Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică ăi Microelectronică
Departamentul Informatică Sofware și Automate

RAPORT

despre lucrarea de laborator nr. 2
la disciplina Metode și modele de calcul

Tema: Rezolvarea sistemelor de ecuații liniare

A efectuat: st. gr. TI-173 Heghea Nicolae

A verificat: conf. univ Tutunaru Eleonora

Chișinău 2018

Cuprins

1. Sarcina lucrării

1. Să se rezolve sistemul de ecuații liniare în forma matricială $Ax = b$.

A_{nxm} → n, m se citesc de la tastieră.

Se afișează matricea Q, și vectorul D, soluția sistemului.

Și dacă condiția nu are loc, să afișeze ce să facem.

2. Noțiuni generale metoda iterativă Jacobi

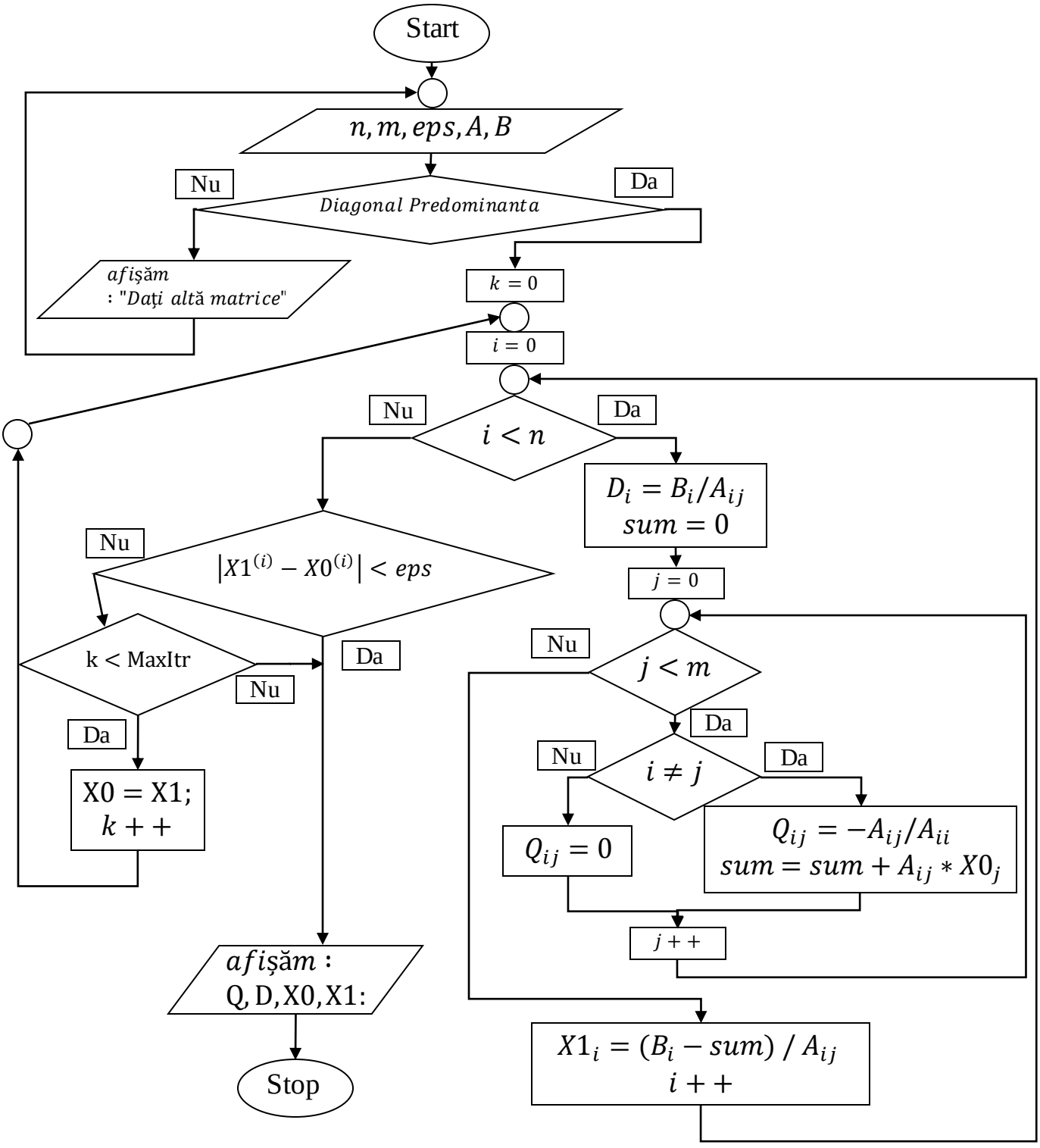
Fie $Ax = b$, și un sistem de n liniar ecuații, ca :

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Atunci soluția se obține iterativ prin formula :

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

2.1 Schema bloc



2.2 Codul Sursă

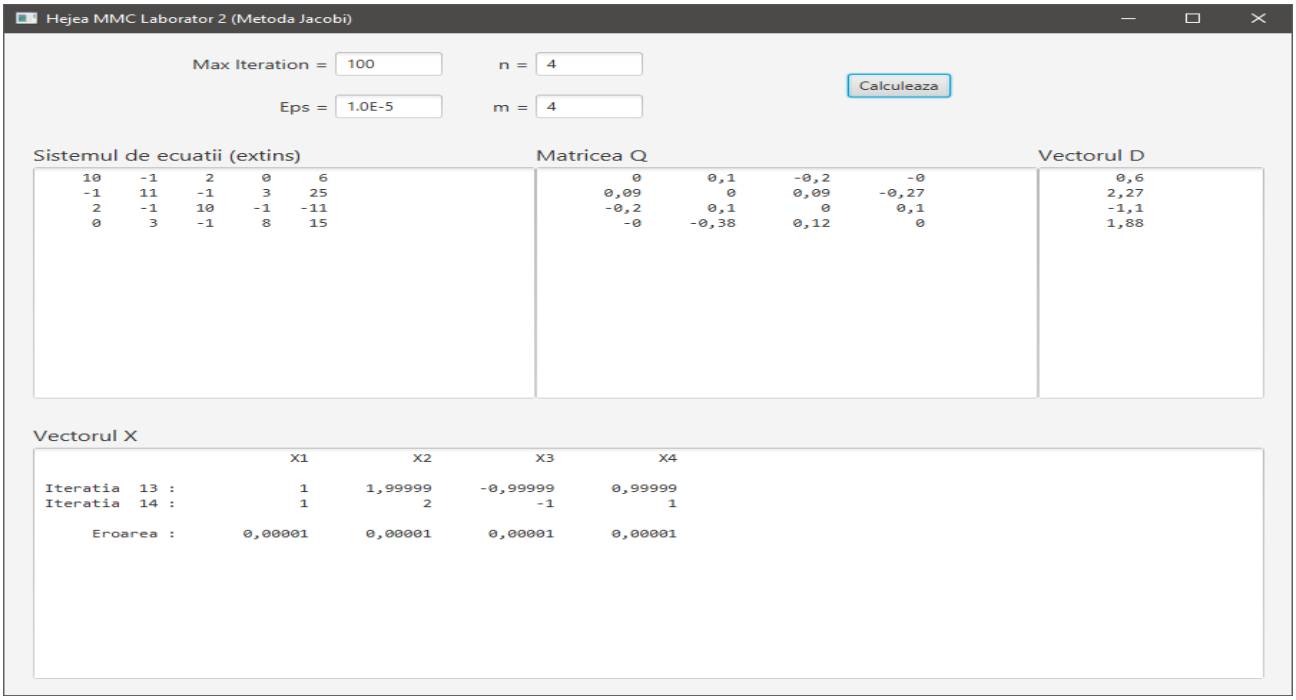
```
public void run() throws Exception {

    sys = new SystemEQ();
    sys.init();

    int    i, j, k;
    double sum;

    if (!Convergenta.diagonalDominant(sys)) {
        throw new Exception("Matricea nu este convergenta");
    } else {

        k = 0;
        do {
            double[] x1 = sys.X.get(k).clone();
            double[] x  = sys.X.get(k).clone();

            for (i = 0; i < sys.Eq.length; i++) {

                sys.D[i] = sys.Eq[i][sys.Eq.length] / sys.Eq[i][i];
                sum = 0.0;

                for (j = 0; j < sys.Eq.length; j++) {
                    if (j != i) {
                        sys.Q[i][j] = -sys.Eq[i][j] / sys.Eq[i][i];
                        sum += sys.Eq[i][j] * x[j];
                    } else {
                        sys.Q[i][j] = 0;
                    }
                }
                x1[i] = sys.Eq[i][sys.Eq.length] - sum;
                x1[i] /= sys.Eq[i][i];
            }

            sys.X.add(x1);
            if (CondStopEps(sys)) break;

            k++;
        } while (k < nrMaxIteration);
    }
}
```

```
public static boolean CondStopEps(SystemEQ sys) {

    int      i;
    boolean  stop = true;
    double[] x1   = sys.X.get(sys.X.size() - 1);
    double[] x0   = sys.X.get(sys.X.size() - 2);
    double   max  = sys.DifE[0] = abs(x1[0] - x0[0]);

    for (i = 1; i < sys.Eq.length; i++) {

        sys.DifE[i] = abs(x1[i] - x0[i]);

        if (max > sys.DifE[i]) {
            max = sys.DifE[i];
        }
    }

    if (max > eps) {
        stop = false;
    }

    return stop;
}
```

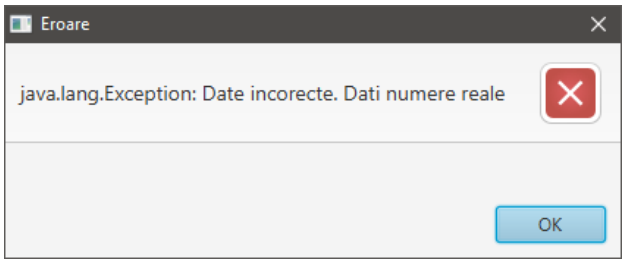
```
public static boolean diagonalDominant(SystemEQ sys) {

    int i, j;
    boolean converge = true;
    double aux;

    for (i = 0; i < sys.Eq.length; i++) {

        aux = 0.0;

        for (j = 0; j < sys.Eq.length; j++) {
            if (i != j) {
                aux += abs(sys.Eq[i][j]);
            }
        }
        if (abs(sys.Eq[i][i]) <= aux) {
            converge = false;
            break;
        }
    }

    return converge;
}
```

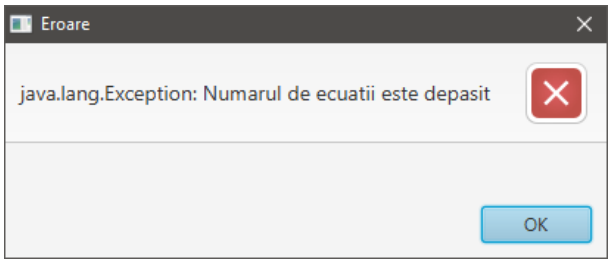
2.3 *Rezultate*

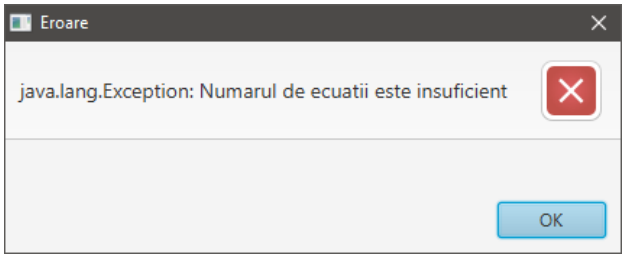
Când are soluții :

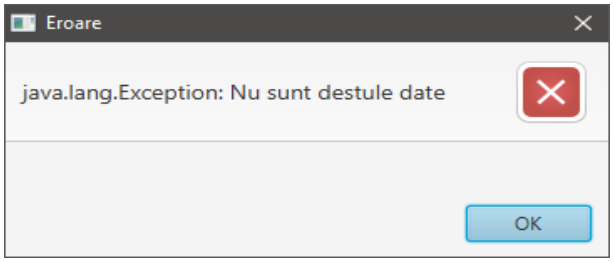


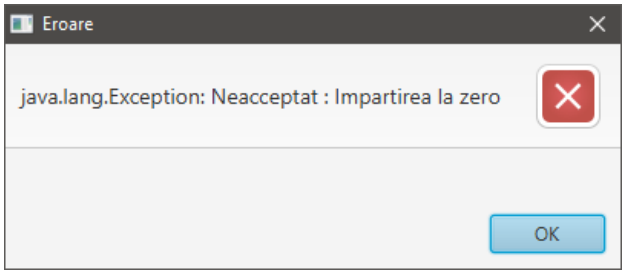
Citeva erori care pot să apară :

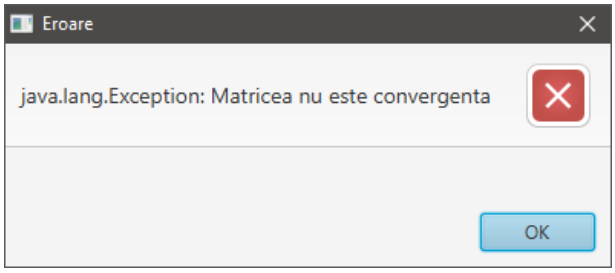












3. Concluzia

Metoda Jacobi este un algoritm iterativ pentru determinarea soluțiilor intr-un sistem de ecuații liniare diagonal dominant. Calculează elementele de pe diagonală, de la o aproximație inițială. Iterează atâta timp cât converge sau nu depășește o eroare dată.

Această metoda converge destul de repede, dar mai poate fi înbinătățit prin :

În aceiași iterație începând cu x_2 ... x_n , pentru calcularea lor se vor folosi x_1 ... x_{n-1} din pasul curet și nu din precedent. Atunci această metodă converge extrem de rapid.