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

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):

$$A = \begin{pmatrix} 8.1 & -0.9 & 0.6 & 0.8 \\ -0.9 & 14.3 & 0.3 & 0.7 \\ 0.6 & 0.3 & 7.9 & -0.4 \\ 0.8 & 0.7 & -0.4 & 10.6 \end{pmatrix} \quad B = \begin{pmatrix} 7.2 \\ 10.3 \\ -11.9 \\ 9.2 \end{pmatrix}$$

2. 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

$$A=\begin{pmatrix} 8.1 & -0.9 & 0.6 & 0.8 \\ -0.9 & 14.3 & 0.3 & 0.7 \\ 0.6 & 0.3 & 7.9 & -0.4 \\ 0.8 & 0.7 & -0.4 & 10.6 \end{pmatrix} \quad B=\begin{pmatrix} 7.2 \\ 10.3 \\ -11.9 \\ 9.2 \end{pmatrix} \quad => \quad \begin{cases} 8.1x_1 - 0.9x_2 + 0.6x_3 + 0.8x_4 = 7.2 \\ -0.9x_1 + 14.3x_2 + 0.3x_3 + 0.7x_4 = 10.3 \\ 0.6x_1 + 0.3x_2 + 7.9x_3 - 0.4x_4 = -11.9 \\ 0.8x_1 + 0.7x_2 - 0.4x_3 + 10.6x_4 = 9.2 \end{cases}$$

Transformarea matricei într-o formă triunghiulară superioară

1. Pivotarea pe primul rând (eliminăm elementele de sub a_{11})

$$\begin{cases} x_1 - 0.1111x_2 + 0.0741x_3 + 0.8889x_4 = 0.8889 \\ -0.9x_1 + 14.3x_2 + 0.3x_3 + 0.7x_4 = 10.3 \\ 0.6x_1 + 0.3x_2 + 7.9x_3 - 0.4x_4 = -11.9 \\ 0.8x_1 + 0.7x_2 - 0.4x_3 + 10.6x_4 = 9.2 \end{cases} \quad \begin{cases} x_1 - 0.1111x_2 + 0.0741x_3 + 0.8889x_4 = 0.8889 \\ x_1 + 14.4x_2 + 0.367x_3 + 0.789x_4 = 11.1 \\ x_1 + 0.367x_2 + 7.8556x_3 - 0.4593x_4 = -12.4333 \\ x_1 + 0.7889x_2 - 0.4593x_3 + 10.521x_4 = 8.4889 \end{cases}$$

Divizăm primul rând la 8.1

Eliminăm a_{21}, a_{31}, a_{41} prin transformările:
$R_2 = R_2 + 0.9*R_1$ | $R_3 = R_3 - 0.6*R_1$ | $R_4 = R_4 - 0.8*R_1$

2. Pivotarea pe al doilea rând (eliminăm elementele de sub a_{22})

$$\begin{cases} x_1 - 0.1111x_2 + 0.0741x_3 + 0.8889x_4 = 0.8889 \\ x_2 + 0.0255x_3 + 0.0548x_4 = 0.7708 \\ 0.3667x_2 + 7.8556x_3 - 0.4593x_4 = -12.4333 \\ 0.7889x_2 - 0.4593x_3 + 10.521x_4 = 8.4889 \end{cases} \quad \begin{cases} x_1 - 0.1111x_2 + 0.0741x_3 + 0.8889x_4 = 0.8889 \\ x_2 + 0.0255x_3 + 0.0548x_4 = 0.7708 \\ 7.8556x_3 - 0.4593x_4 = -12.4333 \\ -0.4794x_3 + 10.4778x_4 = 7.8833 \end{cases}$$

Divizăm al doilea rând la 14.4

Eliminăm a_{32}, a_{42} prin transformările:
$R_3 = R_3 - 0.3667*R_2$ | $R_4 = R_4 - 0.7889*R_2$

3. Pivotarea pe al treilea rând (eliminăm elementele de sub a_{33})

$$\begin{cases} x_1 - 0.1111x_2 + 0.0741x_3 + 0.8889x_4 = 0.8889 \\ x_2 + 0.0255x_3 + 0.0548x_4 = 0.7708 \\ x_3 - 0.0611x_4 = -1.6214 \\ -0.4794x_3 + 10.4778x_4 = 7.8833 \end{cases} \quad \begin{cases} x_1 - 0.1111x_2 + 0.0741x_3 + 0.8889x_4 = 0.8889 \\ x_2 + 0.0255x_3 + 0.0548x_4 = 0.7708 \\ x_3 - 0.0611x_4 = -1.6214 \\ 10.4484x_4 = 7.1053 \end{cases}$$

Divizăm al doilea rând la 7.8462 *Eliminăm a_{43} prin transformările: $R_4 = R_4 + 0.4794*R_3$*

4. Pivotarea pe al patrulea rând (eliminăm elementele de sub a_{44})

$$\begin{cases} x_1 - 0.1111x_2 + 0.0741x_3 + 0.8889x_4 = 0.8889 \\ x_2 + 0.0255x_3 + 0.0548x_4 = 0.7708 \\ x_3 - 0.0611x_4 = -1.6214 \\ x_4 = 0.6800 \end{cases} \quad => x_1 = 0.999, x_2 = 0.912, x_3 = -1.58, x_4 = 0.68$$

Divizăm al doilea rând la 10.4484

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()

```
{
   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;
}
```

static public double[] GetB()

```
{
   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()

```
{
  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();
   }
}
```

static public void PrintInputData()

```
{
  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}");
            }
        }
    }
}
```

Rezultatele testării

```
Metoda eliminarii lui Gauss pentru Ax = B
Matricea A:
  8.1000 -0.9000  0.6000  0.8000
 -0.9000 14.3000  0.3000  0.7000
  0.6000  0.3000  7.9000 -0.4000
  0.8000  0.7000 -0.4000 10.6000
Vectorul B:
  7.2000
 10.3000
-11.9000
  9.2000
Solutia este:
x1 = 1.0260
x2 = 0.7848
x3 = -1.5797
x4 = 0.6791
```

2.2 Metoda lui Cholesky

Metoda Cholesky presupune factorizarea matricei simetrice și pozitiv definite AAA în produsul a două matrice: $A=L\cdot L^{T,}$ unde L este o matrice triunghiulară inferioară, iar $L^{T,}$ este transpusa sa. Apoi rezolvăm sistemul A·X=B prin rezolvarea a două sisteme succesive:

$L\cdot Y=B$ (prin substituție înainte);

$LT\cdot X=Y$ (prin substituție înapoi).

Rezolvarea analitică

Verificarea simetriei și pozitivitatea matricei

1. Matricea A este simetrică. Pentru verificarea pozitivității matricei, calculăm determinanții principalilor minori (M_1, M_2, M_3, M_4). Dacă toți sunt pozitivi, matricea este pozitiv definită.

$$A=\begin{pmatrix} 8.1 & -0.9 & 0.6 & 0.8 \\ -0.9 & 14.3 & 0.3 & 0.7 \\ 0.6 & 0.3 & 7.9 & -0.4 \\ 0.8 & 0.7 & -0.4 & 10.6 \end{pmatrix}$$

$$M_1 = 8.1 > 0, \quad M_2 = \det\begin{bmatrix} 8.1 & -0.9 \\ -0.9 & 14.3 \end{bmatrix} = (8.1)(14.3) - (-0.9)^2 = 115.83 > 0,$$

$M_3 > 0$ (calcul intermediar), $M_4 > 0$.

Factorizarea Cholesky

1. Diagonală: $L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}$,
2. Elemente subdiagonale: $L_{ij} = \frac{1}{L_{ii}}\left(A_{ij} - \sum_{k=1}^{i-1} L_{ik}L_{jk}\right)$.
   1. $L_{11} = \sqrt{8.1} = 2.846$,
   2. $L_{21} = \frac{-0.9}{2.846} = -0.316$,
   3. $L_{22} = \sqrt{14.3 - (-0.316)^2} = \sqrt{14.3 - 0.1} = 3.776$,
   4. $L_{31} = \frac{0.6}{2.846} = 0.211$,
   5. $L_{32} = \frac{0.3-(0.211)(-0.316)}{3.776} = \frac{0.3+0.067}{3.776} = 0.097$,
   6. $L_{33} = \sqrt{7.9 - (0.211)^2 - (0.097)^2} = 2.807$,
   7. $L_{41} = \frac{0.8}{2.846} = 0.281$,
   8. $L_{42} = \frac{0.7-(0.281)(-0.316)}{3.776} = \frac{0.7+0.089}{3.776} = 0.209$,
   9. $L_{43} = \frac{-0.4-(0.281)(0.211)-(0.209)(0.097)}{2.807} = -0.162$,
   10. $L_{44} = \sqrt{10.6 - (0.281)^2 - (0.209)^2 - (-0.162)^2} = 3.220$.

$$L = \begin{bmatrix} 2.846 & 0 & 0 & 0 \\ -0.316 & 3.768 & 0 & 0 \\ 0.211 & 0.097 & 2.801 & 0 \\ 0.281 & 0.209 & -0.171 & 3.232 \end{bmatrix}$$

$$L\cdot L^T = \begin{bmatrix} 8.1 & -0.9 & 0.6 & 0.8 \\ -0.9 & 14.3 & 0.3 & 0.7 \\ 0.6 & 0.3 & 7.9 & -0.4 \\ 0.8 & 0.7 & -0.4 & 10.6 \end{bmatrix}$$

Factorizarea Colesky determinată cu ajutorul Calculator de descompunere

Rezolvarea sistemelor succesive:

- $x_1 = 0.89,\ x_2 = 0.79,\ x_3 = -1.61,\ x_4 = 0.52$

Listingul programului

Cholesky / Program.cs

```
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);
    }
```

```
    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();
       }
     }
    }
}
```

Rezultatele testării

```
Metoda lui Cholecky pentru Ax = B
Matricea A:
  8.1000 -0.9000  0.6000  0.8000
 -0.9000 14.3000  0.3000  0.7000
  0.6000  0.3000  7.9000 -0.4000
  0.8000  0.7000 -0.4000 10.6000
Matricea L:
  2.8460  0.0000  0.0000  0.0000
 -0.3162  3.7683  0.0000  0.0000
  0.2108  0.0973  2.8011  0.0000
  0.2811  0.2093 -0.1712  3.2323
Matricea L transpusa:
  2.8460 -0.3162  0.2108  0.2811
  0.0000  3.7683  0.0973  0.2093
  0.0000  0.0000  2.8011 -0.1712
  0.0000  0.0000  0.0000  3.2323
Verificarea rezultatelor:
Produsul matricelor L*Lt:
  8.1000 -0.9000  0.6000  0.8000
 -0.9000 14.3000  0.3000  0.7000
  0.6000  0.3000  7.9000 -0.4000
  0.8000  0.7000 -0.4000 10.6000
```

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.

Pașii metodei Jacobi:

- Se rescrie fiecare ecuație din sistem astfel încât necunoscuta să fie izolată $x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^{n} a_{ij} x_j^{(k)} \right)$
- 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

Rescrierea sistemului pentru metoda Jacobi. Se izolează fiecare necunoscută și selectăm valorile inițiale de la care vom începe calculul iterativ.

$$
\begin{aligned}
x_1^{(k+1)} &= \frac{1}{8.1}\left(7.2 + 0.9x_2^{(k)} - 0.6x_3^{(k)} - 0.8x_4^{(k)}\right), \\
x_2^{(k+1)} &= \frac{1}{14.3}\left(10.3 + 0.9x_1^{(k)} - 0.3x_3^{(k)} - 0.7x_4^{(k)}\right), \\
x_3^{(k+1)} &= \frac{1}{7.9}\left(-11.9 - 0.6x_1^{(k)} - 0.3x_2^{(k)} + 0.4x_4^{(k)}\right), \\
x_4^{(k+1)} &= \frac{1}{10.6}\left(9.2 - 0.8x_1^{(k)} - 0.7x_2^{(k)} + 0.4x_3^{(k)}\right).
\end{aligned}
$$

$$
\begin{aligned}
x_1^{(1)} &= \frac{1}{8.1}(7.2 + 0.9(0) - 0.6(0) - 0.8(0)) = \frac{7.2}{8.1} = 0.889, \\
x_2^{(1)} &= \frac{1}{14.3}(10.3 + 0.9(0) - 0.3(0) - 0.7(0)) = \frac{10.3}{14.3} = 0.721, \\
x_3^{(1)} &= \frac{1}{7.9}(-11.9 - 0.6(0) - 0.3(0) + 0.4(0)) = \frac{-11.9}{7.9} = -1.506, \\
x_4^{(1)} &= \frac{1}{10.6}(9.2 - 0.8(0) - 0.7(0) + 0.4(0)) = \frac{9.2}{10.6} = 0.868.
\end{aligned}
$$

$$x_1^{(0)} = 0, \quad x_2^{(0)} = 0, \quad x_3^{(0)} = 0, \quad x_4^{(0)} = 0.$$

Izolarea necunoscutelor.
Inițializarea variabilelor inițiale.

$$x_1^{(1)} = 0.889, \quad x_2^{(1)} = 0.721, \quad x_3^{(1)} = -1.506, \quad x_4^{(1)} = 0.868.$$

Valoarea variabilelor după prima iterație.

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:

$$x_1 = 0.89, \quad x_2 = 0.79, \quad x_3 = -1.61, \quad x_4 = 0.52.$$

Listingul programului

IterativeJacobi / Program.cs

```
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);
         }
      }
```

```
        static void PrintActualSolution(double[] x)
        {
            for (int i = 0; i < x.Length; i++)
            {
                Console.Write($"X{i + 1} = {x[i]:f6}; ");
            }
            Console.WriteLine();
        }
    }
}
```

Rezultatele testării

```
Metoda iterativa a lui Jacobi pentru Ax = B
Matricea A:
  8.1000 -0.9000  0.6000  0.8000
 -0.9000 14.3000  0.3000  0.7000
  0.6000  0.3000  7.9000 -0.4000
  0.8000  0.7000 -0.4000 10.6000
Vectorul B:
  7.2000
 10.3000
-11.9000
  9.2000
                X1              X2              X3              X4
Iteratia 1: X1 = 0.888889; X2 = 0.720280; X3 = -1.506329; X4 = 0.867925;
Iteratia 2: X1 = 0.994779; X2 = 0.765339; X3 = -1.557247; X4 = 0.696430;
Iteratia 3: X1 = 1.020495; X2 = 0.781467; X3 = -1.575683; X4 = 0.683542;
Iteratia 4: X1 = 1.024926; X2 = 0.784103; X3 = -1.578901; X4 = 0.679840;
Iteratia 5: X1 = 1.025822; X2 = 0.784630; X3 = -1.579525; X4 = 0.679210;
Iteratia 6: X1 = 1.025989; X2 = 0.784731; X3 = -1.579645; X4 = 0.679084;
Iteratia 7: X1 = 1.026022; X2 = 0.784750; X3 = -1.579668; X4 = 0.679060;
Iteratia 8: X1 = 1.026028; X2 = 0.784754; X3 = -1.579673; X4 = 0.679056;
```

2.4 Metoda Gauss-Seidel

Metoda Gauss-Seidel este o metodă iterativă numerică utilizată pentru rezolvarea sistemelor de ecuații liniare de forma: $A \cdot X = B$ 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 x_1 , a doua - pentru x_2 , a treia - pentru x_3 și așa mai departe. Ca urmare, vom avea:

$$\begin{cases} 8.1 \cdot x_1 - 0{,}9 \cdot x_2 + 0{,}6 \cdot x_3 + 0{,}8 \cdot x_4 = 7.2 \\ -0{,}9 \cdot x_1 + 14.3 \cdot x_2 + 0{,}3 \cdot x_3 + 0{,}7 \cdot x_4 = 10.3 \\ 0{,}6 \cdot x_1 + 0{,}3 \cdot x_2 + 7.9 \cdot x_3 - 0{,}4 \cdot x_4 = -11.9 \\ 0{,}8 \cdot x_1 + 0{,}7 \cdot x_2 - 0{,}4 \cdot x_3 + 10.6 \cdot x_4 = 9.2 \end{cases} ; \quad \begin{cases} x_1 = 0{,}889 + 0{,}111 \cdot x_2 - 0{,}074 \cdot x_3 - 0{,}099 \cdot x_4 \\ x_2 = 0{,}72 + 0{,}063 \cdot x_1 - 0{,}021 \cdot x_3 - 0{,}049 \cdot x_4 \\ x_3 = -1.506 - 0{,}076 \cdot x_1 - 0{,}038 \cdot x_2 + 0{,}051 \cdot x_4 \\ x_4 = 0{,}868 - 0{,}075 \cdot x_1 - 0{,}066 \cdot x_2 + 0{,}038 \cdot x_3 \end{cases}$$

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.

$$\left|x_4^{(1)} - x_4^{(0)}\right| = |0,683 - 0,868| = 0,185 > 0,1;$$

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.

$$\begin{cases} x_1^{(2)} = 0,889 + 0,111 \cdot x_2^{(1)} - 0,074 \cdot x_3^{(1)} - 0,099 \cdot x_4^{(1)} = 0,889 + 0,111 \cdot 0,772 - 0,074 \cdot (-1.567) - 0,099 \cdot 0,683 = 1.023 \\ x_2^{(2)} = 0,72 + 0,063 \cdot x_1^{(2)} - 0,021 \cdot x_3^{(1)} - 0,049 \cdot x_4^{(1)} = 0,72 + 0,063 \cdot 1.023 - 0,021 \cdot (-1.567) - 0,049 \cdot 0,683 = 0,784 \\ x_3^{(2)} = -1.506 - 0,076 \cdot x_1^{(2)} - 0,038 \cdot x_2^{(2)} + 0,051 \cdot x_4^{(1)} = -1.506 - 0,076 \cdot 1.023 - 0,038 \cdot 0,784 + 0,051 \cdot 0,683 = -1.579 \\ x_4^{(2)} = 0,868 - 0,075 \cdot x_1^{(2)} - 0,066 \cdot x_2^{(2)} + 0,038 \cdot x_3^{(2)} = 0,868 - 0,075 \cdot 1.023 - 0,066 \cdot 0,784 + 0,038 \cdot (-1.579) = 0,68 \end{cases};$$

$$\left|x_1^{(2)} - x_1^{(1)}\right| = |1.023 - 0,994| = 0,029 <= 0,1;$$

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.

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.

$$\begin{cases} 8.1 \cdot x_1 - 0,9 \cdot x_2 + 0,6 \cdot x_3 + 0,8 \cdot x_4 = 7.2 \\ -0,9 \cdot x_1 + 14.3 \cdot x_2 + 0,3 \cdot x_3 + 0,7 \cdot x_4 = 10.3 \\ 0,6 \cdot x_1 + 0,3 \cdot x_2 + 7.9 \cdot x_3 - 0,4 \cdot x_4 = -11.9 \\ 0,8 \cdot x_1 + 0,7 \cdot x_2 - 0,4 \cdot x_3 + 10.6 \cdot x_4 = 9.2 \end{cases}; \quad \begin{cases} x_1 = 0,889 + 0,111 \cdot x_2 - 0,074 \cdot x_3 - 0,099 \cdot x_4 \\ x_2 = 0,72 + 0,063 \cdot x_1 - 0,021 \cdot x_3 - 0,049 \cdot x_4 \\ x_3 = -1.506 - 0,076 \cdot x_1 - 0,038 \cdot x_2 + 0,051 \cdot x_4 \\ x_4 = 0,868 - 0,075 \cdot x_1 - 0,066 \cdot x_2 + 0,038 \cdot x_3 \end{cases}$$

$$\left|x_4^{(1)} - x_4^{(0)}\right| = |0,683 - 0,868| = 0,185 > 0,001;$$

$$\left|x_1^{(2)} - x_1^{(1)}\right| = |1.023 - 0,994| = 0,029 > 0,001;$$

$$\left|x_1^{(3)} - x_1^{(2)}\right| = |1.026 - 1.023| = 0,003 > 0,001;$$

$$\left|x_2^{(4)} - x_2^{(3)}\right| = |0,785 - 0,784| = 0,001 > 0,001;$$

$$\begin{cases} x_1^{(5)} = 0,889 + 0,111 \cdot x_2^{(4)} - 0,074 \cdot x_3^{(4)} - 0,099 \cdot x_4^{(4)} = 0,889 + 0,111 \cdot 0,785 - 0,074 \cdot (-1.579) - 0,099 \cdot 0,679 = 1.026 \\ x_2^{(5)} = 0,72 + 0,063 \cdot x_1^{(5)} - 0,021 \cdot x_3^{(4)} - 0,049 \cdot x_4^{(4)} = 0,72 + 0,063 \cdot 1.026 - 0,021 \cdot (-1.579) - 0,049 \cdot 0,679 = 0,785 \\ x_3^{(5)} = -1.506 - 0,076 \cdot x_1^{(5)} - 0,038 \cdot x_2^{(5)} + 0,051 \cdot x_4^{(4)} = -1.506 - 0,076 \cdot 1.026 - 0,038 \cdot 0785 + 0,051 \cdot 0,679 = -1.579 \\ x_4^{(5)} = 0,868 - 0,075 \cdot x_1^{(5)} - 0,066 \cdot x_2^{(5)} + 0,038 \cdot x_3^{(5)} = 0,868 - 0,075 \cdot 1.026 - 0,066 \cdot 0,785 + 0,038 \cdot (-1.579) = 0,679 \end{cases};$$

$$\left|x_1^{(5)} - x_1^{(4)}\right| = |1.026 - 1.026| = 0 <= 0,001;$$

Listingul programului

GaussSeidel / Program.cs

```
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;
        }
    }
}
```

Rezultatele testării

```
Metoda lui Gauss-Seidel pentru Ax = B
                X1              X2              X3              X4
Iteratia 0: X1 = 0.888889; X2 = 0.720280; X3 = -1.506329; X4 = 0.867925;
Iteratia 1: X1 = 0.994779; X2 = 0.765339; X3 = -1.557247; X4 = 0.696430;
Iteratia 2: X1 = 1.020495; X2 = 0.781467; X3 = -1.575683; X4 = 0.683542;
Iteratia 3: X1 = 1.024926; X2 = 0.784103; X3 = -1.578901; X4 = 0.679840;
Iteratia 4: X1 = 1.025822; X2 = 0.784630; X3 = -1.579525; X4 = 0.679210;
Iteratia 5: X1 = 1.025989; X2 = 0.784731; X3 = -1.579645; X4 = 0.679084;
Iteratia 6: X1 = 1.026022; X2 = 0.784750; X3 = -1.579668; X4 = 0.679060;
Iteratia 7: X1 = 1.026028; X2 = 0.784754; X3 = -1.579673; X4 = 0.679056;
```

5. 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.

Metoda 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.

Metodă	Precizie	Convergență	Timp de execuție	Observații
Eliminarea Gauss	Ridicată	Garanție completă	Rapid	Funcționează pentru orice matrice nesingulară.
Cholesky	Foarte ridicată	Condiționată	Foarte rapid	Eficientă doar pentru matrici simetrice și pozitiv definite.
Jacobi	Acceptabilă	Condiționată	Lent	Simplu, dar mai lent și mai puțin robust.
Gauss-Seidel	Ridicată	Condiționată	Moderat	Convergență mai rapidă decât Jacobi.

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.

6. 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
- Calculator online Metoda eliminarea lui Gauss Gaussian elimination calculator
- Calculator online Gauss-Seidel https://www.mathros.net.ua/en/gauss-seidel-method-calculator
- Inteligență artificială https://chatgpt.com/