

Ministerul Educației, Culturii și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul de Informatică și Inginerie a Sistemelor



RAPORT

asupra lucrării de laborator nr. 2

la disciplina *Metode și modele de calcul*

A elaborat:

st.gr.MI-191, Boj Tatiana

A verificat:

Conf.univ, dr. Moraru Vasile

Chișinău 2020

Scopul lucrării:

1. Să se rezolve sistemul de ecuații lineare $Ax=b$, utilizând
 - Metoda eliminării lui Gauss;
 - Metoda lui Cholesky (metoda rădăcinii pătrate);
 - Metoda iterativă a lui Jacobi cu o eroare $\varepsilon=10^{-3}$;
 - Metoda iterativă a lui Gauss-Seidel cu o eroare $\varepsilon=10^{-3}$ și $\varepsilon=10^{-5}$.

2. Să se determine numărul de iterații necesare pentru aproximarea soluției sistemului cu eroarea dată ε . Să se compare rezultatele.

Problema:

$$A = \begin{pmatrix} 18 & -7 & 9 \\ -7 & 20 & 9 \\ 9 & -3 & 13 \end{pmatrix} \quad b = \begin{pmatrix} 9 \\ -4 \\ -4 \end{pmatrix}$$

- Metoda eliminarii Gauss

Codul sursa:

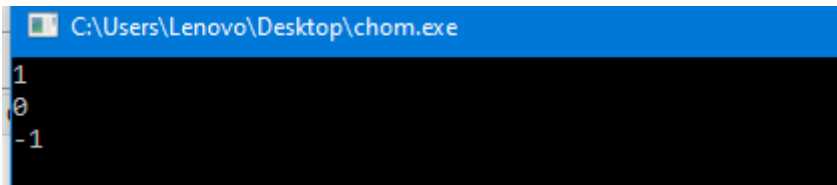
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define mat_elem(a, y, x, n) (a + ((y) * (n) + (x)))
void swap_row(double *a, double *b, int r1, int r2, int n)
{
    double tmp, *p1, *p2;
    int i;
    if (r1 == r2) return;
    for (i = 0; i < n; i++) {
        p1 = mat_elem(a, r1, i, n);
        p2 = mat_elem(a, r2, i, n);
        tmp = *p1, *p1 = *p2, *p2 = tmp;
    }
    tmp = b[r1], b[r1] = b[r2], b[r2] = tmp;
}
void gauss_eliminate(double *a, double *b, double *x, int n)
{
#define A(y, x) (*mat_elem(a, y, x, n))
    int i, j, col, row, max_row, dia;
    double max, tmp;
    for (dia = 0; dia < n; dia++) {
        max_row = dia, max = A(dia, dia);
        for (row = dia + 1; row < n; row++)
            if ((tmp = fabs(A(row, dia))) > max)
                max_row = row, max = tmp;
        swap_row(a, b, dia, max_row, n);
        for (row = dia + 1; row < n; row++) {
            tmp = A(row, dia) / A(dia, dia);
            for (col = dia + 1; col < n; col++)
                A(row, col) -= tmp * A(dia, col);
            A(row, dia) = 0;
            b[row] -= tmp * b[dia];
        }
    }
    for (row = n - 1; row >= 0; row--) {
        tmp = b[row];
        for (j = n - 1; j > row; j--)
            tmp -= x[j] * A(row, j);
        x[row] = tmp / A(row, row);
    }
#undef A
}
int main(void)
{
    double a[] = {
        18.00, -7.00, 9.00,
        -7.00, 20.00, -3.00,
        9.00, -3.00, 13.00
    }
```

```

};
double b[] = { 9.00, -4.00, -4.00 };
double x[3];
int i;
gauss_eliminate(a, b, x, 3);
for (i = 0; i < 3; i++)
    printf("%g\n", x[i]);
return 0;
}

```

Rezultat:



```

C:\Users\Lenovo\Desktop\chom.exe
1
0
-1

```

- Metoda lui Cholesky

Codul sursa:

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
double *cholesky(double *A, int n) {
    int i, j, k;
    double *L = (double *)calloc(n * n, sizeof(double));
    if (L == NULL)
        exit(EXIT_FAILURE);
    for (i = 0; i < n; i++)
        for (j = 0; j < (i+1); j++) {
            double s = 0;
            for (k = 0; k < j; k++)
                s += L[i * n + k] * L[j * n + k];
            L[i * n + j] = (i == j) ?
                sqrt(A[i * n + i] - s) :
                (1.0 / L[j * n + j] * (A[i * n + j] - s));
        }
}

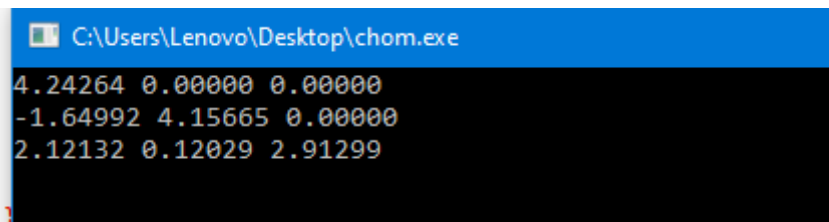
```

```

return L;
}
void show_matrix(double *A, int n) {
    int i, j;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
            printf("%2.5f ", A[i * n + j]);
        printf("\n");
    }
}
int main() {
    int n = 3;
    double m1[] = {
        18, -7, 9,
        -7, 20, -3,
        9, -3, 13};
    double *c1 = cholesky(m1, n);
    show_matrix(c1, n);
    printf("\n");
    free(c1);
    return 0;
}

```

Rezultat:



```

C:\Users\Lenovo\Desktop\chom.exe
4.24264 0.00000 0.00000
-1.64992 4.15665 0.00000
2.12132 0.12029 2.91299

```

- Metoda iterativă a lui Jacobi cu o eroare $\epsilon=10^{-3}$

Codul sursa:

```
#include <cmath>
#include <iostream>
using namespace std;
const double eps = 0.001;
int k_1, k_2;
void Jacobi(int n, double A[3][3], double B[3], double Rez_1[3])
{
    double* temp = new double[n];
    double norm;
    do {
        k_1++;
        for (int i = 0; i<n; i++)
        {
            temp[i] = B[i];
            for (int j = 0; j<n; j++)
            {
                if(i != j)
                    temp[i] -= A[i][j] * Rez_1[j];
            }
            temp[i] /= A[i][i];
        }
        norm = fabs(Rez_1[0] - temp[0]);
        for(int h = 0; h<n; h++)
        {
            if(fabs(Rez_1[h] - temp[h])>norm)
                norm = fabs(Rez_1[h] - temp[h]);
            Rez_1[h] = temp[h];
        }
    } while (norm > eps);
    delete[] temp;
}
bool converge(double *xk, double *xkp)
{
    double norm = 0;
```

```

    for (int i = 0; i<3; i++)
    {
        norm += (xk[i] - xkp[i])*(xk[i] - xkp[i]);
    }
    if(sqrt(norm) >= eps)
        return false;
    return true;
}
void Seidel(int n, double A[3][3], double B[3], double x[3],double p[3])
{
    do
    {
        k_2++;
        for (int i = 0; i<n; i++)
            p[i] = x[i];
        for(int i = 0; i<n; i++)
        {
            double var = 0;
            for(int j = 0; j<i; j++)
                var += (A[i][j] * x[j]);
            for(int j = i + 1; j<n; j++)
                var += (A[i][j] * p[j]);
            x[i] = (B[i] - var) / A[i][i];
        }
    } while(!converge(x, p));
}
int main ()
{
    double A[3][3];
    double B[3];
    double Rez_1[3];
    double Rez_2[3];
    double p[3];

    int n = 3;

    cout<<"Dati datele matricei A: "<<endl;
    for (int i = 0; i<3; i++)
    {
        for (int j = 0; j<3; j++)
        {
            cin>>A[i][j];
        }
    }

```

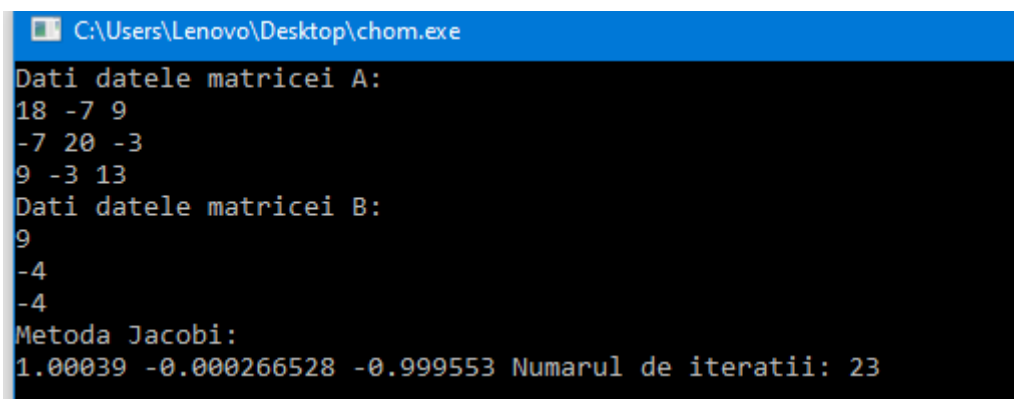
```

    }
    cout<<"Dati datele matricei B: "<<endl;
    for (int i = 0; i<n; i++)
    {
        cin>>B[i];
    }

    for (int i = 0; i<n; i++)
    {
        Rez_1[i] = 0;
        Rez_2[i] = 0;
        p[i] = 0;
    }
    cout<<"Metoda Jacobi: "<<endl;
    Jacobi(n, A, B, Rez_1);
    for (int i = 0; i<n; i++)
    {
        cout<<Rez_1[i]<<" ";
    }
    cout<<"Numarul de iteratii: "<<k_2<<endl;
    return 0;
}

```

Rezultat:



```

C:\Users\Lenovo\Desktop\chom.exe
Dati datele matricei A:
18 -7 9
-7 20 -3
9 -3 13
Dati datele matricei B:
9
-4
-4
Metoda Jacobi:
1.00039 -0.000266528 -0.999553 Numarul de iteratii: 23

```

- Metoda iterativă a lui Gauss-Seidel

Codul sursa:

```
#include <cmath>
#include <iostream>
using namespace std;
const double eps = 0.001;
int k_1, k_2;
void Jacobi(int n, double A[3][3], double B[3], double Rez_1[3])
{
    double* temp = new double[n];
    double norm;
    do {
        k_1++;
        for (int i = 0; i < n; i++)
        {
            temp[i] = B[i];
            for (int j = 0; j < n; j++)
            {
                if (i != j)
                    temp[i] -= A[i][j] * Rez_1[j];
            }
            temp[i] /= A[i][i];
        }
        norm = fabs(Rez_1[0] - temp[0]);
        for (int h = 0; h < n; h++)
        {
            if (fabs(Rez_1[h] - temp[h]) > norm)
                norm = fabs(Rez_1[h] - temp[h]);
            Rez_1[h] = temp[h];
        }
    } while (norm > eps);
    delete[] temp;
}

bool converge(double *xk, double *xkp)
{
    double norm = 0;
    for (int i = 0; i < 3; i++)
    {
        norm += (xk[i] - xkp[i]) * (xk[i] - xkp[i]);
    }
    if (sqrt(norm) >= eps)
```

```

        return false;
    return true;
}
void Seidel(int n, double A[3][3], double B[3], double x[3],double p[3])
{
    do
    {
        k_2++;
        for (int i = 0; i<n; i++)
            p[i] = x[i];
        for(int i = 0; i<n; i++)
        {
            double var = 0;
            for(int j = 0; j<i; j++)
                var += (A[i][j] * x[j]);
            for(int j = i + 1; j<n; j++)
                var += (A[i][j] * p[j]);
            x[i] = (B[i] - var) / A[i][i];
        }
    } while(!converge(x, p));
}
int main ()
{
    double A[3][3];
    double B[3];
    double Rez_1[3];
    double Rez_2[3];
    double p[3];

    int n = 3;

    cout<<"Dati datele matricei A: "<<endl;
    for (int i = 0; i<3; i++)
    {
        for (int j = 0; j<3; j++)
        {
            cin>>A[i][j];
        }
    }
    cout<<"Dati datele matricei B: "<<endl;
    for (int i = 0; i<n; i++)
    {
        cin>>B[i];
    }
}

```

```

    }

    for (int i = 0; i<n; i++)
    {
        Rez_1[i] = 0;
        Rez_2[i] = 0;
        p[i] = 0;
    }
    cout<<endl<<"Metoda Gauss - Seidel: "<<endl;
    Seidel(n, A, B, Rez_2, p);
    for (int i = 0; i<n; i++)
    {
        cout<<Rez_2[i]<<" ";
    }
    cout<<"Numarul de iteratii: "<<k_2<<endl;
    return 0;
}

```

Rezultat:

```

Dati datele matricei A:
18 -7 9
-7 20 -3
9 -3 13
Dati datele matricei B:
9
-4
-4

Metoda Gauss - Seidel:
0.999569 -3.1602e-005 -0.999709 Numarul de iteratii: 8

```

Concluzie:

În cadrul laboratorului nr. 2 am studiat rezolvarea numerică a sistemului de ecuații liniare. Am folosit 4 metode de rezolvare: Gauss, Jacobi, Gauss – Seidel și Choletsky. Analizând datele obținute și iterațiile obținute de la fiecare metodă observăm că din metodele iterative Gauss – Seidel este cea mai eficientă pentru că în cadrul algoritmului folosim rădăcina obținută în cadrul aceleiași iterații dar nu în următoarea ca în cazul metodei Jacobi.