

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**gr. IA-231, Chistol Maxim**

**Raport**

**pentru lucrarea de laborator Nr.3**

***la cursul de “Metode Numerice”***

Verificat:

Moraru Vasile**,** asistent.universitar.

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

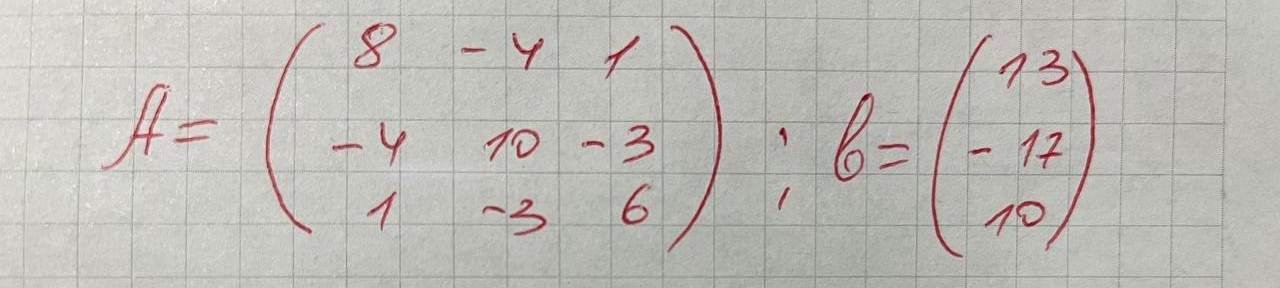
**Chișinău 2024**

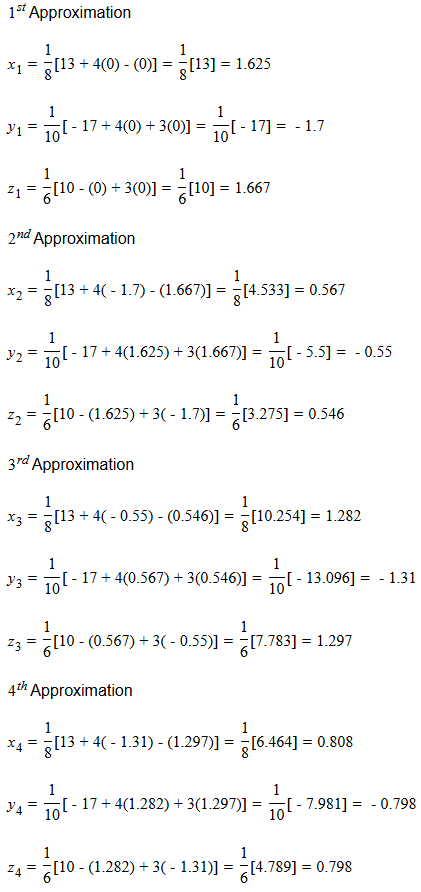
Scopul lucrării:

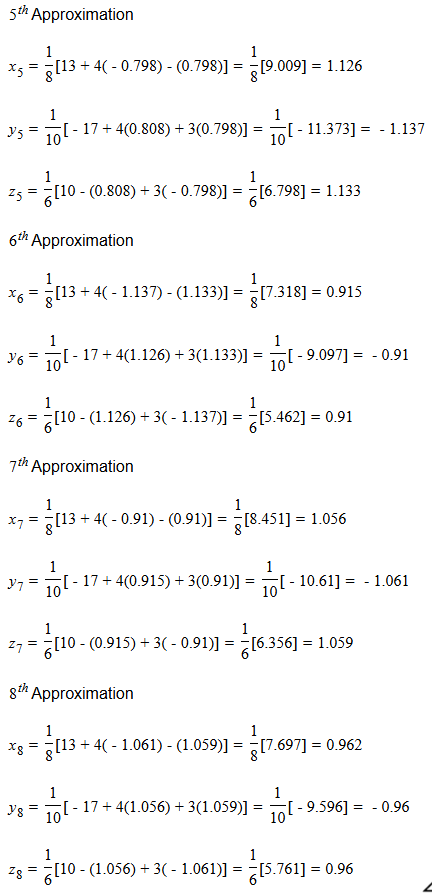
- Metoda iterativă a lui Jacobi cu o eroare ε=10^-3 ;

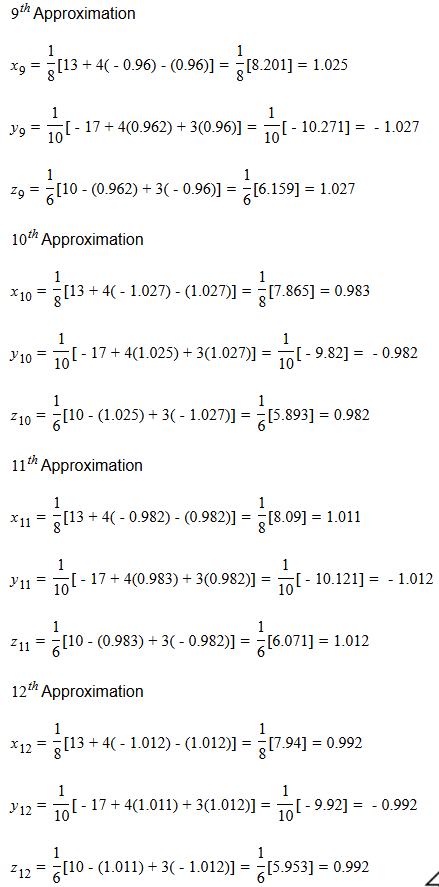
- Metoda iterativă a lui Gauss-Seidel cu o eroare ε=10-3 şi ε=10-5

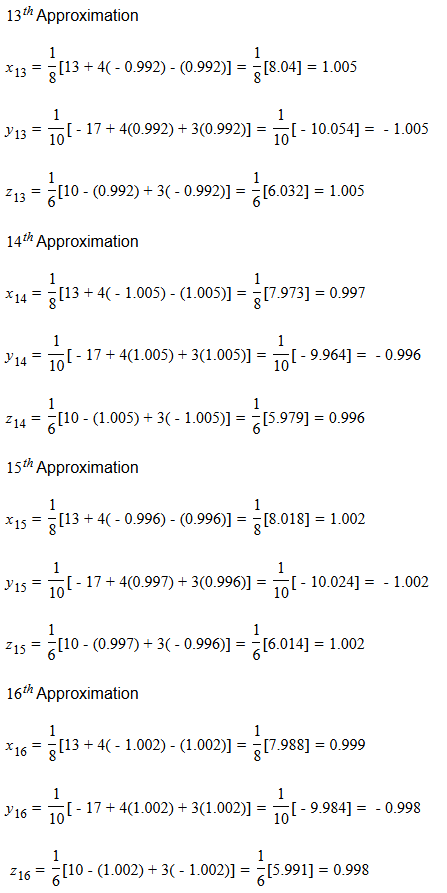
.Calculați descompunerea LU a matricei 𝐴.

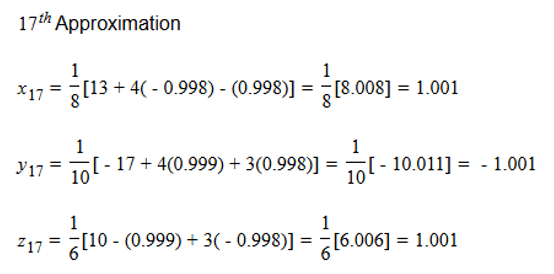


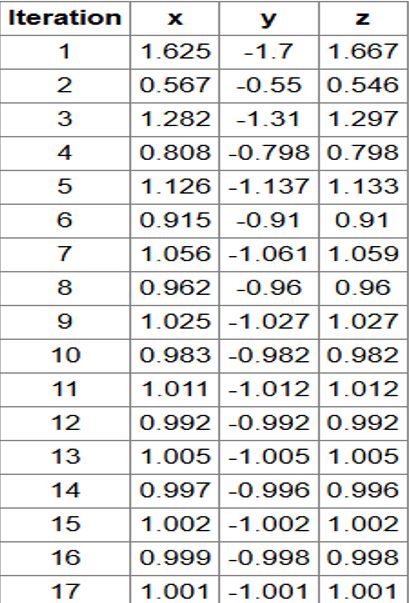




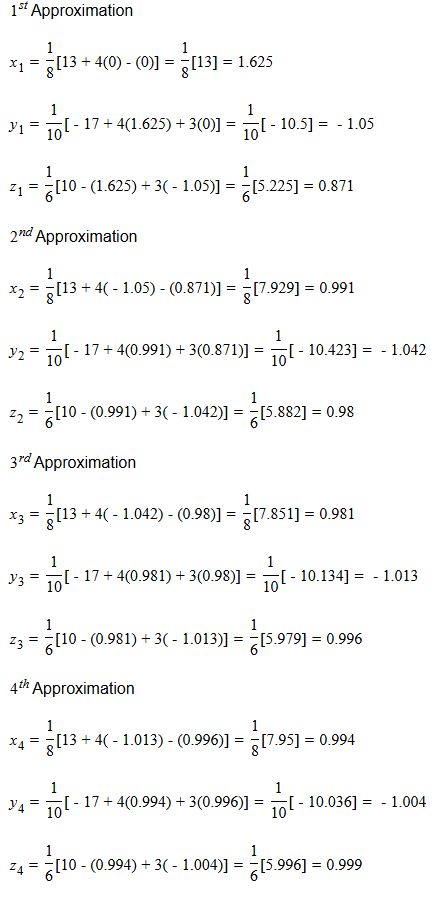


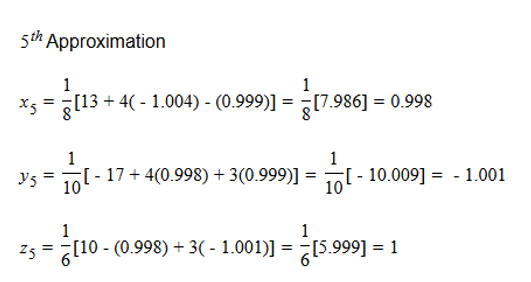


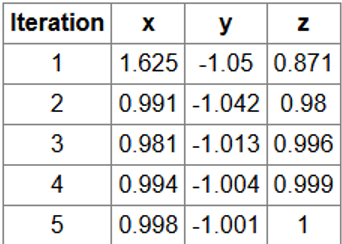




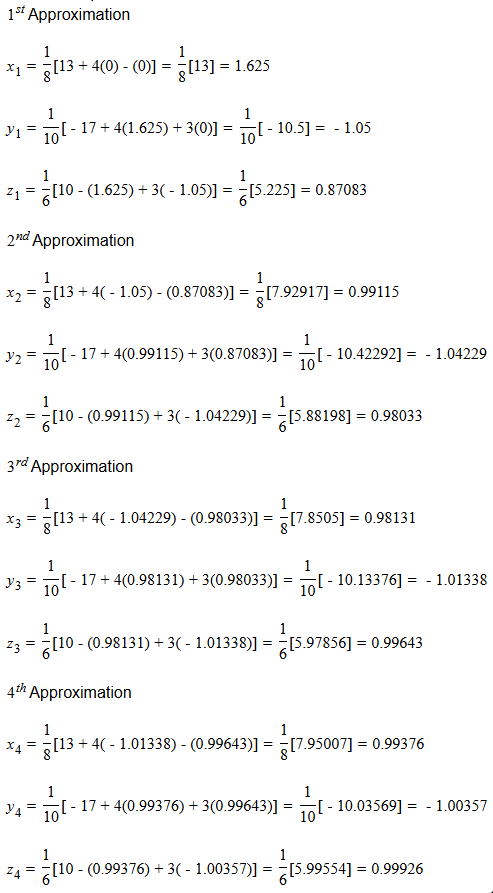
- Metoda iterativă a lui Jacobi cu o eroare ε=10^-3

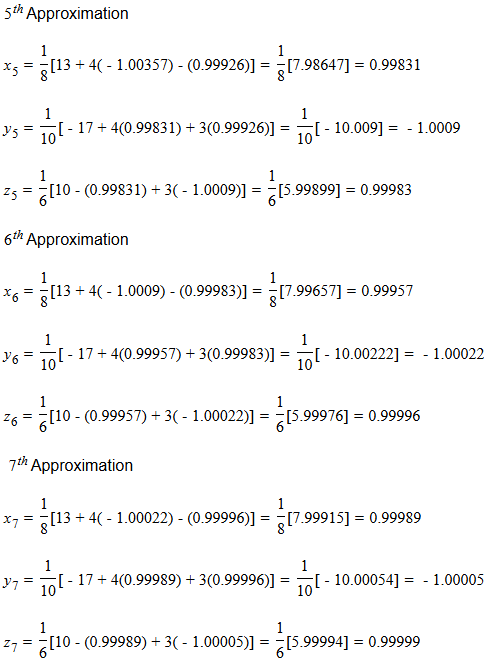


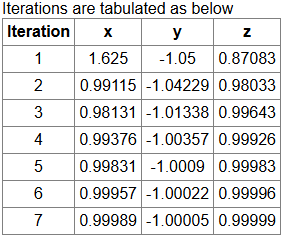




- Metoda iterativă a lui Gauss-Seidel cu o eroare ε=10^-3







- Metoda iterativă a lui Gauss-Seidel cu o eroare ε=10^-5

|  |  |  |  |
| --- | --- | --- | --- |
| Metodele | Metoda lui Jacobi ε=10^-3 | Metoda lui Gauss-Seidel ε=10^-3 | Metoda lui Gauss-Seidel ε=10^-5 |
| Rezultatul | X1=1.001  X2=-1.001  X3=1.001 | X1=0.998,  X2= -1.001  X3=1 | X1=0.99989  ,X2=-1,00005,  X3=0.99999 |

Codul:

Metoda iterativă a lui Jacobi cu o eroare ε=10^-3

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip> // Pentru setarea preciziei

using namespace std;

// Funcție pentru a calcula norma maximă

double calculeazaNorma(const vector<double>& vec1, const vector<double>& vec2, int n) {

double norma = 0.0;

int i = 0;

while (i < n) {

double diferenta = fabs(vec1[i] - vec2[i]);

if (diferenta > norma) {

norma = diferenta;

}

i++;

}

return norma;

}

// Metoda iterativă a lui Jacobi

void metodaJacobi(const vector<vector<double>>& A, const vector<double>& b, double epsilon, int n) {

vector<double> x\_curent(n, 0.0); // Aproximarea curentă

vector<double> x\_urmator(n, 0.0); // Aproximarea următoare

int iteratii = 0;

double eroare = 0.0;

cout << "Iterațiile metodei Jacobi:\n";

do {

int i = 0;

while (i < n) {

double suma = 0.0;

int j = 0;

while (j < n) {

if (i != j) {

suma += A[i][j] \* x\_curent[j];

}

j++;

}

x\_urmator[i] = (b[i] - suma) / A[i][i];

i++;

}

eroare = calculeazaNorma(x\_urmator, x\_curent, n);

int k = 0;

while (k < n) {

x\_curent[k] = x\_urmator[k];

k++;

}

++iteratii;

// Afișare stare curentă

cout << "Iterația " << iteratii << ": ";

int l = 0;

while (l < n) {

cout << fixed << setprecision(2) << x\_curent[l] << " ";

l++;

}

cout << "\n";

} while (eroare > epsilon);

cout << "\nSoluția aproximativă este:\n";

int m = 0;

while (m < n) {

cout << fixed << setprecision(2) << x\_curent[m] << " ";

m++;

}

cout << "\nNumărul de iterații: " << iteratii << "\n";

}

int main() {

int n;

cout << "Introduceți dimensiunea matricei: ";

cin >> n;

vector<vector<double>> A(n, vector<double>(n));

vector<double> b(n);

cout << "Introduceți matricea A (dimensiunea " << n << "x" << n << "):\n";

int i = 0;

while (i < n) {

int j = 0;

while (j < n) {

cin >> A[i][j];

j++;

}

i++;

}

cout << "Introduceți vectorul b (dimensiunea " << n << "):\n";

int k = 0;

while (k < n) {

cin >> b[k];

k++;

}

double epsilon = 0.001; // Eroare

metodaJacobi(A, b, epsilon, n);

return 0;

}

- Metoda iterativă a lui Gauss-Seidel cu o eroare ε=10^-5 și ε=10^-3

#include <iostream>

#include <vector>

#include <cmath>

#include <iomanip> // Pentru setarea preciziei

using namespace std;

// Funcție pentru a calcula norma maximă

double calculeazaNorma(const vector<double>& vec1, const vector<double>& vec2, int n) {

double norma = 0.0;

int i = 0;

while (i < n) {

double diferenta = fabs(vec1[i] - vec2[i]);

if (diferenta > norma) {

norma = diferenta;

}

i++;

}

return norma;

}

// Metoda iterativă a lui Gauss-Seidel

void metodaGaussSeidel(const vector<vector<double>>& A, const vector<double>& b, double epsilon, int n) {

vector<double> x\_curent(n, 0.0); // Aproximarea curentă

vector<double> x\_anterior(n, 0.0); // Aproximarea anterioară

int iteratii = 0;

double eroare = 0.0;

// Setarea preciziei în funcție de eroare

int precizie = (epsilon == 0.001) ? 2 : 5;

cout << "Iterațiile metodei Gauss-Seidel:\n";

do {

int i = 0;

while (i < n) {

double suma1 = 0.0, suma2 = 0.0;

int j = 0;

while (j < i) { // Partea de sub diagonală

suma1 += A[i][j] \* x\_curent[j];

j++;

}

while (j < n) { // Partea de deasupra diagonală

if (i != j) {

suma2 += A[i][j] \* x\_anterior[j];

}

j++;

}

x\_curent[i] = (b[i] - suma1 - suma2) / A[i][i];

i++;

}

eroare = calculeazaNorma(x\_curent, x\_anterior, n);

int k = 0;

while (k < n) {

x\_anterior[k] = x\_curent[k];

k++;

}

++iteratii;

// Afișare stare curentă

cout << "Iterația " << iteratii << ": ";

int l = 0;

while (l < n) {

cout << fixed << setprecision(precizie) << x\_curent[l] << " ";

l++;

}

cout << "\n";

} while (eroare > epsilon);

cout << "\nSoluția aproximativă este:\n";

int m = 0;

while (m < n) {

cout << fixed << setprecision(precizie) << x\_curent[m] << " ";

m++;

}

cout << "\nNumărul de iterații: " << iteratii << "\n";

}

int main() {

int n;

cout << "Introduceți dimensiunea matricei: ";

cin >> n;

vector<vector<double>> A(n, vector<double>(n));

vector<double> b(n);

cout << "Introduceți matricea A (dimensiunea " << n << "x" << n << "):\n";

int i = 0;

while (i < n) {

int j = 0;

while (j < n) {

cin >> A[i][j];

j++;

}

i++;

}

cout << "Introduceți vectorul b (dimensiunea " << n << "):\n";

int k = 0;

while (k < n) {

cin >> b[k];

k++;

}

double epsilon;

cout << "Introduceți valoarea erorii ε (de exemplu 0.00001 sau 0.001): ";

cin >> epsilon;

metodaGaussSeidel(A, b, epsilon, n);

return 0;

}

Figura 3 Outputul programului

# Concluzii:

Metodele iterative Jacobi și Gauss-Seidel sunt instrumente eficiente pentru rezolvarea sistemelor de ecuații liniare mari, mai ales atunci când metodele directe devin costisitoare.

Metoda Gauss-Seidel este mai rapidă decât metoda Jacobi, datorită utilizării valorilor actualizate în timpul calculului. Totuși, ambele metode necesită ca matricea sistemului să fie strict diagonal dominantă sau pozitiv definită pentru a garanta convergența.

Precizia soluției depinde de valoarea erorii tolerate (ε):

- O valoare mai mică de (ε=10^-5) asigură o soluție mai exactă, dar implică un număr mai mare de iterații.

- Pentru aplicații practice, (ε = 10^-3) este adesea suficient, oferind un echilibru între precizie și eficiență.

Metoda Gauss-Seidel este preferată atunci când timpul de calcul este o constrângere, însă ambele metode rămân relevante în funcție de proprietățile sistemului și cerințele de precizie.