iar $H \in R^{mxm}$ este o matrice ortogonală.

Metoda numerică Householder are cea mai bună stabilitate numerică dintre cei trei algoritmi discutați, datorită mecanismului de a produce zero-uri prin intermediul reflectorilor, însă pot apărea erori de aproximare în cazul matricelor rare de dimensiuni mari. În practică, este folosit mai ales în cazul matricelor dense.

1.4 Criteriile de evaluare pentru soluția propusă

Testarea algoritmilor se va face pe un eșantion cu mai multe seturi de teste, dorind să observăm acuratețea soluțiilor, complexitatea și eficiența implementărilor. Evaluarea va fi concepută astfel încât să acopere o sferă cât mai largă de cazuri, inclusiv cele în care algoritmii nu funcționează sau dau rezultate eronate.

Input-ul va fi dat sub forma unui vector coloană b al termenilor liberi și matricea A a coeficienților, variind conținutul acesteia astfel:

- dimensiunea: pornim de la matrice de dimensiuni de ordinul unităților până la cele de ordinul sutelor, pentru a putea vedea acumularea de erori în calcul cauzate de rotunjirea soluției, caracter ce crește concomitent cu dimensiunea matricei;
- tipul matricei: rară sau densă; așa cum am menționat anterior, algoritmii Householder și Gram-Schmidt au o eficiență mai mare în cazul matricelor cu puține zero-uri, în timp ce Givens este mai eficient cu cât sunt mai puține numere nenule în matrice;
- tipul sistemului de ecuații: subdimensionat / supradimensionat / pătratic, compatibil determinat / nedeterminat / incompatibil;

Output-ul va fi reprezentat de vectorul coloană x al soluțiilor, obținut după efectuarea algoritmului de backward-substitution. Timpii de execuție vor fi măsurați inițial doar pentru procesul de factorizare QR al fiecărui algoritm, ulterior măsurând și timpul de execuție al algoritmului backward-substitution.

2 Prezentarea soluțiilor

2.1 Descrierea funcționării algoritmilor aleși

Overview: Așa cum am menționat anterior, algoritmii prezentați au scopul de a scrie matricea sistemului A ca fiind un produs între o matrice ortogonală (Q) și una superior triunghiulară (R). Vectorul coloană x al soluțiilor este ulterior aflat cu ajutorul metodei backward-substitution, ce se bazează pe egalitatea dintre transpusă și inversă într-o matrice ortogonală. Astfel, soluțiile se vor afla astfel:

$$Q^t = Q^{-1}$$
$$Ax = b \Rightarrow QR \cdot x = b \Rightarrow R \cdot x = Q^{-1} \cdot b \Rightarrow R \cdot x = Q^t \cdot b$$

Efectuând înmulțirea din membrul drept, noua ecuație se va rezolva folosind backward-substitution, deoarece matricea R este superior triunghiulară. Practic, esența rezolvării ecuațiilor liniare utilizând factorizarea QR se rezumă la modalitatea de obținere a celor două matrice, lucru posibil utilizând unul dintre cei trei algoritmi menționați anterior.

Gram-Schmidt

Scopul principal al algoritmului Gram-Schmidt este acela de a găsi o bază ortogonală dintr-o bază neortogonală, bazele ortogonale având proprietăți importante care sunt de dorit pentru calculele ulterioare. Algoritmul ia un set de k vectori liniari independenți, $v_i, 1 \leq i \leq k$ și construiește o bază ortonormată pe același subspațiu vectorial. Proiecția vectorului v se va calcula astfel:

$$proj_v = (\frac{<v,u>}{\|u\|_2^2 u})$$

Vectorul $v - proj_u(v)$ este ortogonal față de u și reprezintă baza algoritmului Gram-Schmidt. Pornind de la primul vector, v_1, îl normăm, îl vom numi e_1 și formăm $u_2 = e_1 - proj_{e1}(v_2)$, iar $e_2 = \frac{u_2}{\|u\|_2}$. Continuăm procesul prin scăderea proiecțiilor v_i din e_j, apoi normând pentru a obține e_i. Iterăm până la obținerea unui set ortonormat $e_1, e_2, \cdots, e_k$.

Matematic, fiecare pas incremental $k+1$ este descris de formula:

$$u_{k+1} = a_{k+1} - (a_{k+1} \cdot e_1)e_1 - ... - (a_{k+1} \cdot e_k)e_k, e_{k+1} = \frac{u_{k+1}}{||u_{k+1}||} \text{ [4]}$$

Ulterior, matricea A va fi factorizată astfel:

$$A = [a_1|a_2|...|a_n] = [e_1|e_2|...e_n] \cdot \begin{pmatrix} a_1 \cdot e_1 & a_2 \cdot e_1 & \cdots & a_n \cdot e_1 \\ 0 & a_2 \cdot e_2 & \cdots & a_n \cdot e_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \cdot e_n \end{pmatrix} = Q \cdot R \text{[4]}$$

Deoarece scăderea proiectiilor lui v_i din e_j într-o singură operație cauzează instabilitate numerică, a fost implementat algoritmul MGS (Modified Gram-Schmidt), care împarte implementarea pe bucăți mai mici, scăzându-se proiecțiile una câte una. Astfel se ajunge la aceeași soluție așteptată, însă erorile de rotunjire sunt mai mici față de algoritmul original.

Givens

Factorizarea QR folosind rotațiile Givens au la bază faptul că fiecare rotație formează 0 în locul unui element subdiagonal din matrice, formându-se matricea R, iar concatenarea tuturor rotațiilor Givens ducând la formarea matricei ortogonale Q.

Considerând A matrice de dimensiune $m \times n$, iar c și s constante, o matrice $m \times n$ Givens G (i, j, c, s) $i < j$, numită și rotație Givens, plasează c la indicii (i, i) și (j, j), -s la indicii (j, i) și s la indicii (i, j) în matricea identitate. G (i, j, c, s) este ortogonală, iar printr-o alegere atentă de constante, rotația G (i, j, c, s) a matricei A va avea efect doar asupra liniilor i și j din A și va forma zero-uri pe elementele a_{ji}.

$$G(i,j,\theta) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}$$

Produsul este format explicit prin schimbarea liniilor i și j, deci nu este neapărat necesară construirea unei matrice Givens. Făcând zero-uri în locul elementelor de sub diagonala principală, algoritmul Givens produce o matrice superior triunghiulară R prin factorizare. Păstrând produsul matricelor Givens, putem afla Q.

Householder

Reflectorul Householder este o transformare care reflectă un vector într-un plan sau hiperplan și poate fi utilizat pentru a calcula factorizarea QR a unei matrice A de dimensiune $m \times n$, cu $m \geq n$.

Dacă u este un vector de dimensiune $m \times 1$, matricea Householder este ortogonală și simetrică.

$$H_u = I_m - 2\frac{uu^T}{u^T u} \text{ [6]}$$

Produsele $H_u v$, $H_u A$ și AH_u, unde A este matricea sistemului și v este un vector de dimensiune $m \times 1$, pot fi calculate implicit fară a construi H_u. Prin alegerea unei valori convenabile pentru u, $H_u A$ va forma zero-uri în locul elementelor de sub diagonala principală, mai exact pentru fiecare element subdiagonal de pe coloana j din $A_{i-k} := H_{j-1} \cdot H_{j-2} \cdot \cdots \cdot H_2 \cdot H_1 \cdot A$.

În consecință, putem începe prin setarea unui pivot (luând, la rând, fiecare element de pe diagonala principală), luăm vectorul coloană ce îl cuprinde și eliminăm toate elementele de sub pivot. Astfel, pentru pasul k, vom avea matricea Householder H_k. Rezultatul aplicării algoritmului va fi obținerea unei matrice superior triunghiulare R, Q aflându-se ulterior din relația $R = Q^t A$.

2.2 Analiza complexității soluțiilor

Backward-substitution

Deoarece toți cei trei algoritmi utilizează algoritmul de backward-substitution pentru găsirea vectorului soluție x, vom discuta succint despre complexitatea acestui algoritm.

$$\begin{gathered} x_n = b_n/A_{nn} \\ \mathrm{x}_{n-1} = (b_{n-1} - A_{n-1,n}x_n)/A_{n-1,n-1} \\ \mathrm{x}_{n-2} = (b_{n-2} - A_{n-2,n-1}x_{n-1} - A_{n-2,n}x_n)/A_{n-2,n-2} \\ \cdots \\ \mathrm{x}_1 = (b_1 - A_{1,2}x_2 - A_{1,3}x_3 - \cdots - A_{1,n}x_n)/A_{1,1} \text{ [10]} \end{gathered}$$

Se observă cu ușurință că algoritmul are complexitatea de O(n), unde n este dimensiunea matricei A a sistemului.

Gram-Schmidt

Complexitatea algoritmului poate fi dedusă din pseudocodul său:

$$\begin{gathered} \text{for k = 1, n} \\ R1k = q_1^T ak \\ \text{R2k} = \mathrm{q}_2^T ak \\ \cdots \\ \text{Rk-1, k} = \mathrm{q}_{k-1}^T ak \\ \tilde{\mathrm{q}} = \mathrm{a}_k - (R_{1k}q_1 + R_{2k}q_2 + \cdots + R_{k-1}kQ_{k-1} \\ \text{Rk, k} = \|\tilde{q}\| \\ \mathrm{q}_k = \frac{1}{R_{kk}}\tilde{q}_k \text{ [10]} \end{gathered}$$

Pentru al k-lea pas din algoritm avem $(4m-1)(k-1)+3m$ pași, obținuți din următoarele operații:

- k - 1 produse interne cu a_k : $(k-1)(2m-1)$ pași
- calculul lui $\tilde{\mathrm{q}}$: $2(k-1)$ pași
- calculul lui R_{kk} și q_k: $(4m-1)(k-1)+3m$ pași

Astfel, considerând că matricea sistemului A are dimensiunea $m \times n$, vom avea complexitatea:

$$(4m-1)\tfrac{n(n-1)}{2} + 3mn \approx 2mn^2 \text{ pași} \Rightarrow O(2\mathrm{mn}^2).$$

Pentru o matrice pătratică de dimensiuni $n \times n$, complexitatea va avea valoarea $\mathrm{O}(n^3)$.

Householder

Urmărind pașii pe care trebuie să îi urmăm în vederea aplicării algoritmului, putem deduce complexitatea acestuia astfel:

for k = 1, n
1. definim $y = A_{k:m,k}$ și calculăm $(m-k+1)-$vector v_k:
$w = y + sign(y_1)\|y\|e_1$, $v_k = \frac{1}{\|w\|}w$
2. înmulțim $A_{k:m,k:n}$ cu reflectorul $I - 2v_kv_k^T$:
$A_{k:m,k:n} := A_{k:m,k:n} - 2v_k(v_k^T A_{k:m,k:n})$ [10]

Pentru al k-lea pas din algoritm avem $4(m-k+1)(n-k+1)$ pași, obținuți din următorii termeni dominanți:

- $(2(m-k+1)-1)(n-k+1)$ pași pentru produsul $v_k^T A_{k:m,k:n}$
- $(m-k+1)(n-k+1)$ pași pentru produsul exterior v_k
- (m - k + 1)(n - k + 1) pași pentru scăderea din $A_{k:m,k:n}$

Astfel, considerând că matricea sistemului A are dimensiunea $m \times n$, vom avea complexitatea:

$$\int_0^n 4(m-t)(n-t)dt \approx 2mn^2 - \frac{2}{3}n^3 \text{ pași} \Rightarrow O(2\text{mn}^2).$$

Pentru o matrice pătratică de dimensiuni $n \times n$, complexitatea va avea valoarea $O(n^3)$.

Una din concluziile pe care le putem trage este faptul că pentru o matrice pătratică, algoritmul Householder este mai eficient decât Gram-Schmidt, însă în cazul în care $m > n$, eficiența temporală este similară.

Givens

Pseudocodul algoritmului este:

Q = I
R = A for j = 1, n
for i = m: -1: j + 1
[c, s] = givens$(r_{i-1,j}, r_{i,j})$
$R = G(i, j, c, s,)^T R$
$Q = QG(i, j, c, s,)$

Având în vedere algoritmul prezentat, cât și faptul că algoritmul QR bazat pe rotații Givens este analog algoritmului bazat pe reflexii Householder, diferența stând în faptul că atunci când anulăm coloana i se anulează un element la un moment dat [11], putem afirma că cei doi algoritmi au aceeași complexitate, rezultând deci că și algoritmul Givens are o complexitate de $O(2mn^2)$ pentru matrice de dimensiunea $m \times n$, și $O(n^3)$ pentru matrice de dimensiunea $n \times n$.

Cu toate acestea, numărul de pași necesari pentru o matrice de dimensiune $m \times n$ este $3mn^2 - n_3$, aproape cu 50% mai mult față de Householder, rezultând faptul că dintre cei trei algoritmi, prin Householder se va la matricile Q și R în cel mai mic număr de pași. [12]

2.3 Avantajele și dezavantajele soluțiilor

Gram-Schmidt

Avantaje: Principalul avantaj al algoritmului Gram-Schmidt provine din ușurința de implementare, fiind un algoritm util pentru prototipare atunci când nu avem la dispoziție biblioteci pentru algebră liniară deja implementate. De asemenea, folosirea algoritmului este recomandată în cazul în care avem nevoie de forma ortogonală directă.

Dezavantaje: Dezavantajul algoritmului provine din faptul că, în timpul execuției, vectorii u_i de multe ori nu sunt ortogonali din cauza erorilor de rotunjire, iar în alte cazuri, unii vectori sunt aproape liniar dependenți. De aceea, algoritmul Gram-Schmidt clasic este considerat instabil numeric. Față de celelalte metode, depinde de condiția numerică a matricei A dacă matricea Q va avea coloane ortonormate sau nu. Similar, deși algoritmul Gram-Schmidt modificat prezintă o variantă cu mai puține erori a algoritmului original deoarece putem corecta erorile la fiecare pas, rămâne însă problema că stabilitatea numerică depinde de condiția numerică a matricei A.

Givens

Avantaje: Stabilitatea numerică a algoritmului reprezintă cel mai mare avantaj al său, analiza sa perturbațională neimplicând condiția numerică a matricei A. De asemenea, deoarece datorită premultiplicărilor făcute de matricea Givens vom obține zero-uri sub diagonala principală în matricea A, va apărea un avantaj computațional atunci când A este matrice rară sau în cazul paralelizărilor.

Dezavantaje: Unul din dezavantajele algoritmului provine tocmai din avantajul menționat anterior, și anume că trebuie să formăm zero-uri sub diagonala principală la fiecare iterație pentru fiecare element, va apărea un dezavantaj temporal în cazul matricelor dense față de ceilalți doi algoritmi, în mod special față de Householder, care are un caracter "all at once". De asemenea, așa cum am ilustrat în secțiunea dedicată analizei complexităților, numărul de pași necesari pentru obținerea factorizării în cazul matricelor nepătratice este aproape dublu față de numărul de pași necesari pentru Householder.

Householder

Avantaje: Este cel mai stabil numeric algoritm dintre cei prezentați, analiza sa perturbațională nedepinzând de condiția numerică a matricei A. Faptul că se utilizează reflectorii drept mecanism de producere a zero-urilor reprezintă un avantaj din punct de vedere al timpului computațional, formându-se zero-uri sub diagonala necesară fără a fi nevoie să luăm fiecare element în parte. Tot din acest motiv, algoritmul este utilizat aproape exclusiv pentru matricele dense, deoarece formarea de zero-uri se face mult mai eficient față de ceilalți doi algoritmi.

Dezavantaje: Un dezavantaj provine din faptul că nu este paralelizabil, din cauza faptului că fiecare element nou de 0 schimbă integritatea matricelor Q și R, acest motiv fiind și cauza pentru care, în cazul matricelor sparse, Householder nu este algoritmul recomandat. De asemenea, în cazul în care avem nevoie de forma ortogonală directă (așa cum este obținută prin Gram-Schmidt), în cazul aplicării Householder vom obține forma ortogonală într-o formă factorizată (produsul reflexiilor elementare). [13]

3 Evaluare

3.1 Descrierea modalității de construire a setului de teste folosite pentru validare

Testele au fost concepute astfel încât să reliefeze atât performanțele legate de corectitudine și de timpul de execuție ale algoritmilor, cât și cazurile speciale în care algoritmii implementați nu generează soluții corecte sau nu funcționează deloc.

În generarea testelor, am avut în vedere cuprinderea atât a cazurilor de matrice dense, cât și de cele rare, având în vedere faptul că acestea joacă un rol primordial în diferențierea eficienței algoritmilor. Astfel, am conceput următoarele seturi de teste:

1. Set de teste sistem pătratic:
   - are ca scop testarea algoritmilor pe matrice pătratice atât dense cât și rare, cu dimensiuni m = n cuprinse între $5 < n < 200$; am considerat necesară extinderea testelor până la aproape 200 de elemente pentru a putea evidenția variația eficienței operațiilor algoritmilor în funcție de dimensiunea matricei date.
2. Set de teste sisteme supradimensionate, subdimensionate:
   - are ca scop observarea comportamentului algoritmilor implementați pentru cele două tipuri de sisteme; dimensiunile matricelor ecuațiilor au valori între $5 < n < 50$, scopul fiind strict observarea funcționalității pentru aceste cazuri particulare de sisteme de ecuații.

3. Set de teste sisteme nedeterminate, incompatibile, sisteme cu matricea A densă și cu elementele diagonalei principale egale cu 0:
   – este un set de teste experimental pentru a vedea cazurile speciale de funcționare ale algoritmilor; dimensiunile matricelor ecuațiilor sunt cuprinse între $3 < n < 30$, scopul fiind acela de a găsi cazurile în care algoritmii nu funcționează sau funcționează parțial.

Observații:

– Pentru generarea testelor, s-au folosit două programe implementate în C pentru generarea de numere aleatoare, în funcție de caz (pentru matrice dense sau rare).
– Validarea soluțiilor a fost făcută comparând vectorul coloană rezultat x cu soluția generată de comanda linsolve din Matlab pentru rezolvarea sistemelor de ecuații liniare.
– Pentru algoritmul Gram-Schmidt, pentru cazul matricelor pătratice, am utilizat algoritmul Modified Gram-Shmidt, pentru a obține mai puține erori de calcul (în special în cazul matricelor sparse).

3.2 Specificațiile sistemului de calcul

Hardware:

– CPU: AMD Ryzen 7 4800HS with Radeon Graphics 2.90 GHz (8M Cache, up to 4.20 GHz)
– RAM : 16GB DDR4 3200 MHz
– Stocare: SSD 512GB M.2 PCIe
– Memorie disponibilă: 142GB

Software:

– Sistem de operare principal: Windows 10 Pro
– Sistem de operare folosit pentru rularea testelor: Ubuntu 20.04.3 LTS, rulat folosind VMware Workstation Pro, 4GB memorie RAM, 2 processor cores, 50GB hard disk, memorie disponibilă 22GB

Rularea testelor:

– Pentru fiecare algoritm, testele au fost rulate de 5 ori pentru perioada de warm-up și de 5 ori pentru perioada de testare a timpilor
– Timpii afișați în reprezentările grafice ulterioare reprezintă media aritmetică a timpilor obținuți din perioada de testare

3.3 Ilustrarea rezultatelor evaluării soluțiilor pe setul de teste

Notă: În reprezentările tabelare:

- QR reprezintă timpii de execuție pentru obținerea matricelor Q și R, BS reprezintă timpii de execuție pentru backward-substitution, iar Total reprezintă timpul total de execuție până la obținerea soluției
- cu bold sunt accentuați timpii cei mai mici, cu roșu timpii de execuție în cazul în care se generează o soluție diferită față de cea așteptată, iar în cazul în care nu se generează o soluție, timpii au fost marcați cu '-'.

Rezultatele rulării testelor pe matrice pătratice dense										
Nr. Test	Dimensiune matrice	Timp execuție (ms)								
		Householder			Givens			Gram-Schmidt		
		QR	BS	Total	QR	BS	Total	QR	Total	BS
1	10x10	0.58293	0.68903	**1.272**	1.3499	0.47803	1.828	0.93102	0.53406	1.4651
2	9x9	0.53406	0.45896	**0.99301**	1.0991	0.85402	1.9531	0.80085	1.888	2.6889
5	16x16	1.4629	0.68712	**2.1501**	3.7751	0.58603	4.3612	2.1861	0.5331	2.7192
6	28x28	1.8289	0.76008	**2.589**	16.953	2.043	18.996	7.5412	0.73409	8.2753
7	37x37	2.193	1.0009	**3.1939**	28.432	0.85783	29.29	11.06	0.89407	11.954
8	48x48	3.629	0.96011	**4.5891**	68.101	1.0309	69.132	17.969	1.055	19.024
13	58x58	5.5821	1.0591	**6.6411**	137.76	1.6949	139.46	34.04	1.153	35.193
14	67x67	15.288	2.1682	**17.456**	256.63	1.3161	257.95	41.195	1.595	42.79
15	79x79	19.886	1.3421	**21.228**	463.74	1.5941	465.33	51.62	4.0929	55.713
16	93x93	26.074	1.6131	**27.687**	848.74	1.6088	850.35	67.659	1.6909	69.35
21	107x107	40.076	2.0211	**42.097**	1622	2.073	1624	93.746	6.084	99.83
22	125x125	58.625	2.1582	**60.783**	3126.8	2.0921	3128.9	126.49	6.691	133.18
23	135x135	80.233	2.5122	**82.745**	4129.3	2.5179	4131.8	146.6	6.0639	152.66
24	146x146	102.96	2.4209	**105.38**	5892.4	3.0921	5895.5	176.52	2.934	179.45
29	167x167	131.57	2.5539	**134.12**	10438	2.8491	10440	231.75	6.988	238.73
30	178x178	207.43	3.104	**210.53**	16590	2.8539	16592	380.19	3.495	383.69
31	191x191	225.02	2.9788	**228**	19991	3.202	19994	313.97	4.5071	318.47

Table 1. Reprezentarea tabelară a timpilor de execuție ai algoritmilor pentru matrice dense

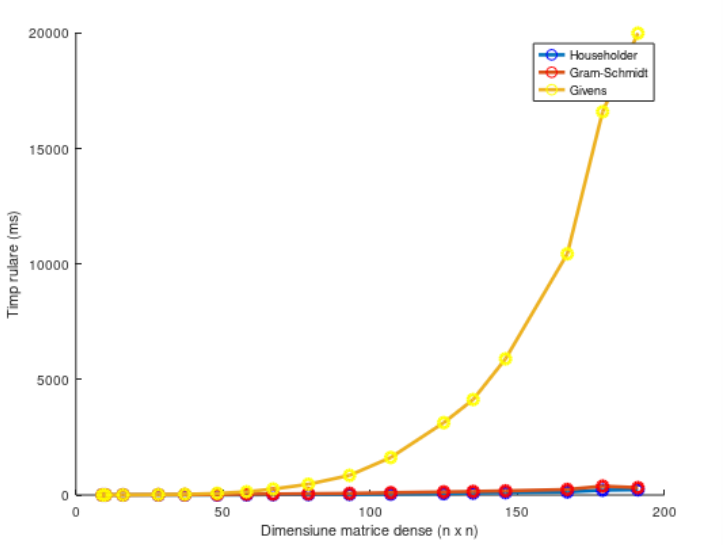

Fig. 1. Reprezentarea grafică a timpilor algoritmilor în funcție de dimensiunea matricei

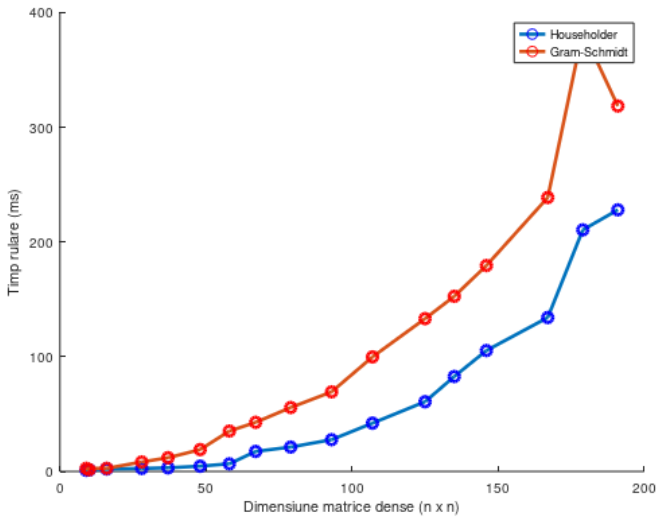

Fig. 2. Reprezentarea grafică a timpilor algoritmilor Householder și Gram-Schmidt în funcție de dimensiunea matricei dense

Rezultatele rulării testelor pe matrice pătratice rare										
Nr. Test	**Dimen-siune matrice**	**Timp execuție (mse)**								
		Householder			**Givens**			**Gram-Schmidt**		
		QR	**BS**	**Total**	**QR**	**BS**	**Total**	**QR**	**BS**	**Total**
3	7x7	0.71192	0.43797	1.1499	0.76008	0.53096	1.291	0.561	0.40293	**0.96393**
4	9x9	0.54908	0.47994	**1.029**	1.8332	0.4909	2.3241	0.76818	0.44608	1.2143
9	16x16	0.81801	0.55504	**1.3731**	3.83	0.61083	4.4408	2.2659	0.61202	2.878
10	28x28	1.3981	0.68593	**2.084**	13.241	0.90289	14.144	7.077	1.009	8.086
11	36x36	2.0151	0.81015	**2.8253**	29.866	0.85998	30.726	10.498	0.81992	11.318
12	48x48	3.7451	0.96488	**4.71**	67.858	0.98395	68.842	18.301	1.2281	19.529
17	57x57	5.5051	1.1339	**6.639**	128.69	1.132	129.82	28.043	1.3719	29.415
18	68x68	15.115	1.4861	**16.601**	261.28	1.3289	262.61	36.549	1.41	37.959
19	81x81	21.226	1.534	**22.76**	459.23	1.646	460.87	54.869	1.456	56.325
20	96x96	28.968	-	-	899.65	1.8041	901.46	71.811	6.088	77.899
25	106x160	36.618	1.7021	**38.32**	1544	1.997	1546	97.27	6.0511	103.32
26	121x121	58.959	2.033	**60.992**	2759.3	2.301	2761.6	115.03	6.506	121.53
27	132x132	54.75	2.2581	**57.008**	3588.5	2.12	3590.6	142.13	2.255	144.39
28	148x148	81.685	2.5461	**84.231**	6113.2	2.7289	6116	182.05	2.8391	184.89
32	167x167	128.69	2.7552	**131.44**	10678	2.9161	10680	432.6	3.4599	436.06
33	183x183	195.97	2.749	**198.72**	18396	3.2799	18399	281.37	3.077	284.45
34	196x196	257.52	4.1149	**261.64**	22040	3.366	22044	326.05	8.8959	334.95

Table 2. Reprezentarea tabelară a timpilor de execuție ai algoritmilor pentru matrice rare

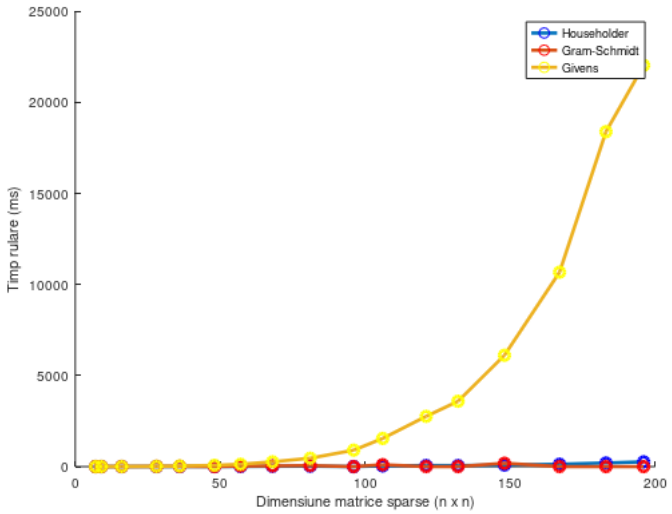

Fig. 3. Reprezentarea grafică a timpilor algoritmilor în funcție de dimensiunea matricei

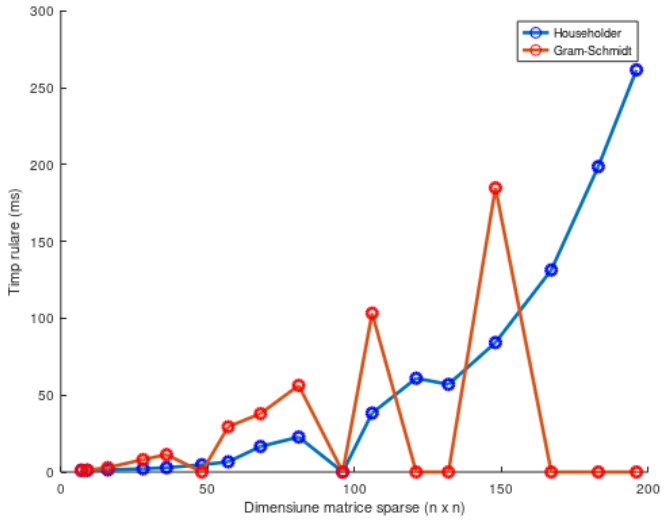

Fig. 4. Reprezentarea grafică a timpilor algoritmilor Householder și Gram-Schmidt în funcție de dimensiunea matricei sparse

Rezultatele rulării testelor pe sisteme de ecuații specifice											
Nr. Test	Tip sistem	Dimen-siune matrice	Timp execuție (ms)								
			Householder			Givens			Gram-Schmidt		
			QR	BS	Total	QR	BS	Total	QR	BS	Total
35	Supradim.	5x4	0.94795	-	-	0.36883	-	-	0.37885	0.61321	**0.99206**
36		14x13	0.87404	-	-	5.398	-	-	1.029	0.68498	**1.714**
37		36x34	-	-	-	26.895	-	-	6.124	-	-
38		48x47	1.8289	-	-	69.221	-	-	12.251	-	-
39	Subdim.	5x6	0.42605	0.74315	1.1692	0.40102	0.39196	0.79298	0.37789	-	-
40		14x15	0.96297	0.61607	1.579	2.8112	1.1549	3.9661	1.3392	-	-
41	Nedeterminat	5x5	0.39601	0.63992	1.0359	0.42701	0.6659	1.0929	0.33307	0.54097	**0.87404**
42		10x10	0.664	0.55099	**1.215**	1.4589	0.47994	1.9388	0.93389	0.80514	1.739
43		16x16	0.95105	0.66209	1.6131	3.834	0.57888	4.4129	2.2941	0.61488	2.9089
44	Incompatibil	3x3	0.32783	-	-	0.35691	-	-	-	-	-
45	0-uri pe diago-nala principală	12x12	0.69618	-	-	2.0499	0.71716	2.7671	1.2851	0.64492	**1.93**
46		16x16	0.88596	-	-	3.9618	0.61393	4.5757	2.3129	0.71001	**3.0229**
47		25x25	1.4369	-	-	10.267	0.75293	11.02	5.0752	0.95201	**6.0272**

Table 3. Reprezentarea tabelară a timpilor de execuție ai algoritmilor pentru cazul sistemelor specifice

3.4 Prezentarea succintă a valorilor obținute

Din reprezentările grafice anterioare, putem observa atât corectitudinea algoritmilor, cât și timpul acestora de execuție în funcție de dimensiunea și tipul matricei A a sistemului de ecuații.

Pentru matrice dense, conform Tabelului 1, observăm că toți algoritmii au ajuns la rezultatul așteptat, cea mai bună eficiență temporală având-o algoritmul Householder. Cea mai bună ilustrare a timpilor de execuție se observă însă din Figura 1, unde observăm creșterea aproape exponențială a timpilor pentru algoritmul Householder concomitentă cu creșterea dimensiunii matricei. Pentru matrice de dimensiuni mici, timpii de execuție ai algoritmilor sunt relativ similari (diferente de ordinul zecilor de milisecunde), cele mai mici diferențe fiind între Householder și Gram-Schmidt, așa cum se observă în Figura 2.

Pentru matrice sparse, constatăm apariția erorilor de calcul în cazul algoritmului Gram-Schmidt, așa cum reiese din Tabelul 2, în special pentru matrice de dimensiuni mari, eșuând 7 din 17 teste. Observăm de asemenea faptul că și algoritmul Householder eșuează testul 20, matricea R nefiind generată corect (din această cauză nu se poate aplica backward-substitution, neajungându-se la soluție). Conform Figurii 3 și Figurii 4, observăm că timpii sunt în mare parte similari cu cei din cazul matricei dense.

În ceea ce privește tipurile specifice de sisteme de ecuații, conform Tabelului 3 observăm că pentru matrice supradimensionată doar Gram-Schmidt funcționează parțial, ceilalți doi algoritmi deși generând matricele Q și R, nu ajung la vectorul soluție deoarece matricea R obținută nu este superior triunghiulară. Pentru matrice subdimensionate observăm opusul, însă deși Householder și Givens generează soluții, acestea nu sunt corecte, generându-se m soluții în loc de n (se generează mai puține soluții). Pentru sistemul compatibil nedeterminat, toți algoritmii funcționează parțial (toți eșuează testul 43 din cauza erorilor de aproximare). Pentru sistemul incompatibil, Householder și Givens generează matrice Q și R, însă apar erori de împărțire la 0 în calcularea soluției folosind backward-substitution; pentru Gram-Schmidt, nu se generează nici măcar Q și R - similar, apar erori de împărțire la 0. Surprinzător este rezultatul obținut pentru sistemele care au 0 pe diagonala principală, Householder eșuând toate testele deoarece matricea R obținută nu este superior triunghiulară.

4 Concluzii

Corelând rezultatele obținute în urma testelor generate, nu putem afirma că aplicarea unuia dintre cei trei algoritmi va duce cu siguranță la soluția corectă într-un timp relativ scurt indiferent de mărimea matricei de intrare. După cum am observat, pentru matrice de dimensiuni mici, indiferent de tipul acestora (dense sau rare), rularea algoritmilor a dus la rezultatul așteptat într-un timp relativ scurt și similar între cei trei. În practică, adevărata problemă se pune însă atunci când avem de a face cu matrice de dimensiuni medii-mari, în care atât corectitudinea, cât și timpul computațional joacă un rol cheie în determinarea algoritmului folosit.

Din acest motiv, algoritmul Gram-Schmidt, deși cel mai ușor de înțeles și aplicat, nu este potrivit pentru acest scop din cauza erorilor de calcul care pot apărea, așa cum am argumentat în capitolele anterioare.

Algoritmul Givens, deși a avut cele mai bune performanțe în ceea ce privește acuratețea soluției în fiecare dintre cele trei seturi de teste, prezintă un timp de execuție exponențial mai mare față de ceilalți doi algoritmi în cazul matricelor de dimensiuni mari, ceea ce poate deveni un dezavantaj în cazul în care avem nevoie de algoritmi cu timp de execuție mic, așa cum este de dorit de cele mai multe ori în industrie. Cu toate acestea, dacă ne dorim siguranța unei soluții corecte în cazul matricelor rare, iar factorul de timp nu reprezintă o prioritate, Givens reprezintă cea mai bună alternativă dintre cei trei algoritmi.

Făcând o comparație atât a performanței temporale a algoritmilor, cât și a corectitudinii acestora, ajungem la concluzia că algoritmul Householder reprezintă varianta de mijloc ce poate fi utilizată indiferent de dimensiunea sau tipul matricei, având cea mai bună eficiență a timpilor de execuție (după cum am ilustrat în secțiunea anterioară) și o acuratețe a soluției aproape întotdeauna corectă, existând însă șanse de eroare mai ales în cazul matricelor rare de dimensiuni mari.

Concluzionând, deși alegerea algoritmului depinde strict de tipul sistemului de ecuații, algoritmul de factorizare QR Householder reprezintă cea mai bună soluție atât din punctul de vedere al corectitudinii soluției, cât și a timpului de execuție, așa cum am ilustrat de-a lungul analizei celor trei algoritmi.

Referințe

1. https://images1.wikia.nocookie.net/nccmn/ro/images/c/cf/Sisteme_de_ecua%C5%A3ii_liniare.pdf
   Ultima accesare: 30 Oct 2021.

2. https://machinelearningmastery.com/examples-of-linear-algebra-in-machine-learning/
   Ultima accesare: 01 Nov 2021.

3. Curs 4 Metode Numerice, Florin Pop, 2021
   https://ctipub.sharepoint.com/:b:/s/03-ACS-L-A1-S2MetodenumericeSeriaCACBCCCD-2020/EfI2bH5I1-JNjCBr393SkpABWfGY8JiBtePaU9CmYnHKSw?e=OfRZPK
   Ultima accesare: 6 Dec 2021.

4. https://rpubs.com/aaronsc32/qr-decomposition-gram-schmidt
   Ultima accesare: 15 Dec 2021.

5. https://www.sciencedirect.com/topics/mathematics/gram-schmidt-process
   Ultima accesare: 17 Dec 2021.

6. https://docplayer.net/97619159-Metode-numerice-laborator-3-transformari-ortogonale-\householder-si-givens-algoritmul-gram-schmidt-polinoame-ortogonale.html
   Ultima accesare: 03 Dec 2021.

7. https://dokumen.tips/documents/laboratorul-3-actorizari-ortogonale-metode-\de-factorizare-1metoda-householder.html
   Ultima accesare: 03 Dec 2021.

8. https://laurenthoeltgen.name/post/gram-schmidt/
   Ultima accesare: 15 Dec 2021.

9. https://www.math.uci.edu/~ttrogdon/105A/html/Lecture23.html
   Ultima accesare: 15 Dec 2021.

10. https://www.scribd.com/document/328180479/qr
    Ultima accesare: 16 Dec 2021.

11. http://math.ubbcluj.ro/~tradu/sliderom/sistemeMD.pdf
    Ultima accesare: 17 Dec 2021.

12. http://www.ams.sunysb.edu/~jiao/teaching/ams526_fall12/lectures/lecture13.pdf
    Ultima accesare: 17 Dec 2021.

13. https://math.stackexchange.com/questions/781872/householder-vs-gram-schmidt-orthogonalization-which-should-i-use
    Ultima accesare: 17 Dec 2021.