**MINISTERUL EDUCAȚIEI ȘI CERCETĂRII AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**RAPORT**

Lucrare de laborator nr. 2

la cursul ***„Metode numerice”***

**Varianta 13**

**A efectuat :**   **St. gr. CR-221FR Serba Cristina**

**A verificat: Seiciuc Veaceslav**

**Chișinău 2023**

# Sarcina 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 e=10-3;

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

2) Să se determine numărul de iteraţii necesare pentru aproximarea soluţiei

sistemului cu eroarea dată e. Să se compare rezultatele.

# Mersul programului:

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#define JACOBI\_EROARE 0.01

#define GAUSS\_SEIDEL\_EROARE1 0.01

#define GAUSS\_SEIDEL\_EROARE2 0.0001

double A[4][4] = { {8.7, -1.2, 0.8, 0.7},

{-1.2, 9.6, -1.2, 0.8},

{0.8, -1.2, 8.8, 0.9},

{0.7, 0.8, 0.9, 11.3}};

double B[4] = {-2.7, 8.9, 7.2, 6.4};

void gauss(int N, double a[4][4], double b[4]);

void choelsky(double A[4][4], double B[4]);

void jacobi(double a[4][4], double b[4]);

void gauss\_seidel(double a[4][4], double b[4]);

int main(void)

{

int choice;

do{

printf("Meniu: \n1. Metoda eliminarii lui Gauss\n2. Metoda lui Cholesky\n3. Metoda iterativă a lui Jacobi cu o eroare e=10-3\n4. Metoda iterativă a lui Gauss-Seidel cu o eroare e=10-3 şi e=10-5\n0. Iesire din program\n");

printf("Alegerea: ");

scanf("%d", &choice);

switch(choice)

{

case 0:

{

exit(0);

}

case 1:

{

gauss(4, A, B);

break;

}

case 2:

{

choelsky(A, B);

break;

}

case 3:

{

jacobi(A, B);

break;

}

case 4:

{

gauss\_seidel(A, B);

break;

}

default:

{

printf("Alegere invalida\n");

break;

}

}

} while(choice != 0);

return 0;

}

void choelsky(double A[4][4], double B[4])

{

double s;

int n = 4;

double matrice[4][4], y[4], x[4];

//calculam matricea Lij

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

{

for (int j = 0; j < (i + 1); j++)

{

s = 0;

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

{

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

}

matrice[i][j] = (i == j) ? sqrt(A[i][i] - s) : (1.0 / matrice[j][j] \* (A[i][j] - s));

}

}

//calculam suma Lik \* Dk

//apoi Di

y[0] = B[0]/matrice[0][0];

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

{

s = 0;

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

s += matrice[i][k] \* y[k];

y[i] = (1 / matrice[i][i]\* (B[i] - s));

}

//calculam suma tuturor Lik \* Ljk \* Dk

// apoi Lij

// s = 0;

x[n - 1] = y[n - 1] / matrice[n-1][n-1];

for (int i = n - 2; i >= 0; i--)

{

s = 0;

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

s += (matrice[i][k] \* x[k]);

x[i] = (1 / matrice[i][i] \* (y[i] - s));

}

printf("Solutia prin metoda Cholesky\n");

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

{

printf("x%d = %.4f\n", i + 1, x[i]);

}

}

void gauss(int n, double a[4][4], double b[4]) {

// Eliminarea in directia pozitiva

for (int k = 0; k < n - 1; k++) {

for (int i = k + 1; i < n; i++) {

double factor = a[i][k] / a[k][k];

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

a[i][j] -= factor \* a[k][j];

}

b[i] -= factor \* b[k];

}

}

// Substitutia in directia negativa

double X[n];

for (int i = n - 1; i >= 0; i--) {

X[i] = b[i] / a[i][i];

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

X[i] -= a[i][j] \* X[j] / a[i][i];

}

}

// Rezultatele obtinute

printf("Solutia prin metoda Gauss:\n");

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

printf("x%d = %.4f\n", i + 1, X[i]);

}

}

void jacobi(double a[4][4], double b[4])

{

int n = 4, itr = 1;

double eroare = 0, sum, x0[4], x[4];

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

{

x0[i] = b[i] / a[i][i]; //vectorul x0 ia elementele de pe diagonala in relatie cu vectorul b

}

do

{

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

{

sum = 0;

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

{

if (i != j)

{

sum = sum + a[i][j] \* x0[j]; //suma elementelor ce nu apartin diagonalei

}

}

x[i] = (b[i] - sum) / a[i][i]; // atribuim valoarea din xprim lui x

eroare = fabs(x[i] - x0[i]);

}

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

{

x0[i] = x[i];

}

itr++;

} while (eroare > JACOBI\_EROARE);

//Afisare rezultate

printf("Metoda Jacobi converge catre solutie in %d iteratii\n", itr); // daca converge afisam x1..xn si iesim din ciclu

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

{

printf("x%d = %.4f\n", i+1, x[i]);

}

}

void gauss\_seidel(double a[4][4], double b[4])

{

double x[4], Meroare, eroare, sum, sum2, Xprim;

int n = 4, iter = 1;

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

{

x[i] = b[i] / a[i][i];

}

do {

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

{

sum2 = 0;

sum = 0;

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

{

if (j != i)

{

sum += a[i][j] \* x[j];

}

}

//suma elementelor ce nu apartin diagonalei

for (int j = 1; j < i - 1; j++)

{

if (j != i)

{

sum2 += a[i][j] \* x[j];

}

}

Xprim = (b[i] - sum - sum) / a[i][i];

eroare = fabs(x[i] - Xprim);

x[i] = Xprim;

}

iter++;

} while (eroare > GAUSS\_SEIDEL\_EROARE2);

//Afisare rezultate

printf("Metoda Gauss Seidel converge in %d iteratii catre solutia:\n", iter);

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

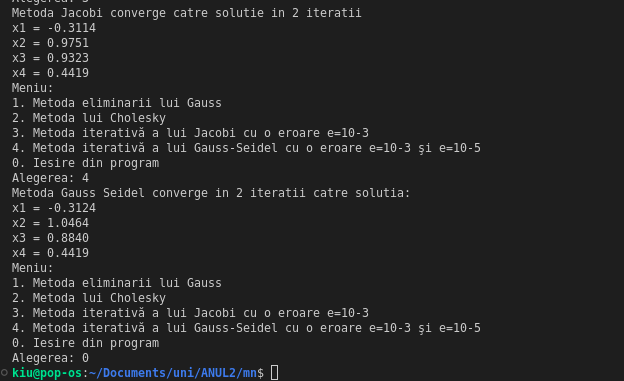
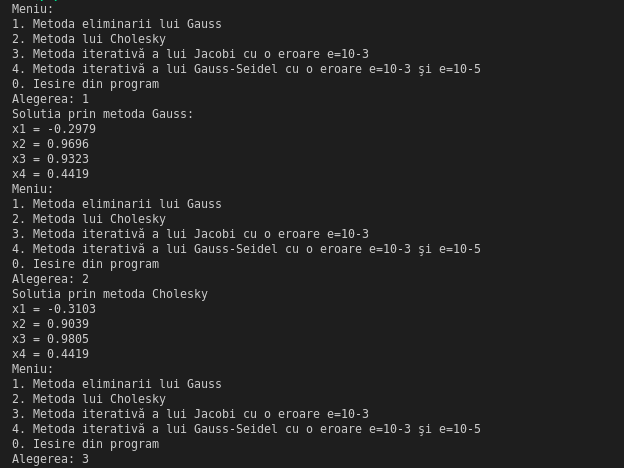
{

printf("x%d = %.4f\n", i+1, x[i]);

}

}

# Rezultatul executiei programului:



# Concluzii:

În concluzie, alegerea metodei rezolvării unei ecuații de forma Ax=B depindeisticile sistemului de ecuații, precum dimensiunea matricei, simetria și definitețea acesteia, precum și de precizia și eficiența dorită. Este important să se analizeze specificul problemei și să se aleagă metoda care se potrivește cel mai bine contextului particular. Metodele iterativie, cum ar fi Jacobi și Gauss-Seidel, pot fi mai potrivite pentru sistemele mari, în timp ce metodele directe, cum ar fi Gauss și Cholesky, pot oferi precizie mai mare în anumite condiții.