**MINISTERUL EDUCAŢIEI**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică și Microelectronică**

**Tehnologia Informației**

**Raport**

**Disciplina:**Metode numerice

**Lucrarea de laborator nr. 2**

**Tema:** *“* REZOLVAREA NUMERICĂ A SISTEMELOR DE ECUAŢII LINIARE”

Varianta: 23

|  |  |  |
| --- | --- | --- |
| **Student:** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Raevschi Grigore TI-231** |
| **Coordonator:** | **\_\_\_\_\_\_\_\_\_\_\_\_\_** | **Conf. univ. Pațiuc Vladimir** |
|  |  |  |

Chișinău 2024

Cuprins

[**Scopul lucrării** 3](#_Toc181216307)

[**Ecuația liniară propusă spre rezolvare:** 3](#_Toc181216308)

[**Rezolvarea manuală ( Metoda eliminării lui Gauss )** 3](#_Toc181216309)

[**Codul programului(scris în limbajul Java)** 7](#_Toc181216310)

[**Rezultatul programului prin metoda lui Cholesky** 16](#_Toc181216311)

[**Rezultatul programului prin metoda Jacobi** 16](#_Toc181216312)

[**Rezultatul programului prin metoda Gauss-Seidel** 16](#_Toc181216313)

[**Compararea rezultatelor** 17](#_Toc181216314)

[**Concluzie** 17](#_Toc181216315)

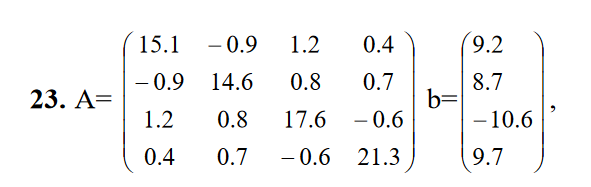
# **Scopul lucrării**

1. Să se rezolve sistemul de ecuaţii lineare , utilizând

* Metoda eliminării lui Gauss;
* Metoda lui Cholesky (metoda rădăcinii pătrate);
* Metoda iterativă a lui Jacobi cu o eroare ;
* Metoda iterativă a lui Gauss-Seidel cu o eroare şi

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

# **Ecuația liniară propusă spre rezolvare:**



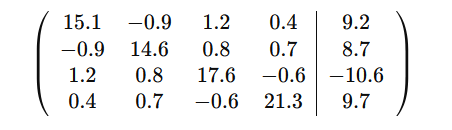
Figură 1: Varianta propusă spre rezolvare

# **Rezolvarea manuală ( Metoda eliminării lui Gauss )**

Pentru a rezolva complet sistemul dat folosind metoda eliminării Gauss, voi parcurge pașii detaliați pentru a transforma matricea extinsă (A∣b)(A|b)(A∣b) într-o formă treptată și a obține soluția sistemului.

**Pasul 1: Formarea matricei extinse**

Formăm matricea extinsă (A∣b)(A|b)(A∣b):



**Pasul 2: Aplicarea eliminării Gauss pentru a obține un 1 în colțul stânga-sus (pivot)**

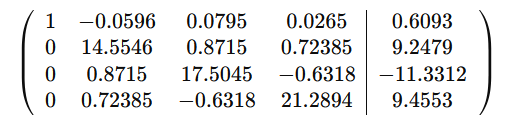
Împărțim prima linie la 15.1 pentru a obține pivotul 1 în poziția (1,1).

A number of numbers on a white background

Description automatically generated

**Pasul 3: Anularea elementelor de sub pivotul (1,1)**

Pentru a anula elementele sub pivotul din prima coloană, adăugăm sau scădem multiplii primei linii la liniile 2, 3 și 4.



**Pasul 4: Obținerea unui pivot 1 pe diagonala principală în poziția (2,2)**

Împărțim a doua linie la 14.5546 pentru a obține un pivot 1.

A group of numbers with black text

Description automatically generated

**Pasul 5: Anularea elementelor sub pivotul (2,2)**

Anulăm elementele de sub pivotul de pe poziția (2,2) prin operații pe linii.

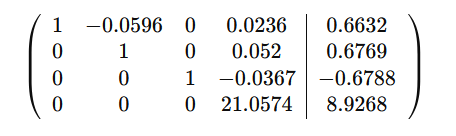
A number of numbers and a number of digits

Description automatically generated with medium confidence

**Pasul 6: Anularea elementelor de deasupra și sub pivotul (3,3)**

Acum anulăm elementele de deasupra și sub pivotul din coloana a treia (pozitia 3,3) folosind a treia linie:

1. Pentru a doua linie, scădem 0.0599 ori linia a treia.
2. Pentru prima linie, scădem 0.0795 ori linia a treia.
3. Pentru a patra linie, adăugăm 0.6318 ori linia a treia.



**Pasul 7: Obținerea unui pivot de 1 pe poziția (4,4)**

Pentru a obține un pivot de 1 în poziția (4,4), împărțim a patra linie la 21.057421 :

A number and numbers on a white background

Description automatically generated

**Pasul 8: Anularea elementelor de deasupra pivotului (4,4)**

Folosim a patra linie pentru a anula elementele de deasupra pivotului din coloana a patra:

1. Pentru a treia linie, adunăm 0.03670 ori linia a patra.
2. Pentru a doua linie, scădem 0.0520 ori linia a patra.
3. Pentru prima linie, scădem 0.02360 ori linia a patra.

A number of numbers and a number of numbers

Description automatically generated with medium confidence

**Rezultatul final al soluției**

# **Codul programului(scris în limbajul Java)**

import java.util.\*;

import java.io.\*;

import java.math.BigDecimal;

public class Lab2 {

    final static int lung = 4; // Updated to match the problem size (4x4 matrix)

    public static void main(String[] args) throws IOException {

        float[][] a = {

            {15.1f, -0.9f, 1.2f, 0.4f},

            {-0.9f, 14.6f, 0.8f, 0.7f},

            {1.2f, 0.8f, 17.6f, -0.6f},

            {0.4f, 0.7f, -0.6f, 21.3f}

        };

        float[] b = {9.2f, 8.7f, -10.6f, 9.7f};

        System.out.println("\nMatrix A:");

        for (float[] row : a) {

            for (float value : row) {

                System.out.print(value + "\t");

            }

            System.out.println();

        }

        System.out.println("\nVector b:");

        for (float value : b) {

            System.out.println(value);

        }

        GaussSeidel gaussSeidel = new GaussSeidel();

        Jacobi jacobi = new Jacobi();

        Cholesky cholesky = new Cholesky();

        float[][] l = new float[lung][lung]; // Initialize matrix l for Cholesky

        System.out.println("\nResults using Cholesky Method:");

        cholesky.cholesky(lung, b, l, a);

        System.out.println("\nResults using Jacobi Method (Error = 0.001):");

        jacobi.jacobi(lung, 0.001, a, b);

        System.out.println("\nResults using Gauss-Seidel Method (Error = 0.001):");

        gaussSeidel.gaussSeidel(lung, 0.001, a, b);

        System.out.println("\nResults using Gauss-Seidel Method (Error = 0.00001):");

        gaussSeidel.gaussSeidel(lung, 0.00001, a, b);

        // Menu-driven approach

        Scanner sc = new Scanner(System.in); // Scanner initialized here for menu input

        int menu;

        while (true) {

            System.out.println("1 - Metoda lui Cholesky");

            System.out.println("2 - Metoda Jacobi");

            System.out.println("3 - Metoda Gauss-Seidel");

            System.out.println("0 - Exit");

            System.out.print(">>> ");

            menu = sc.nextInt();

            try {

                switch (menu) {

                    case 1:

                        System.out.println("Rezultatele:");

                        System.out.println("\nMetoda Cholesky:\n");

                        cholesky.cholesky(lung, b, l, a);

                        System.out.print("\nIntroduceți orice caracter pentru a continua:\n>>> ");

                        System.in.read();

                        break;

                    case 2:

                        System.out.println("Rezultatele:");

                        System.out.println("\nMetoda Jacobi (Eroarea = 0.001):");

                        jacobi.jacobi(lung, 0.001, a, b);

                        System.out.print("\nIntroduceți orice caracter pentru a continua:\n>>> ");

                        System.in.read();

                        break;

                    case 3:

                        System.out.println("Rezultatele");

                        System.out.println("\nMetoda Gauss-Seidel (Eroarea = 0.001)");

                        gaussSeidel.gaussSeidel(lung, 0.001, a, b);

                        System.out.println("\nMetoda Gauss-Seidel (Eroarea = 0.00001)");

                        gaussSeidel.gaussSeidel(lung, 0.00001, a, b);

                        System.out.print("\nIntroduceți orice caracter pentru a continua:\n>>> ");

                        System.in.read();

                        break;

                    case 0:

                        System.out.println("Exiting the program.");

                        System.exit(0);

                    default:

                        System.out.println("Opțiune greșită!");

                        break;

                }

            } catch (IOException e) {

                System.out.println("An error occurred: " + e.getMessage());

            }

            System.out.println("\n\n\n");

        }

    }

}

class DiagonalaDominanta {

    protected int DiagDom(float[][] a, int lung) {

        float s;

        for (int i = 0; i < lung; i++) {

            s = 0;

            for (int j = 0; j < lung; j++) {

                if (i != j) {

                    s += a[i][j];

                }

            }

            if (a[i][i] < s) {

                return 0;

            } else if (a[i][i] == 0) {

                return 0;

            }

        }

        return 1;

    }

}

class GaussSeidel extends DiagonalaDominanta {

    public void gaussSeidel(int lung, double Eps, float[][] a, float[] b) throws IOException {

        float[] x = new float[20], x1 = new float[20], d = new float[20];

        float[][] q = new float[20][20];

        float t, max;

        int ni = 0;

        for (int i = 0; i < lung; i++) {

            for (int j = 0; j < lung; j++) {

                if (i != j) {

                    q[i][j] = -(a[i][j] / a[i][i]);

                } else {

                    q[i][j] = 0;

                }

            }

            if (DiagDom(a, lung) != 1) {

                System.out.println("Aceasta matricea nu este diagonal dominata");

                return;

            }

            d[i] = b[i] / a[i][i];

            x[i] = d[i];

        }

        do {

            for (int i = 0; i < lung; i++) {

                x1[i] = x[i];

            }

            for (int i = 0; i < lung; i++) {

                t = 0;

                for (int j = 0; j < lung; j++) {

                    t += (q[i][j] \* x[j]);

                }

                x[i] = ((float)t + d[i]);

            }

            max = (float) Math.abs(x[0] - x1[0]);

            for (int i = 1; i < lung; i++) {

                if ((float) Math.abs(x[i] - x1[i]) > max) {

                    max = (float) Math.abs(x[i] - x1[i]);

                }

            }

            ni++;

        } while (max > Eps);

        System.out.println("\nRezultatele x:");

        for (int i = 0; i < lung; i++) {

            System.out.println("x" + (i + 1) + " = " + BigDecimal.valueOf(x[i]));

        }

        System.out.println("\nNumarul iteratiilor = " + ni);

    }

}

class Jacobi extends DiagonalaDominanta {

    public void jacobi(int lung, double Eps, float[][] a, float[] b) throws IOException {

        float[] x = new float[20], x1 = new float[20], d = new float[20];

        float[][] q = new float[20][20];

        float t, max;

        int ni = 0;

        for (int i = 0; i < lung; i++) {

            for (int j = 0; j < lung; j++) {

                if (i != j) {

                    q[i][j] = -(a[i][j] / a[i][i]);

                } else {

                    q[i][j] = 0;

                }

            }

        }

        if (DiagDom(a, lung) != 1) {

            System.out.println("Aceasta matricea nu este diagonal dominata");

            return;

        }

        for (int i = 0; i < lung; i++) {

            d[i] = b[i] / a[i][i];

            x[i] = d[i];

        }

        do {

            for (int i = 0; i < lung; i++) {

                x1[i] = x[i];

            }

            for (int i = 0; i < lung; i++) {

                t = 0;

                for (int j = 0; j < lung; j++) {

                    t += q[i][j] \* x1[j];

                }

                x[i] = t + d[i];

            }

            max = (float) Math.abs(x[0] - x1[0]);

            for (int i = 1; i < lung; i++) {

                if ((float) Math.abs(x[i] - x1[i]) > max) {

                    max = (float) Math.abs(x[i] - x1[i]);

                }

            }

            ni++;

        } while (max > Eps);

        System.out.println("\nRezultatele x:");

        for (int i = 0; i < lung; i++) {

            System.out.println("x" + (i + 1) + " = " + BigDecimal.valueOf(x[i]));

        }

        System.out.println("\nNumarul iteratiilor = " + ni);

    }

}

class Cholesky {

    public void cholesky(int lung, float[] b, float[][] l, float[][] a) throws IOException {

        if (Pozitiv(lung, a) == 0) {

            System.out.println("\nAceasta matricea nu este pozitiv definita!");

            return;

        } else if (Simetric(lung, a) == 0) {

            System.out.println("\nAceasta matricea nu este simetrica!");

            return;

        }

        determinare(lung, b, l, a);

    }

    private void determinare(int lung, float[] b, float[][] l, float[][] a) {

        factor(lung, l, a);

        float[] y = new float[10], x = new float[10];

        float t;

        y[0] = b[0] / l[0][0];

        for (int i = 1; i < lung; i++) {

            t = 0;

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

                t += l[i][j] \* y[j];

            }

            y[i] = (b[i] - t) / l[i][i];

        }

        x[lung - 1] = y[lung - 1] / l[lung - 1][lung - 1];

        for (int i = lung - 2; i >= 0; i--) {

            t = 0;

            for (int j = lung - 1; j > i; j--) {

                t += l[j][i] \* x[j];

            }

            x[i] = (y[i] - t) / l[i][i];

        }

        for (int i = 0; i < lung; i++) {

            System.out.println("\nx" + (i + 1) + " = " + x[i]);

        }

    }

    private void factor(int lung, float[][] l, float[][] a) {

        float s;

        for (int i = 0; i < lung; i++) {

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

                s = 0;

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

                    s += l[i][k] \* l[j][k];

                }

                l[i][j] = (a[i][j] - s) / l[j][j];

            }

            s = 0;

            for (int k = 0; k < i; k++) {

                s += l[i][k] \* l[i][k];

            }

            l[i][i] = (float) Math.sqrt(a[i][i] - s);

        }

    }

    private int Pozitiv(int lung, float[][] a) {

        float d;

        for (int i = 0; i < lung; i++) {

            d = a[i][i];

            if (d <= 0) {

                return 0;

            }

        }

        return 1;

    }

    private int Simetric(int lung, float[][] a) {

        for (int i = 0; i < lung; i++) {

            for (int j = 0; j < lung; j++) {

                if (a[i][j] != a[j][i]) {

                    return 0;

                }

            }

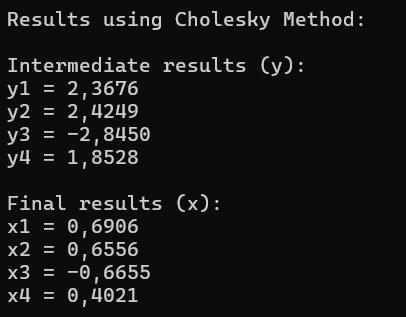
        }

        return 1;

    }

}

# **Rezultatul programului prin metoda lui Cholesky**



# **Rezultatul programului prin metoda Jacobi**

A computer screen with white text

Description automatically generated

# **Rezultatul programului prin metoda Gauss-Seidel**

A computer screen shot of a black screen

Description automatically generated

**Rezultatul programului prin metoda Gauss-Seidel**

A computer screen shot of a black screen

Description automatically generated

# **Compararea rezultatelor**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metoda | Radacina | | Iterații | Eroarea |
| x | y |  |  |
| Cholesy | x1 = 0,6906  x2 = 0,6556  x3 = -0,6655  x4 = 0,4021 | y1 = 2,3676  y2 = 2,4249  y3 = -2,8450  y4 = 1,8528 | - | = |
| Jacobi | x1 = 0.69054853916  x2 = 0.65561300516  x3 = -0.6654222011  x4 = 0.40215513110 |  | 4 | 0.001 |
| Gauss-Seidel | x1 = 0.690561413764  x2 = 0.655636906623  x3 = -0.66544818878  x4 = 0.402139008045 |  | 3 | 0.001 |
| x1 = 0.690580308437  x2 = 0.655642867088  x3 = -0.66545045375  x4 = 0.402138382196 |  | 5 | 0.00001 |

# **Concluzie**

În aceasta lucrare de laborator am facut cunostinta cu trei metode de rezolvare numerica a sistemelor de ecuatii liniare: Metoda Cholesky, Metoda Jacobi și Metoda Gauss - Seidel. În urma rezultatelor obținute din exemplul dat, cea mai eficientă metodă iterativă de calculare a rădăcinelor sistemelor de ecuații a fost metoda Gauss-Seidel.