Ministerul Educaţiei, Culturii si Cercetarii al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică şi Microelectronică
Departamentul Ingineria Software şi Automatică

Disciplina: Metode si Modele de Calcul
Modulul: Metode numerice

Raport

Lucrarea de laborator nr.2

Tema: Rezolvarea numerică a sistemelor de ecuaţii liniare

A efectuat: studentul Culea Constantin, gr. TI-183
A controlat: prof.univ., dr. Ştefan Buzurniuc

Chişinău 2019

Sarcina lucrării: (Varianta 15)

Sa se rezolve sistemul de ecuatii:

a) Prin metoda Gauss;
b) Prin metoda Iacobi;
c) Prin metoda Gauss-Seidel.

N = 15;

X1	X2	X3	X4	B
3.4 + 0.5 * N	0.7 + 0.1 * N	0.2 + 0.2 * N	-0.2	5.1 + N
0.7 + 0.2 * N	5.2 + 0.1 * N	0.3 + 0.1 * N	0.5	4.2 + 12 * N
0.2 + 0.1 * N	0.3 + 0.2 * N	3.8 + 0.7 * N	-0.4	5.3 - 0.1 * N
-0.2 - 0.1 * N	0.5 + 0.1 * N	-(0.4 + 0.1 * N)	4.7 + N	5.4 + N

Consideraţii teoretice:

Metodele numerice de rezolvare a sistemelor algebrice de ecuaţii liniare sunt de două tipuri: metode directe şi metode indirecte (sau iterative).

Metodele directe constau în transformarea sistemului într–un sistem triunghiular echivalent, care se rezolvă uşor. Cele mai cunoscute metode directe sunt: metoda Gauss, metoda Gauss-Jordan, metoda Cholesky (utilizată pentru sistemele în care matricea A este simetrică şi pozitiv definită) şi metoda Householder. Metodele directe permit determinarea soluţiei exacte a sistemului în cazul ideal, când nu avem erori de rotunjire. Numărul operaţiilor aritmetice efectuate este de ordinul n3.(În programare, pentru sisteme cu un număr de ecuaţii mai mare de 100, metodele directe devin inutilizabile datorită acumulării erorilor de rotunjire care alterează soluţia.)

Metodele indirecte (sau iterative) constau în construcţia unui şir {x(k)} de vectori n–dimensionali, care converge la soluţia exactă a sistemului. Se alege ca soluţie aproximativă a sistemului un termen x(s) al şirului, al cărui ordin depinde de precizia impusă.

O iteraţie presupune efectuarea unui număr de operaţii aritmetice de ordinul n2. Metodele iterative sunt utilizate la rezolvarea sistemelor mari de ecuaţii. Cele mai cunoscute metode iterative sunt: Jacobi, Gauss–Seidel, metodele de relaxare.

Metoda lui Gauss :

Metoda lui Gauss este una din metodele tradiţionale directe de rezolvare a sistemelor de ecuaţii liniare. Ideea de bază a metodei constă în aducerea sistemului de ecuaţii prin transformări elementare la o formă echivalentă, având matrice superior sau inferior triunghiulară, urmată de rezolvarea sistemului rezultat prin procedee recurente specifice, foarte eficiente (substituirea succesivă, în sens invers, a necunoscutelor, obţinându-se astfel vectorul soluţiei.).

Transformarea sistemului iniţial într-un sistem de formă triunghiulară se realizează cu ajutorul a trei operaţii elementare:

a) interschimbarea a două ecuaţii între ele;

b) înmulţirea unei ecuaţii cu o constantă nenulă;

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

Transformarea sistemului este echivalentă cu eliminarea succesivă a necunoscutelor din ecuaţii şi se numeşte de aceea ***faza eliminării***.

Rezolvarea sistemului cu matrice triunghiulară constă în determinarea necunoscutelor şi substituţia lor în ecuaţiile sistemului în ordine inversă, fiind denumită din acest motiv ***faza substituţiei inverse***.

Fie pentru exemplificare un sistem de trei ecuaţii cu trei necunoscute:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \qquad (1)$$

sau sub forma matriceală

respectiv: $A \cdot x = b$

cu A=[a ij] – matricea sistemului,

x=[xi] – matricea coloană a necunoscutelor

b=[bi] – matricea coloană a termenilor liberi.

Codul programului:

```
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include <iomanip>
#define N 15

using namespace std;

//sistemul cercetat
float A[4][5] = {
        {(3.4 + 0.5 * N), (0.7 + 0.1 * N), (0.2 + 0.2 * N), -0.2, (5.1 + N)},
```

```
        {(0.7 + 0.2 * N), (5.2 + 0.1 * N), (0.3 + 0.1 * N), 0.5, (4.2 + 12 * N)},
        {(0.2 + 0.1 * N), (0.3 + 0.2 * N), (3.8 + 0.7 * N), -0.4, (5.3 - 0.1 * N)},
        {(-0.2 - 0.1 * N), (0.5 + 0.1 * N), -(0.4 + 0.1 * N), (4.7 + N), (5.4 + N)}
};

int iter = 0;

//functie pentru verificarea zerourilor pe diagonala
bool checkZeroDiagonal(bool AUX[], int linii, int coloane) {
    int k = 0;
    for (int i = 0; i < linii; i++) {
        for (int j = 0; j < coloane; j++) {
            if (i == j) {
                if (A[i][i] == 0) {
                    AUX[i] = true;
                    k++;
                } else {
                    AUX[i] = false;
                }
            }
        }
    }
    if (k > 0) {
        return true;
    } else {
        return false;
    }
}

//functie pentru schimbarea liniilor cu locul pentru evitarea zerourilor pe diagonala
void changeRows(bool AUX[], int linii, int coloane) {
    float auxiliar;
    for (int i = 0; i < linii; i++) {
```

```
                if (AUX[i] == true) { //schimbam linia ce contine zerou pe diagonala
                        for (int j = 0; j < coloane; j++) {
                                if (i == linii - 1) { //daca trebuie schimbata ultima linie
                                        auxiliar = A[i - linii + 1][j];
                                        A[i - linii + 1][j] = A[i][j];
                                        A[i][j] = auxiliar;
                                } else { //daca trebuie schimbate celelalte linii
                                        auxiliar = A[i + 1][j];
                                        A[i + 1][j] = A[i][j];
                                        A[i][j] = auxiliar;
                                }
                        }
                }
        }
}

//implementarea metodei Gauss
void Metoda_Gauss(int linii, int coloane)
{
        int i, j, k;
        float pivot;
        bool AUX[4];

        //interschimbam liniile daca exista zerouri pe diagonala
        while (checkZeroDiagonal(AUX, linii, coloane) == true) {
                checkZeroDiagonal(AUX, linii, coloane);
        }

    cout << "Matricea cercetata:" << endl << endl;
        for (i = 0; i < 4; i++) {
                for (j = 0; j < 5; j++) {
                        cout << A[i][j] << "   ";
                }
```

```
        cout << endl;
}

for (i = 0; i <= linii - 1; i++) {
        pivot = A[i][i];

        if (pivot != 0) {

                cout << endl;

                for (k = i + 1; k < linii; k++) { //calculam elementele urmatoarelor linii
                        for (j = i + 1; j < coloane; j++) {
                                A[k][j] = ((A[k][j] * pivot) - (A[k][i] * A[i][j])) / pivot;
                        }

                }

                for (j = 0; j < coloane; j++) { //impartirea liniei la elementul pivot
                        A[i][j] = A[i][j] / pivot;
                }

                cout << endl;

                for (k = i; k < linii - 1; k++) {
                        A[k + 1][i] = 0;
                }

        }

        cout << endl;

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

```
                        if (A[i][j] == -0) {
                                cout << setprecision(3) << fabs(A[i][j]) << "\t";
                        }
                        else {
                                cout << setprecision(3) << A[i][j] << "\t";
                        }
                }
                cout << endl;
        }
    }
    float* p;
    p = new float[linii]();

    float x; float aux1;
    //solution calculation
    for (int i = linii - 1; i >= 0; i--) {
        x = 0;
        aux1 = A[i][coloane - 1];
        for (int j = coloane - 2; j >= 0; j--) {
            x = x + A[i][j] * p[j];
            iter++;
        }
        p[i] = aux1 - x;
    }

    cout << endl;

    for (int g = 0; g < linii; g++) {
        cout << "x" << g + 1 << " = " << p[g] << "\t";
    }
}

//implementarea metodei Iacobi
```

```
void Metoda_Iacobi_GaussSeidel(int linii, int coloane, char c = 'i')
{
	int i, j, k, max_i, max_j;
	float aux, max;
	float* sol = new float[linii]();
	float* aux_sol = new float[coloane]();

	bool solFound = false;
	//verificam daca matricea e dominant diagonala si schimbam liniile daca e cazul
	for (i = 0; i < linii; i++) {
		max = A[i][0];
		for (j = 0; j < linii; j++) {
			if (A[i][j] > max) {
				max = A[i][j];
				max_i = i;
				max_j = j;
			}
		}

		if (A[i][i] != max) {
			for (k = 0; k < coloane; k++) {
				aux = A[max_j][k];
				A[max_j][k] = A[i][k];
				A[i][k] = aux;
				cout << endl;
			}
			cout << endl << endl;
		}
	}
   cout << "Matricea dominant diagonala:" << endl << endl;
	for (i = 0; i < linii; i++) {
		for (j = 0; j < coloane; j++) {
			cout << A[i][j] << "\t";
```

```
        }
        cout << endl << endl;
}

do {
        for (i = 0; i < linii; i++) {
                aux_sol[i] = A[i][coloane - 1];
                cout << "x= " << aux_sol[i];
                for (j = 0; j < linii; j++) {
                        if (i != j) {
                                if (c == 'g') {
                                        aux_sol[i] = aux_sol[i] - (A[i][j] * aux_sol[j]);
                                }
                                else {
                                        aux_sol[i] = aux_sol[i] - (A[i][j] * sol[j]);
                                }
                                iter++;
                        }
                }
                aux_sol[i] = aux_sol[i] / A[i][i];
                cout << endl;
        }
        cout << endl << endl;

        for (i = 0; i < linii; i++) {
                if (fabs(aux_sol[i] - sol[i]) <= 0.00001) {
                        solFound = true;
                        break;
                }
                else {
                        sol[i] = aux_sol[i];
                        solFound = false;
                }
```

```
            }
        } while (!solFound);

        for (i = 0; i < linii; i++) {
            cout << "x" << i + 1 << " = " << sol[i] << "\t";
        }
}

int main()
{
        int linii = 4, coloane = 5;

        //apelarea metodei Gauss
        Metoda_Gauss(linii, coloane);

        //apelarea metodei Iacobi so Gauss-Seidel
        //pentru metoda Iacobi al treilea parametru nu este necesar sau se indica 'i'
        //pentru metoda Gauss-Seidel al treilea parametru se indica 'g'
        Metoda_Iacobi_GaussSeidel(linii, coloane, 'g');

        //afisarea numarului de ietarii
        cout << endl << "Iteratii: " << iter << endl;

        return 0;
}
```

Rezultatele obtinute:

Metoda Gauss:

```
Matricea cercetata:

10.9   2.2   3.2   -0.2   20.1
3.7   6.7   1.8   0.5   184.2
1.7   3.3   14.3   -0.4   3.8
-1.7   2   -1.9   19.7   20.4
```

Matricea rezultata si solutiile respective:

```
1       0.202   0.294   -0.0183 1.84
0       1       0.12    0.0954  29.8
0       0       1       -0.0484 -6.5
0       0       0       1       -2.95

x1 = -2.49      x2 = 30.9       x3 = -6.65      x4 = -2.95
Iteratii: 16
```

Metoda Iacobi si Gauss-Seidel:

```
Matricea dominant diagonala:

10.9    2.2     3.2     -0.2    20.1

3.7     6.7     1.8     0.5     184.2

1.7     3.3     14.3    -0.4    3.8

-1.7    2       -1.9    19.7    20.4
```

Solutiile Metoda Iacobi:

```
x1 = -2.49068   x2 = 30.8737    x3 = -6.64562   x4 = -2.95474
Iteratii: 216
```

Solutiile Metoda Gauss-Seidel:

```
x1 = -2.4906    x2 = 30.8738    x3 = -6.64555   x4 = -2.95472
Iteratii: 96
```

Concluzie:

În decursul efectuării lucrării de laborator am studiat rezolvarea sistemelor de ecuații liniare. Pentru aceasta s-au folosit unele din cele mai populare metode iterative, cum ar fi metoda Gauss, Iacobi și Gauss-Seidel. Fiecare din aceste metode se caracterizeaza prin metoda sa specifica de rezolvare și nivelul de complexitate. Astfel, metoda Gauss, se bazează pe depistarea elementului pivot și transformarea matricei în unitar diagonală. În acest mod se obține foarte simplificat soluțiile, datorita faptului că sub diagonala unitară toate elementele iau valoarea zero. Metoda Iacobi și Gauss-Seidel au trasaturi foarte comune prin faptul ca în metoda a doua se i-a în considerație soluția obținută la pasul precedent. În acest fel se ajunge foarte repede la soluția cercetată cu o anumită exactitate. O trasătură importantă al acestor două metode este că înainte de calcule sistemul de ecuații trebuie neapărat să fie dominant diagonală. Adică, liniile trebuie sa fie aranjate în așa mod încât să aibă coeficientul maxim pe diagonală.