**Universitatea Tehnică a Moldovei**

**Facultatea *Calculatoare, Informatică și Microelectronică***

**Specialitatea *Tehnologii Informaționale***



Raport

**la lucrarea de laborator nr. 2**

**Tema:*“******Rezolvarea numerica a sistemelor de ecuatii liniare”***

**Disciplina: “Metode numerice”**

Varianta 3

|  |  |  |
| --- | --- | --- |
| **A efectuat:** | Student grupa TI-231 FR | Apareci Aurica |
| **A verificat:** | Asistent universitar | Strună Vadim |

**Chișinău 2024**

**Cuprins**

[**1.** **Cadru teoretic** 3](#_Toc186125493)

[**2.** **Rezolvarea sistemului de ecuatii liniare Ax=b** 4](#_Toc186125494)

[**2.1 Metoda eliminarii lui Gauss** 4](#_Toc186125495)

[**2.2 Metoda lui Cholesky** 7](#_Toc186125496)

[**2.3 Metoda iterative a lui Jacobi** 9](#_Toc186125497)

[**2.4 Metoda Gauss-Seidel** 11](#_Toc186125498)

[**5.** **Concluzii** 15](#_Toc186125499)

[**6.** **Webobrafie** 16](#_Toc186125500)

# **Cadru teoretic**

**Tema lucrării:** Rezolvarea numerica a sistemelor de ecuatii liniare

**Scopul lucrării:**

1) Să se rezolve sistemul de ecuaţii lineare Ax=b, utilizând

- Metoda eliminării lui Gauss;

- Metoda lui Cholesky (metoda rădăcinii pătrate);

- Metoda iterativă a lui Jacobi cu o eroare ε=10-3;

- Metoda iterativă a lui Gauss-Seidel cu o eroare ε=10-3 şi ε=10-5.

2) Să se determine numărul de iteraţii necesare pentru aproximarea soluţiei sistemului cu eroarea dată ε. Să se compare rezultatele.

**Sarcina (V3):**

# **Rezolvarea sistemului de ecuatii liniare Ax=b**

**2.1 Metoda eliminarii lui Gauss**

Metoda eliminării a lui Gauss constă în a aduce sistemul iniţial la un sistem echivalent avînd matricea coeficienţilor superior triunghiulară. Transformarea sistemului sistem de formă triunghiulară fără ca să se modifice soluţia sistemului se realizează cu ajutorul următoarelor trei operaţii:

1) rearanjarea ecuaţiilor (schimbarea a două ecuaţii între ele);

2) înmulţirea unei ecuaţii cu o constantă (diferită de zero);

3) scăderea unei ecuaţii din alta şi înlocuirea celei de-a doua cu rezultatul scăderii

**Rezolvarea analitică**

*Divizăm al doilea rând la 10.4484*

|  |
| --- |
|  |
| Transformarea matricei într-o formă triunghiulară superioară   * 1. Pivotarea pe primul rând (eliminăm elementele de sub a11)   *Eliminăm a21, a31, a41 prin transformările:*  *R2 = R2 + 0.9\*R1  | R3 = R3 - 0.6\*R1 | R4 = R4 -0.8\*R1*  *Divizăm primul rând la 8.1*   * 1. Pivotarea pe al doilea rând (eliminăm elementele de sub a22)   *Divizăm al doilea rând la 14.4*  *Eliminăm a32, a42 prin transformările:*  *R3 = R3 - 0.3667\*R2 | R4 = R4 -0.7889\*R2*   1. Pivotarea pe al treilea rând (eliminăm elementele de sub a33)   *Divizăm al doilea rând la 7.8462*  *Eliminăm a43 prin transformările: R4 = R4 + 0.4794\*R3*   1. Pivotarea pe al patrulea rând (eliminăm elementele de sub a44)   => |

**Listingul programului**

|  |  |
| --- | --- |
| public class InpData | |
| static public int n = 4;  static public double[,] AB = { {0, 0, 0, 0, 0, 0},  {0, 8.1, -0.9, 0.6, 0.8, 7.2 },  {0, -0.9, 14.3, 0.3, 0.7, 10.3 },  {0, 0.6, 0.3, 7.9, -0.4, -11.9 },  {0, 0.8, 0.7, -0.4, 10.6, 9.2 }}; | |
| static public double[,] GetA() | **static public double[] GetB()** |
| {  double[,] A = new double[n, n];  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  A[i, j] = AB[i+1, j+1];  }  }  return A;  } | **{**  **double[] B = new double[n];**  **for (int i = 1; i <= n; i++)**  **{**  **B[i-1] = AB[i , n + 1];**  **}**  **return B;**  **}** |
| static public void PrintA() | **static public void PrintInputData()** |
| {  Console.WriteLine("Matricea A:");  for (int i = 0; i < n; i++)  {  for (int j = 0; j < n; j++)  {  Console.Write("{0,8:0.0000}",  AB[i + 1, j + 1]);  }  Console.WriteLine();  }  } | **{**  **PrintA();**  **Console.WriteLine("Vectorul B:");**  **for (int i = 1; i <= n; i++)**  **{**  **Console.WriteLine("{0,8:0.0000}",**  **AB[i, n + 1]);**  **}**  **}** |
| Gauss-Seidel functions | |
| static public double X1(double x2, double x3, double x4)  {  return (AB[1,n + 1] - AB[1,2] \* x2 - AB[1,3] \* x3 - AB[1,4] \* x4)/AB[1,1];  }  static public double X2(double x1, double x3, double x4)  {  return (AB[2,n + 1] - AB[2,1] \* x1 - AB[2,3] \* x3 - AB[2,4] \* x4)/AB[2,2];  }  static public double X3(double x1, double x2, double x4)  {  return (AB[3,n + 1] - AB[3,1] \* x1 - AB[3,2] \* x2 - AB[3,4] \* x4)/AB[3,3];  }  static public double X4(double x1, double x2, double x3)  {  return (AB[4,n + 1] - AB[4,1] \* x1 - AB[4,2] \* x2 - AB[4,3] \* x3)/AB[4,4];  } | |

*Gauss / Program.cs*

**using InputData;**

**namespace Gauss**

**{**

**internal class Program**

**{**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Metoda eliminarii lui Gauss pentru Ax = B");**

**InpData.PrintInputData();**

**Solution(Solve(EliminariGauss(InpData.AB, InpData.n), InpData.n));**

**}**

**static double[,] EliminariGauss(double[,]a, int n)**

**{**

**double ratio = 0;**

**for (int i = 1; i <= n - 1; i++)**

**{**

**if (a[i,i] == 0.0)**

**{**

**Console.WriteLine("Eroare matematica");**

**break;**

**}**

**for (int j = i + 1; j <= n; j++)**

**{**

**ratio = a[j,i] / a[i,i];**

**for (int k = 1; k <= n + 1; k++)**

**{**

**a[j,k] = a[j,k] - ratio \* a[i,k];**

**}**

**}**

**}**

**return a;**

**}**

**//Obtinerea solutiei prin metoda de inlocuire inversa**

**static double[] Solve(double[,] a, int n)**

**{**

**double [] x = new double[n + 1];**

**x[n] = a[n,n + 1] / a[n,n];**

**for (int i = n - 1; i >= 1; i--)**

**{**

**x[i] = a[i,n + 1];**

**for (int j = i + 1; j <= n; j++)**

**{**

**x[i] = x[i] - a[i,j] \* x[j];**

**}**

**x[i] = x[i] / a[i,i];**

**}**

**return x;**

**}**

**//Afisarea solutiei**

**static void Solution(double[] x)**

**{**

**Console.WriteLine("Solutia este:");**

**for (int i = 1; i < x.GetLength(0); i++)**

**{**

**Console.WriteLine($"x{i} = {x[i]:F4}");**

**}**

**}**

**}**

**}**

**A screenshot of a computer

Description automatically generatedRezultatele testării**

**2.2 Metoda lui Cholesky**

Metoda Cholesky presupune factorizarea matricei simetrice și pozitiv definite AAA în produsul a două matrice: *A=L⋅LT*, unde L este o matrice triunghiulară inferioară, iar *LT*, este transpusa sa. Apoi rezolvăm sistemul A⋅X=B prin rezolvarea a două sisteme succesive:

*L⋅Y=B* (prin substituție înainte);

*LT⋅X=Y* (prin substituție înapoi).

**Rezolvarea analitică**

|  |
| --- |
| Verificarea simetriei și pozitivitatea matricei   * 1. **A math problem with numbers       Description automatically generated**Matricea A este simetrică. Pentru verificarea pozitivității matricei, calculăm determinanții principalilor minori (M1, M2, M3, M4). Dacă toți sunt pozitivi, matricea este pozitiv definită.   Factorizarea Cholesky    *Factorizarea Colesky determinată cu ajutorul* [*Calculator de descompunere Cholesky*](https://www.omnicalculator.com/math/cholesky-decomposition)  Rezolvarea sistemelor succesive: |

**Listingul programului**

**using InputData;**

**namespace Cholesky**

**{**

**internal class Program**

**{**

**static public double[,] L;**

**static public double[,] Lt;**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Metoda lui Cholecky pentru Ax = B");**

**InpData.PrintA();**

**CholeskyDecomposition(InpData.GetA(), InpData.n);**

**Console.WriteLine("Verificarea rezultatelor:\nProdusul matricelor L\*Lt:");**

**MatrixProduct(L, Lt);**

**}**

*Cholesky / Program.cs*

**static void CholeskyDecomposition(double[,] matrix, int n)**

**{**

**double[,] lower = new double[n, n];**

**for (int i = 0; i < n; i++)**

**{**

**for (int j = 0; j <= i; j++)**

**{**

**double sum = 0;**

**if (j == i){**

**for (int k = 0; k < j; k++)**

**sum += Math.Pow(lower[j, k] , 2);**

**lower[j, j] = Math.Sqrt(matrix[j, j] - sum);**

**}else{**

**for (int k = 0; k < j; k++)**

**sum += (lower[i, k] \* lower[j, k]);**

**lower[i, j] = (matrix[i, j] - sum) / lower[j, j];**

**}**

**}**

**}**

**L = new double[n, n];**

**Lt = new double[n, n];**

**Console.WriteLine("Matricea L:");**

**for (int i = 0; i < n; i++)**

**{**

**for (int j = 0; j < n; j++)**

**{**

**Console.Write("{0,8:0.0000}", lower[i, j]);**

**L[i, j] = lower[i, j];**

**}**

**Console.WriteLine();**

**}**

**Console.WriteLine("Matricea L transpusa:");**

**for (int i = 0; i < n; i++){**

**for (int j = 0; j < n; j++){**

**Console.Write("{0,8:0.0000}", lower[j, i]);**

**Lt[i, j] = lower[j, i];**

**}**

**Console.WriteLine();**

**}**

**}**

**static void MatrixProduct(double[,]A, double[,]B)**

**{**

**double[,] C = new double[A.GetLength(0), B.GetLength(1)];**

**for (int i = 0; i < A.GetLength(0); i++)**

**{**

**for (int j = 0; j < B.GetLength(1); j++)**

**{**

**for (int k = 0; k < A.GetLength(1); k++)**

**{**

**C[i, j] += A[i, k] \* B[k, j];**

**}**

**}**

**}**

**for (int i = 0; i < A.GetLength(0); i++)**

**{**

**for (int j = 0; j < B.GetLength(1); j++)**

**{**

**Console.Write("{0,8:0.0000}", C[i, j]);**

**}**

**Console.WriteLine();**

**}**

**}**

**}**

**}**

**A screenshot of a computer

Description automatically generatedRezultatele testării**

**2.3 Metoda iterative a lui Jacobi**

Metoda iterativă a lui Jacobi este utilizată pentru rezolvarea sistemelor de ecuații liniare de forma A⋅X=BA \cdot X = BA⋅X=B. Aceasta este o metodă numerică care presupune descompunerea matricei AAA și obținerea unei soluții aproximative prin repetarea unui set de ecuații până când soluția converge la o valoare suficient de exactă.

***A math equations with numbers

Description automatically generated with medium confidencePașii metodei Jacobi:***

* Se rescrie fiecare ecuație din sistem astfel încât necunoscuta să fie izolată
* Pornind de la aproximarea inițială X(0), se calculează succesiv valori pentru fiecare variabilă.
* Procesul se repetă până când norma diferenței dintre două iterații succesive este mai mică decât un prag dat (e).

**Rezolvarea analitică**

A number lines with numbers

Description automatically generated with medium confidenceRescrierea sistemului pentru metoda Jacobi. Se izolează fiecare necunoscută și selectăm valorile inițiale de la care vom începe calculul iterativ.

A math equations on a white background

Description automatically generated



****

*Valoarea variabilelor după prima iterație.*

*Izolarea necunoscutelor. Inițializarea variabilelor inițiale.*

 Se continua procesul de iterare până diferența dintre două iterații consecutive devine mai mică decât (e). În cazul dat după 8 iterații, soluția converge la:

**Listingul programului**

**using InputData;**

**namespace IterativeJacobi**

**{**

**internal class Program**

**{**

**static void Main(string[] args){**

**Console.WriteLine("Metoda iterativa a lui Jacobi pentru Ax = B");**

**InpData.PrintInputData();**

**Console.WriteLine("\t\tX1\t\tX2\t\tX3\t\tX4");**

**Jacobi(InpData.GetA(), InpData.GetB(), new double[InpData.n], 0.00001);**

**}**

**static void Jacobi(double[,] A, double[] b, double[] x, double epsilon){**

**int size = A.GetLength(0);**

**double[] D = new double[size];**

**double[,] R = new double[size, size];**

**for (int i = 0; i < size; i++)**

**{**

**D[i] = A[i, i];**

**for (int j = 0; j < size; j++)**

**{**

**if (i != j){**

**R[i, j] = A[i, j];**

**}**

**}**

**}**

**bool converged = false;**

**int iterations = 0;**

**while (!converged)**

**{**

**double[] xNew = new double[size];**

**for (int j = 0; j < size; j++){**

**double sum = 0;**

**for (int k = 0; k < size; k++){**

**if (k != j){**

**sum += R[j, k] \* x[k];**

**}**

**}**

**xNew[j] = (b[j] - sum) / D[j];**

**}**

**iterations++;**

**converged = true;**

**for (int j = 0; j < size; j++)**

**{**

**if (Math.Abs(x[j] - xNew[j]) > epsilon)**

**{**

**converged = false;**

**break;**

**}**

**}**

**x = xNew;**

**Console.Write($"Iteratia {iterations}: ");**

**PrintActualSolution(x);**

**}**

**}**

*IterativeJacobi / Program.cs*

**static void PrintActualSolution(double[] x)**

**{**

**for (int i = 0; i < x.Length; i++)**

**{**

**Console.Write($"X{i + 1} = {x[i]:f6}; ");**

**}**

**Console.WriteLine();**

**}**

**}**

**}**

**A screenshot of a computer

Description automatically generatedRezultatele testării**

**2.4 Metoda Gauss-Seidel**

Metoda Gauss-Seidel este o metodă iterativă numerică utilizată pentru rezolvarea sistemelor de ecuații liniare de forma: *A⋅X=B* unde:

A este o matrice pătratică n×nn

X este un vector al necunoscutelor [x1,x2,…,xn]

B este un vector constant [b1,b2,…,bn]

Această metodă este o îmbunătățire a metodei Jacobi, deoarece folosește imediat valorile calculate la iterația curentă pentru a accelera convergența.

**Rezolvarea analitică**

Rezolvarea unui sistem de 4 ecuații liniare cu 4 necunoscute cu precizie **ε=0,01**. La primul pas, sistemul dat este scris într-o formă prietenoasă pentru iterații. Pentru aceasta, rezolvăm prima ecuație a sistemului pentru necunoscutul x1 , a doua - pentru x2 , a treia - pentru x3 și așa mai departe. Ca urmare, vom avea:

**A number and numbers on a white background

Description automatically generatedA math equations with numbers

Description automatically generated with medium confidence**

În continuare, luând aproximarea inițială a rădăcinilor ca valoare x 1(0) = 0,889, x 2(0) = 0,72, x 3(0) = -1,506, x 4(0) = 0,868 și înlocuindu-le în sistemul obținut în pasul precedent, am determinat primele valori aproximative ale rădăcinilor dorite. Am verificat condiția de terminare a procesului iterativ, adică găsim valoarea modulului diferenței dintre elementele corespunzătoare ale vectorilor.

****

**A number equation with numbers

Description automatically generated with medium confidenceA number with numbers and symbols

Description automatically generated with medium confidence**Prin urmare, continuăm procesul iterativ. În pasul următor, înlocuim valorile obținute în sistemul pregătit pentru calcul convenabil și, astfel, găsim a doua aproximare. Pentru această aproximare, verificăm din nou starea de oprire.

****

Rezolvarea unui sistem de 4 ecuații liniare cu 4 necunoscute cu precizie **ε=0,001.** Rezolvarea cuprinde pașii decsriși mai sus, doar că din cauza preciziei mai mari a rezultatului, numărul de iterații necesar este mai mare.

**A math equations with numbers

Description automatically generated with medium confidenceA number and numbers on a white background

Description automatically generated**În continuare, luând aproximarea inițială a rădăcinilor ca valoare x 1(0) = 0,889, x 2(0) = 0,72, x 3(0) = -1,506, x 4(0) = 0,868 și înlocuindu-le în sistemul obținut în pasul precedent, găsim primele valori aproximative ale rădăcinilor dorite.

A number with numbers on it

Description automatically generated







A number with numbers and lines

Description automatically generated with medium confidenceA number of numbers and symbols

Description automatically generated with medium confidence



**Listingul programului**

**using InputData;**

**namespace GaussSeidel**

**{**

**internal class Program**

**{**

**public static double Epsilon { get; set; } = 0.00001;**

**public static int MaxIterations { get; set; } = 100;**

**public static double[] value0 { get; set; } = new double[InpData.n];**

**public static double[] value1 { get; set; } = new double[InpData.n];**

**public static double[] epsilons { get; set; } = new double[InpData.n];**

**static void Main(string[] args)**

**{**

**Console.WriteLine("Metoda lui Gauss-Seidel pentru Ax = B");**

**int iter = 0;**

**Console.WriteLine("\t\tX1\t\tX2\t\tX3\t\tX4");**

**do**

**{**

**value1[0] = InpData.X1(value0[1], value0[2], value0[3]);**

**value1[1] = InpData.X2(value0[0], value0[2], value0[3]);**

**value1[2] = InpData.X3(value0[0], value0[1], value0[3]);**

**value1[3] = InpData.X4(value0[0], value0[1], value0[2]);**

**Console.Write($"Iteratia {iter}: ");**

**PrintActualSolution();**

**doEpsilons(); iter++;**

**Transfer();**

**} while (CheckEpsilons() && iter <= MaxIterations);**

**}**

**static void PrintActualSolution()**

**{**

**for (int i = 0; i < value1.Length; i++)**

**{**

**Console.Write($"X{i + 1} = {value1[i]:f6}; ");**

**}**

**Console.WriteLine();**

**}**

**static void doEpsilons()**

**{**

**for (int i = 0; i < value0.Length; i++)**

**{**

**epsilons[i] = Math.Abs(value0[i] - value1[i]);**

**}**

**}**

**static void Transfer()**

**{**

**for (int i = 0; i < value0.Length; i++)**

**{**

**value0[i] = value1[i];**

**}**

**}**

**static bool CheckEpsilons()**

**{**

**for (int i = 0; i < epsilons.Length; i++)**

**if (epsilons[i] < Epsilon)**

**return false;**

**return true;**

**}**

**}**

**}**

*GaussSeidel / Program.cs*

**Rezultatele testării**

**A black screen with white text

Description automatically generated**

# **Concluzii**

Rezolvarea sistemelor de ecuații liniare A⋅X= B este esențială în numeroase aplicații științifice și inginerești, iar alegerea metodei potrivite poate influența semnificativ eficiența și precizia calculului. În cadrul acestui laborator, s-au utilizat patru metode distincte: eliminarea lui Gauss, metoda Cholesky, metoda iterativă a lui Jacobi și metoda iterativă a lui Gauss-Seidel. Aceste metode au fost evaluate în funcție de acuratețe, convergență și timp de execuție, oferind perspective variate asupra modului în care problemele pot fi abordate, în funcție de tipul și dimensiunea matricii.

Lucrarea a fost implementată utilizând limbajul de programare C#, cu ajutorul mediului de dezvoltare Visual Studio, cunoscut pentru flexibilitatea și eficiența sa. Alegerea C# s-a bazat pe capacitățile avansate de manipulare a matricilor și suportul excelent pentru programarea orientată pe obiecte, facilitând implementarea logicii algoritmilor pentru rezolvarea sistemelor de ecuații liniare.

A white grid with black text

Description automatically generatedMetoda eliminării lui Gauss s-a remarcat prin generalitate și robusteză, fiind aplicabilă pentru orice matrice nesingulară. Metoda Cholesky, deși limitată la matrici simetrice și pozitiv definite, a demonstrat o eficiență ridicată, fiind ideală pentru sisteme mari cu astfel de proprietăți. Metodele iterative, Jacobi și Gauss-Seidel, au oferit soluții alternative bazate pe aproximări succesive, evidențiind avantajul unui consum redus de resurse pentru matrici. Dintre acestea, Gauss-Seidel a avut o convergență mai rapidă, dar ambele metode au fost dependente de condițiile impuse de matrice.

Importanța acestor metode constă în adaptabilitatea lor la diferite tipuri de probleme, permițând rezolvarea sistemelor liniare în moduri eficiente și precise. Alegerea metodei adecvate implică o înțelegere profundă a structurii matricii și a cerințelor aplicației. Acest laborator subliniază relevanța acestor tehnici în informatică și matematică, oferind un cadru practic pentru utilizarea lor în contexte reale, cum ar fi simulările, optimizările și alte domenii computaționale.

1. **Webobrafie**

* Curs ***Metode numerice*** <https://else.fcim.utm.md/course/view.php?id=1689>
* Suport curs ***Metode numerice*** <https://elth.ucv.ro/fisiere/probleme%20studentesti/Cursuri/Metode%20numerice/curs_met_num.pdf>
* Calculator online Decompoziția Cholesky [Cholesky Decomposition Calculator](https://www.omnicalculator.com/math/cholesky-decomposition)
* Calculator online Metoda eliminarea lui Gauss [Gaussian elimination calculator](https://onlinemschool.com/math/assistance/equation/gaus/)
* Calculator online Gauss-Seidel <https://www.mathros.net.ua/en/gauss-seidel-method-calculator>
* Inteligență artificială <https://chatgpt.com/>