Proiect Analiza Algoritmilor
Rezolvarea Sistemelor de Ecuații Liniare
Etapa Finală

Mîrza Ana-Maria, 321CA

Facultatea de Automatică și Calculatoare

Abstract. Analiza performanței unor algoritmi cunoscuți pentru rezolvarea sistemelor de ecuații liniare în funcție de tipul sistemului.

Keywords: GPPS · Householder · Gauss-Seidel.

1 Introducere

1.1 Descrierea problemei rezolvate

Rezolvarea sistememlor de ecuații este o problemă bine cunoscută în Computer Science și cu aplicabilități directe în multe aplicații software. Necesitatea de rezolvare a sistemelor de ecuații cât mai optim este datorată dorinței de a putea rezolva eficient astfel de sisteme cu software specializat, pentru a ne ușura viața. Forma generala pentru astfel de ecuații cu n necunoscute arată în modul următor:

$$\begin{cases} \mathrm{a}_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... + a_{2n}x_n = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + ... + a_{3n}x_n = b_3 \\ ..... \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + ... + a_{mn}x_n = b_m \end{cases}$$

Iar forma matriceală este $A \cdot x = b$, unde $\mathrm{A} \in R^{m \times n}$ este matricea coeficienților, $x \in R^{n \times 1}$ este vectorul necunoscutelor, iar $b \in R^{m \times 1}$ este vectorul termenilor liberi. Pentru rezolvarea unui astfel de sistem există două tipuri de metode: metode directe (sau exacte) și metode indirecte (sau iterative).

Metode directe. Acestea constau în efectuarea unor operații asupra matricei coeficienților, sau factorizarea acesteia, pentru a o aduce intr-o formă superior sau inferior triunghiulară. Odată ajuns triunghiular, sistemul se poate rezolva relativ ușor prin substituție. Câțiva dintre cei mai cunoscuți algoritmi ce folosesc metoda directă sunt *metoda Gauss*, *metoda Gauss-Jordan*, *metoda Cholesky* și *metoda Householder*. Limitările aduse de aceste metode sunt faptul că matricea coeficienților trebuie să fie patratică și nesingulară, iar complexitatea algoritmilor

este de $O(n^3)$, ceea ce devine o problemă atunci avem matrici foarte mari. La astfel de matrici de ordin mare (ordinul sutelor sau miilor), aceste metode de rezolvare devin inutilizabile și datorită erorilor de calcul ce se acumulează prin rotunjirile efectuate.

Metode indirecte. Dacă metodele directe furnizau o soluție exactă, metodele indirecte construiesc un șir de vectori n-dimensionali ce converg la soluția sistemului, urmând ca soluția să fie aleasă în urma unor restricții de precizie. În contrast cu metodele exacte, cele iterative durează mai mult ca timp pentru soluționarea matricelor de dimensiuni mici cu aceași acuratețe, dar sunt mai eficiente în cazul matricelor Sparse, unde complexitatea operațiilor este este de $O(n^2)$ per iterație. Câteva dintre cele mai cunoscute includ *Jacobi*, *Gauss-Seidel*, și *metodele de relaxare*. Metoda iterativă a *gradientului conjugat* ce poate fii oprită dupa exact n pași, poate ajunge până la complexitate de $O(m\sqrt{k})$. [3]

1.2 Aplicații

Aplicatie 1 Metodele de rezolvare a sistemelor de ecuații liniare sunt folosite inclusiv în Matlab, un limbaj de programare cât și un mediu de dezvoltare. Acesta este folosit în industrie și în universități pentru calcul numeric și analiză statistică. Un exemplu concret este operatorul *linsolve* ($x = LINSOLVE(A, b)$) ce rezolvă un sistem liniar $A \cdot x = b$. Soluția este calculată de Matlab folosind factorizare LU cu pivotare parțială atunci când matricea A este pătratică, și factorizare QR cu pivotare pe coloane. [5]

Aplicatie 2 Încă o aplicație practică pentru sistemele de ecuații liniare sunt în soft-uri de simulare a circuitelor electrice ca **LTSpice** ce permite utilizarea surselor de tensiune/curent, bobine, condensatoare, diode, și multe altele. Un circuit de acest fel, poate fii rezolvat cu metoda potențialelor nodurilor, reprezentându-l ca un graf orientat, având tensiunile pe coarde și nodurile din circuit ca noduri ale grafului, sau prin metoda Kirchhoff pe tensiuni sau curenți. Acestă aplicație folosește metodele prezentate, atât exacte cât și iterative, pentru a rezolva cele $n - 1$ ecuații rezultate.

1.3 Specificarea Soluțiilor Alese

Pentru analiză, am ales trei algoritmi cunoscuți ce rezolvă un sistem de ecuații liniar. Metodele alese sunt Gauss cu pivotare parțială cu pivot scalat, Householder și Gauss-Seidel. Aceste metode sunt de diferite tipuri (exacte și iterative) pentru a putea observa diferențele diferitelor metode.

1.4 Evaluarea Soluțiilor

Ne dorim să observăm eficiența metodelor menționate testând viteza de execuție a acestora în funcție de mărimea matricelor, pe un număr de teste generate

aleator cu ajutorul Octave/Matlab-ului și a unor teste făcute manual.

Testele vor conține:
1. Matrice de dimensiuni normale, compatibile și pătratice
2. Matrice mari, compatibile, convergente
3. Matrice dense / rare

Cazuri particulare:
1. Matrice compatibile nedeterminate
2. Matrice incompatibile
3. Matrice subdimensionate
4. Matrice supradimensionate

Voi testa pe metodele exacte toate testele, iar metoda Gauss-Seidel o voi testa doar cu acele teste ce conțin matrice convergente. Pentru determinarea acurateței, voi determina eroarea medie a metodelor exacte folosind formula $\in (x) = \|b - A \cdot x\|$, ce compară rezultatul obținut cu valoarea de adevăr efectivă, pentru a seta eroarea acceptată la metoda Gauss-Seidel.

2 Prezentatarea Soluțiilor

2.1 Descrierea Algoritmilor

GPPS. Eliminarea Gaussiană este cea mai cunoscută metodă de rezolvare exactă a sistemelor de ecuații datorită simplității, dar nu și cea mai eficientă. Această metodă se bazează pe operații de interschimbare, multiplicare și adunare a liniilor matricei $A \in R^{m \times n}$ a coeficienților, în urma cărora matricea A ajunge superior triunghiulară. Pentru minimizarea erorilor de rotunjire a calculelor, alegem eliminarea gaussiană cu pivotare parțială și scalare: se alege linia de sub pivot cu cel mai mare raport (în modul) dintre elementul de sub pivot și cel mai mare element de pe linia respectivă. Odată ales pivotul, scădem din liniile de sub pivot linia pivotului, dacă e nevoie cu scalare, astfel încât toate elementele de pe coloana pivotului să fie 0 sub pivot. Se trece la următorul pivot de pe diagonala principală și se aplică aceiași pași până se ajunge la un sistem superior triunghiular ce poate fii rezolvat prin substituție înapoi.

Householder. Householder este o metodă de facgorizare QR ce se pretează relativ bine pe sisteme de dimensiuni mari și cu multe elemente nule (matrice Sparse). Metoda folosește transformări ortogonale pentru fiecare coloană a matricei utilizând reflectorul elementar Householder,

$$H = I_n - 2\frac{vv^T}{v^T v}, \tag{1}$$

unde H este matrice ortogonală și simetrică, cu proprietatea $H^2 = I_n$. După aplicarea factorizării asupra fiecărei coloane din matricea A a coeficienților, se

obțin matricile Q, ortogonală, și R, superior triunghiulară ($A \rightarrow QR$). Astfel, sistemul inițial

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

devine

$$RQ \cdot x = b \tag{3}$$

Cum o matricea Q ortogonală are proprietatea $Q^{-1} = Q^T$, rezultă următoarea relație

$$R \cdot x = Q^T \cdot b \tag{4}$$

Sistem ce îl putem rezolva prin substituție.

Gauss-Seidel. Pentru diversitate, am ales și metoda Gauss-Seidel pentru a contrasta cu o metodă iterativă, deși aceasta poate fii folosită doar pentru matrice ce converg (matricea de iterație **G** este subunitară și valorile de pe diagonala principală sunt nenule). Această metodă rezolvă sistemul liniar $A \cdot \mathbf{x} = b$ începând de la o soluție inițială $\mathbf{x}^{(0)}$ și generează un vector $\{\mathbf{x}^{(k)}{}_{k=0}^{\infty}\}$ ce converge la soluția reală **x**. Algoritmul se oprește după un număr maxim de pași specificați, sau când atinge acuratețea dorită. Formula generală folosită este

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=i}^{i-1} a_{i,j} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(k)}}{a_{i,i}}, \tag{5}$$

care este o îmbunătățire a formulei inițiale folosită de metoda Jacobi

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=i}^{n} a_{i,j} x_j^{(k)}}{a_{i,i}} \tag{6}$$

2.2 Analiza Complexității Soluțiilor

GPPS. Cunoaștem că pentru eliminarea gaussiană standard se fac aproximativ $\frac{n(n+1)}{2}$ operații de împărțire, $2 * \frac{2n^3+3n^2+5n}{6}$ de înmulțire și scalare. De unde ajungem la o complexitate de $\theta(n^3)$. Similar, clasa de complexitate a algoritmului de eliminare gaussian cu pivotare parțială și scalare îl calculăm determinând complexitatea găsirii pivotului și îl adunăm cu costul algorimului de eliminare gaussiană efectivă prin scăderi și scalări. Pentru găsirea pivotului avem $(n - k + 1)^2$ pași și încă $2 * (n - k + 1)$ pași pentru scalare și scăderi, la pasul k. Ajungem la următoarea ecuație: $\sum_{k=1}^{n}(n - k + 1)^2 + 2 * (n - k + 1)$, unde n este dimensiunea matricei și k un pas oarecare. Eliminând termenii neglijabili, rămânem cu $\sum_{k=1}^{n} k^2 + k$, ce se desface în $\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}$. Așadar, avem tot o complexitate operațională de $\theta(n^3)$.

Householder. Householder este o metoda de tip QR, ce se folosește de transformări liniare pentru a transforma o matrice într-una ortogonală, numită reflector Householder, și una superior triunghiulară. Pentru transformări se folosesc $n-1$ pași, iar la fiecare pas avem de calculat reflectorul Householder, $H = I_n - \frac{2u \cdot u^t}{u^t u}$, unde $u = -sign(a_{i,j}\sqrt{\sum_{j=i}(a_{i,j})^2})$ este vectorul unitate, și noua matrice A pentru următoarea iterație. Cum $A \leftarrow HA$, iar inmulțirile de matrice pătratice se fac în complexitate de $\theta(n^3)$, concludem că și metoda Householder se incadrează tot in aceași complexitate de $\theta(n^3)$, așa cum ne așteptam, fiind tot o metodă exactă.

Gauss-Seidel. Această metodă se incadrează la metodele de rezolvare iterativă, rezultatul fiind unul aproximativ, cu o toleranță dată. Deși soluția oferită nu este una exactă, vom demonstra că algoritmul reușește să reducă costul la $\theta(n^2)$, ceea ce îl face extrem de util pentru sisteme mari/sparse. Formula generală de calcul

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=i}^{i-1} a_{i,j} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(k)}}{a_{i,i}}, \tag{7}$$

face $n-1$ pași de adunare pentru toate cele n ecuații din sistem. De unde rezultă că avem n^2 numărul total de operații, care este în $\theta(n^2)$.

2.3 Avantaje și Dezavantaje

GPPS. Deși cea mai ușor de folosit metodă de rezolvare a sistemelor de ecuații, eliminarea gaussiană (cu pivotare parțială și scalare) oferă următoarele:

Avantaje:
- ușor de înțeles
- furnizează o soluție exactă
- stabilitate numerică
- eficiență sporită pe sisteme de dimensiuni mici/normale

Dezavantaje:
- cost computatțional suplimentar
- nu este eficient pentru sisteme de ecuații mari datorită operațiilor multe
- fezabil doar pentru matrice pătratice

Householder. Algoritmul de transformare ortogonală Householder oferă următoarele:

Avantaje:
- oferă o soluție exactă
- algorirm rapid pentru matrici normale
- util pentru algoritmi QR cu deflație
- elimină toate elementele de sub pivot într-un singur pas

Dezavantaje:
- complexitate mare
- nu este eficientă pentru sisteme mari/sparse
- folosește memorie suplimentară față de alte metode (ex. eliminare Gaussiană)

Gauss-Seidel. Metoda iterativă Gauss-Seidel oferă următoarele:

Avantaje:
- complexitate redusă de calcul
- fezabilă pentru matrice mari/sparse
- util pentru algoritmul SOR (Successive Overrelaxation)
- util pentru rezolvarea sistemelor neliniare [7]

Dezavantaje:
- oferă soluții aproximative, nu exacte
- nu poate fii folosit decât pe matrice convergente
- nu este eficientă pentru matrice de dimensiuni normale

3 Evaluare

3.1 Descrierea modalității de construire a setului de teste

Pentru generarea testelor, am folosit un script în Matlab $R2022b$ ce a generat întâi un număr de 20 de teste cu matrice pătratice de dimensiuni normale, ce creștea la fiecare test, apoi încă 20 de teste cu matrice pătratice convergente, făcând matricele diagonal dominante. Pentru fiecare test generat, s-a generat o matrice A pătratică și un vector b. Deoarece testele inițiale au fost generate cu o limită de 100 penru dimensiunea matricelor, am mai generat la sfârșit un număr de 10 teste cu matrice de dimensiuni mari (de ordinul sutelor), în folderul "$other\_tests/$", pentru a observa mai bine diferențele dintre algoritmi și comportamentul acestora. Aceste matrici generate au fost salvate în folderul "$in/$", iar soluția pentru ele a fost calculată folosind $A \backslash b$ și salvată în folderul "$out/$" de un alt script ce citea matricea A și b din fișierul de input. Deși acestea au fost generate random, nu au fost cazuri de matrici singulare/incompatibile/compatibile nedeterminate, iar ținând cont că aceste matrici nu ar fii testat performanța algoritmilor, nu am generat alte teste speciale cu aceste cazuri.

Pentru a testa acuratețea am folosit formula $\in (x) = \|b - A \cdot x\|$, iar pentru timpul de execuție am folosit funcțiile tic, toc puse la dipoziție de matlab. Acestea au fost calculate cu un alt script și salvate în folderul "$results/$" după multiple rulări pentru acurateție.

3.2 Specificațiile sistemului de calcul

Procesor: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
RAM: 16 GB
Core-uri: 8

3.3 Ilustrarea, folosind grafice/tabele, a rezultatelor obținute pe teste

Nr. Test	Dimensiune	GPPS(ms)	Householder(ms)	Gauss-Seidel(ms)
1	20	1.298	2.134	-
2	19	1.275	1.991	-
3	24	1.342	2.250	-
4	23	1.268	2.001	-
5	20	1.224	1.902	-
6	26	1.515	2.159	-
7	24	1.374	2.106	-
8	26	1.419	2.045	-
9	32	1.559	2.285	-
10	30	1.469	2.295	-
11	31	1.551	2.393	-
12	35	1.633	2.642	-
13	34	1.698	2.602	-
14	36	1.756	4.284	-
15	37	1.843	4.19	-
16	45	1.66	3.905	-
17	43	1.718	4.110	-
18	44	1.962	4.120	-
19	54	2.043	4.627	-
20	55	2.066	4.978	-
21	53	1.954	4.591	5.763
22	61	2.018	5.587	6.021
23	54	1.968	4.546	5.581
24	61	1.982	5.363	5.753
25	61	2.077	5.334	5.852
26	61	2.145	5.591	6.035
27	63	2.326	5.998	6.236
28	74	2.393	7.048	6.632
29	73	2.217	7.175	7.314
30	70	2.102	7.031	7.323
31	74	2.300	7.324	7.153
32	75	2.213	7.699	7.487
33	75	2.226	7.607	7.612
34	77	2.231	8.135	7.972
35	85	2.341	9.073	8.253
36	80	2.309	8.910	7.993
37	85	2.381	9.425	8.413
38	87	2.553	10.853	8.781
39	91	2.739	11.000	9.686
40	88	2.576	10.087	9.011

Aceste valori reprezintă timpul de execuție mediu, în milisecunde, al algoritmilor pentru primele 40 de teste cu matrice de dimensiuni normale (maxim 100) fără timpul de IO. Reprezentarea grafică este următoarea:

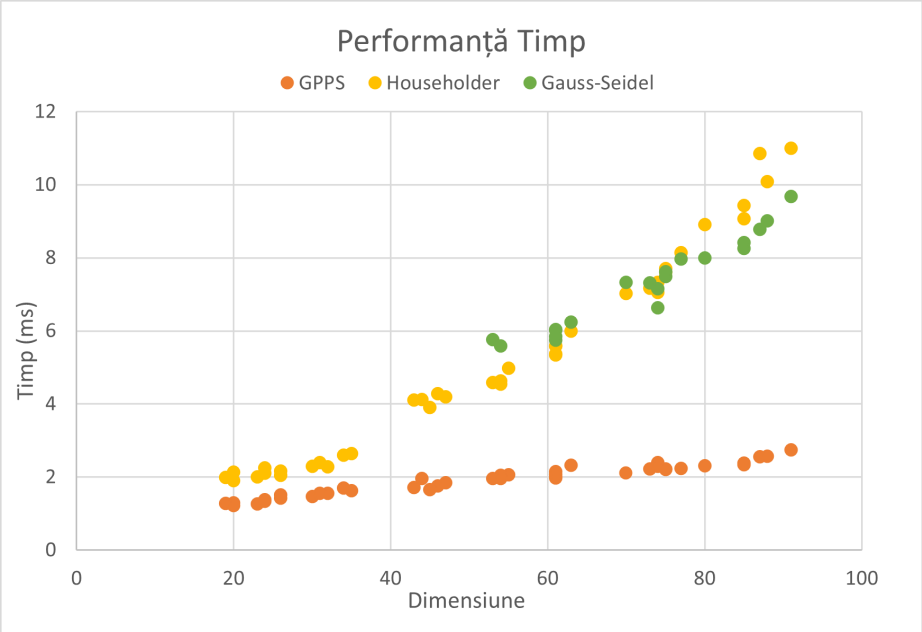

Fig. 1. Timp Execuție

Erorile soluțiilor calculate de fiecare metodă în parte sunt ilustrate în următorul tabel:

Nr. Test	Dimensiune	GPPS	Householder	Gauss-Seidel
1	20	1.64E-14	3.59E-14	-
2	19	2.38E-14	5.99E-14	-
3	24	2.87E-14	2.69E-14	-
4	23	4.44E-14	4.16E-14	-
5	20	7.53E-14	1.45E-13	-
6	26	1.94E-16	2.71E-13	-
7	24	3.63E-14	6.46E-14	-
8	26	7.63E-14	1.28E-13	-
9	32	3.69E-14	5.46E-14	-
10	30	7.20E-14	1.17E-13	-
11	31	2.15E-14	3.37E-14	-
12	35	3.47E-14	5.91E-14	-
13	34	6.50E-14	7.84E-14	-
14	36	1.29E-13	2.26E-13	-
15	37	1.16E-13	1.10E-13	-
16	45	5.29E-14	7.42E-14	-
17	43	6.09E-14	6.94E-14	-
18	44	7.32E-13	7.04E-13	-
19	54	6.62E-13	1.01E-12	-
20	55	1.57E-12	3.00E-12	-
21	53	1.44E-14	3.93E-14	1.23E-14
22	61	1.76E-14	7.72E-14	1.07E-14
23	54	1.61E-14	4.92E-14	1.05E-14
24	61	1.36E-14	6.05E-14	1.05E-14
25	61	1.75E-14	5.51E-14	1.28E-14
26	61	1.85E-14	6.46E-14	1.37E-14
27	63	1.64E-14	5.69E-14	1.17E-14
28	74	2.39E-14	6.47E-14	1.60E-14
29	73	2.15E-14	8.13E-14	1.36E-14
30	70	1.72E-14	5.99E-14	1.08E-14
31	74	1.91E-14	5.65E-14	1.14E-14
32	75	2.17E-14	7.32E-14	1.89E-14
33	75	2.02E-14	7.31E-14	1.64E-14
34	77	2.13E-14	6.50E-14	1.57E-14
35	85	2.52E-14	8.26E-14	1.39E-14
36	80	2.06E-14	6.79E-14	1.56E-14
37	85	2.31E-14	8.22E-14	1.63E-14
38	87	2.20E-14	8.56E-14	1.84E-14
39	91	1.95E-14	8.01E-14	1.78E-14
40	88	2.32E-14	8.80E-14	1.82E-14

Iar reprezentarea grafică arată în felul următor:

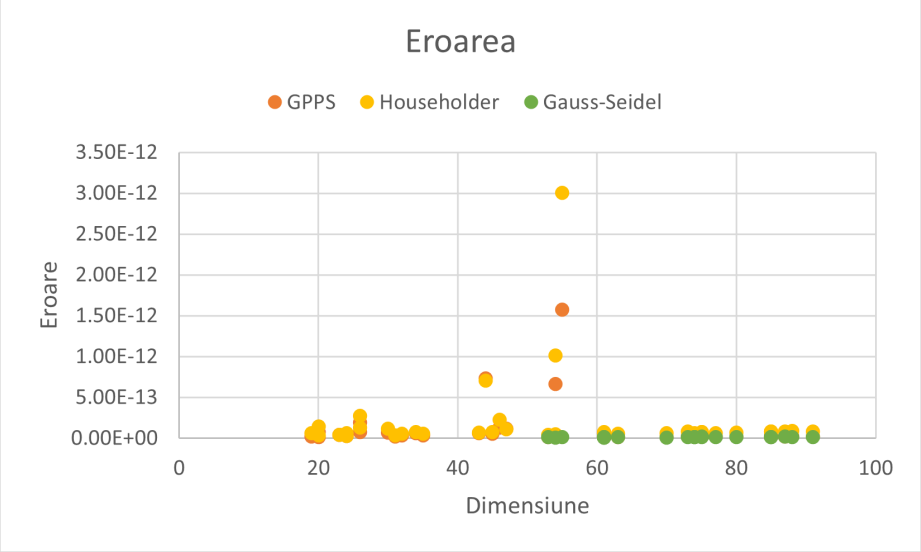

Fig. 2. Eroare de Calcul

Valorile obținute pentru timpii de execuție, în secunde, al matricelor mari (de ordinul sutelor) arată în felul următor:

Nr. Test	Dimensiune(s)	GPPS(s)	Householder(s)	Gauss-Seidel(s)
1	290	0.037175	0.258868	0.038427
2	318	0.046572	0.361161	0.047205
3	365	0.07434	0.886756	0.056615
4	406	0.100546	1.233952	0.068115
5	499	0.255585	2.619122	0.101147
6	464	0.235647	1.980961	0.091959
7	589	0.434549	5.130257	0.153587
8	618	0.50704	5.89449	0.17741
9	605	0.346395	5.610553	0.166818
10	681	0.516551	8.246866	0.214677

Graficul pentru aceste valori este următorul:

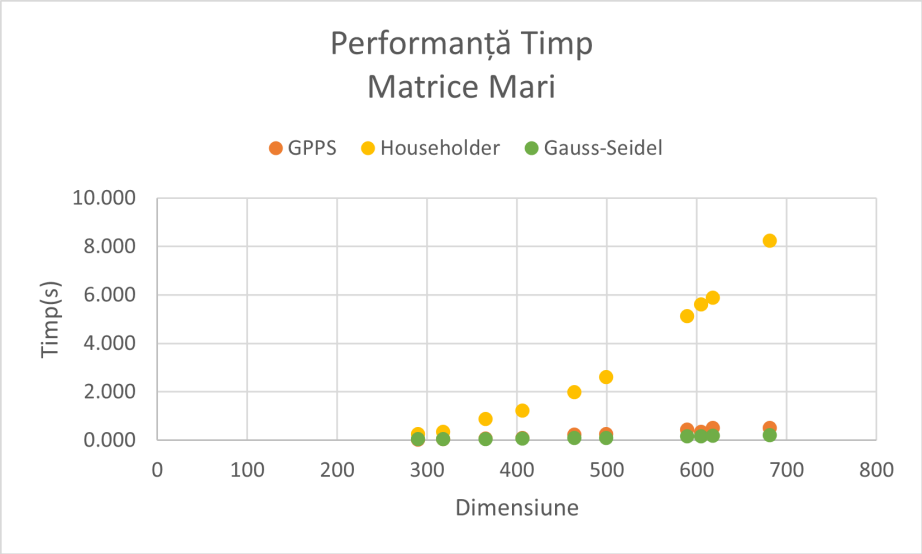

Fig. 3. Timp de Execuție Matrici Mari

Erorile afișate de aceste metode pentru testele cu matrice de dimensiuni mari sunt afișate în următorul tabel.

Nr. Test	Dimensiune	GPPS	Householder	Gauss-Seidel
1	290	6.66E-14	5.19E-13	4.97E-14
2	318	7.39E-14	6.36E-13	4.62E-14
3	365	6.88E-14	6.37E-13	5.67E-14
4	406	8.79E-14	7.43E-13	4.94E-14
5	499	1.13E-13	9.01E-13	6.62E-14
6	464	1.10E-13	8.73E-13	7.07E-14
7	589	1.32E-13	1.11E-12	8.97E-14
8	618	1.24E-13	1.20E-12	7.57E-14
9	605	1.19E-13	1.02E-12	7.54E-14
10	681	1.38E-13	1.35E-12	8.78E-14

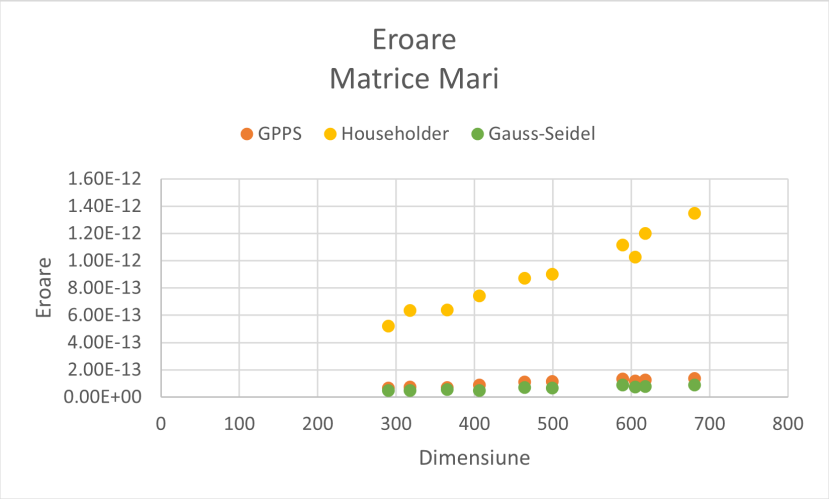

Fig. 4. Eroare de Calcul Matrici Mari

3.4 Interpretarea valorilor obținute pe teste

În urma testelor efectuate și a graficelor expuse mai sus, putem observa următoarele lucruri:

- GPPS este cel mai eficient algoritm pe matrici de dimensiuni mici

- Pe primele 20 de teste nu a fost testat metoda Gauss-Seidel deoarece aceste matrici nu converg, așa că metoda nu va putea furniza o soluție

- Householder și Gauss-Seidel au performanțe de timp asemănătoare pentru matrice de dimensiuni normale, probabil datorită matricelor dense ce necesită mai multe calcule

- Eroare obținută este aproximativ aceeași pentru GPPS și Householder; eroarea metodei Gauss-Seidel fiind aleasă de mine pentru a putea compara rezultatele obținute la execuția timpului, deci nu face parte din studiul de caz

- Testele executate pe matrice mari au arătat clar superioritatea metodei Gauss-Seidel: metoda are cea mai bună soluție atât ca timp cât și ca eroare

- Deși Householder este o metodă fezabilă pentru matrici mari/sparse, în cazul de față, algoritmul a fost cel mai ineficient din punct de vedere al timpului de rulare și al erorii. Probabil că dacă am fii testat pe matrici și mai mari, sau sparse, am fii putut vedea o îmbunătățire față de GPPS

4 Concluzii

În concluzie, fiecare algoritm are beneficiile lui și metoda aleasă pentru rezolvarea unui sistem de ecuații depinde de datele în cauză. Pentru matrice de dimensiuni mici/normale, cea mai bună metodă din cele prezentate ar fii GPPS. Această metodă rezolvă sistemul eficient din punct de vedere al timpului de execuție și cu o eroare realativ mică. Celelalte metode prezentate ar face risipă de resurse. Dacă avem de-aface cu matrice mai mari, vom lua în considerare folosirea unui algoritm mai eficient computațional, cum ar fii un algoritm iterativ. Putem folosi Householder dacă avem un sistem sparse, sau metoda iterativă Gauss-Seidel pentru o matrice mare. Pentru cea din urmă avem demonstrația testelor efectuate că vom avea o soluție foarte exactă, într-un timp foarte scurt, cât și demonstrația complexității reduse față de o metodă exactă. Pentru uz general, cel mai fezabil algoritm este GPPS, cele iterative fiind extrem de folositoare pentru cazuri de matrice mari/sparse.

References

1. Fadi N. Sibai: Performance modeling and analysis of parallel Gaussian elimination on multi-core computers. Journal of King Saud University – Computer and Information Sciences (2013)
2. Youyi Jiang:Convergence of The Gauss-Seidel Iterative Method. In: Procedia Engineering, Volume 15, pp. 1647–1650.School of Mathematics and Statistics, Chongqing Three Gorges University,Chongqing, 404100, P. R. China (2011) (2011)
3. Richard L. Burden, J. Douglas Faires: Numerical Analysis. 9th edn. ch. 6-7, Richard Stratton, Boston, MA USA (2011)
4. How ordinary elimination became Gaussian elimination, https://www.sciencedirect.com/science/article/pii/S0315086010000376. Ultima accesare 20 Noiembrie 2022
5. Solution of system of linear equation in MATLAB https://www.geeksforgeeks.org/solution-of-system-of-linear-equation-in-matlab/. Ultima accesare 20 Noiembrie 2022
6. Numerical linear algebra and matrix factorizations https://pi.math.cornell.edu/ web6140/TopTenAlgorithms/Householder.html. Ultima accesare 19 Decembrie 2022
7. José Manuel Gutiérrez, Ángel Alberto Magreñán, Juan Luis Varona.: The "Gauss-Seidelization" of iterative methods for solving nonlinear equations in the complex plane. Applied Mathematics and Computation 218(6):2467-2479 (2011)