НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ ТА СПЕЦІАЛІЗОВАНИХ КОМП’ЮТЕРНИХ СИСТЕМ

**Лабораторна робота №3  
з дисципліни «Алгоритми та методи обчислення»  
на тему: «РОЗВ’ЯЗАННЯ СИСТЕМ ЛІНІЙНИХ АЛГЕБРАЇЧНИХ РІВНЯНЬ»**

**Варіант 21**

Виконав:  
студент 3-го курсу,  
гр. КВ-41,  
Яковенко Максим

Київ – 2016

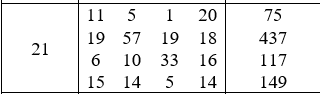
**Завдання для лабораторної роботи**

Розв’язати СЛАР (відповідно до варіанту) одним прямим й одним ітераційним методом.

3 – схема з вибором головного елемента, метод ітерації Зейделя;

Звіт про лабораторну роботу має містити вихідний текст програми, таблицю з результатами та висновки.

**Варіант 21:**



**Текст програми**:

**Methods.h**

#ifndef \_METHODS\_H

#define \_METHODS\_H

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <cmath>

#include <iostream>

#include <fstream>

#include <ios>

#include <iomanip>

#define methods

#define M 4

#define N 5

void gaussian\_elimination(double \*\*MATRIX, double x[M], int si, int sj);

void gauss\_seidel(double MATRIX[M][N], double x[M], double eps);

#endif

**Methods.cpp**

#include "methods.h"

void gaussian\_elimination(double \*\*MATRIX, double x[M], int si, int sj)

{

int index, sub\_index;

double tmp;

tmp = MATRIX[si][sj];

for(sub\_index = sj; sub\_index < N; sub\_index++) MATRIX[si][sub\_index] = MATRIX[si][sub\_index]/tmp;

for(index = si+1; index < M; index++)

{

tmp = MATRIX[index][sj];

for(sub\_index = sj; sub\_index < N; sub\_index++)

MATRIX[index][sub\_index] = MATRIX[index][sub\_index] - tmp \* MATRIX[si][sub\_index];

}

if (si < M) gaussian\_elimination(MATRIX, x, ++si, ++sj);

si -= 1;

sj -= 1;

tmp = 0.0;

for(index = 0; index < (M-si-1); index++)

tmp = MATRIX[si][sj+index+1]\*x[si+index+1] + tmp;

x[si] = MATRIX[si][N-1] - tmp;

return;

}

//Gauss-Seidel Method

void gauss\_seidel(double MATRIX[M][N], double x[M], double eps)

{

int i, j;

double tmp2, maxx;

double \*\*S\_MATRIX = new double\*[M];

for (i = 0; i < N; i++) S\_MATRIX[i] = new double[N];

double \*tmp = new double[M];

for (i = 0; i < M; i++) tmp[i] = 0.0;

for(i = 0; i < M; i++)

if((i == 1) || (i == 2))

for(j = 0; j < N; j++)

S\_MATRIX[i][j] = MATRIX[i][j];

for(j = 0; j < N; j++) S\_MATRIX[0][j] = MATRIX[0][j] + 2\*MATRIX[2][j] - 5 \* MATRIX[3][j];

for (j = 0; j < N; j++) S\_MATRIX[3][j] = MATRIX[3][j] + 3 \* MATRIX[0][j] - MATRIX[1][j] - MATRIX[2][j] ;

//vector beta

double \*beta;

beta = new double[M];

for (i = 0; i < M; i++) beta[i] = S\_MATRIX[i][N-1] / S\_MATRIX[i][i];

//matrix alfa

double \*\*alfa;

alfa = new double\*[M];

for(i = 0; i < M; i++) alfa[i] = new double[M];

for(i = 0; i < M; i++)

for(j = 0; j < M; j++)

{

if(i != j) alfa[i][j] = - S\_MATRIX[i][j] / S\_MATRIX[i][i];

else alfa[i][j] = 0;

};

for(i = 0; i < M; i++) x[i] =beta[i];

double q = 0.0, max;

for(i = 0; i < M; i++)

{

max = 0.0;

for(j = 0; j < M; j++) max = fabs(alfa[i][j]) + max;

if(max > q) q = max;

};

do

{

for(i = 0; i < M; i++) tmp[i] = x[i];

for(i = 0; i < M; i++)

{

tmp2 = 0.0;

for(j = 0; j < M; j++) if (i!=j) tmp2 = alfa[i][j] \* x[j] + tmp2;

x[i] = beta[i] + tmp2;

};

maxx = 0.0;

tmp2 = 0.0;

for(i = 0; i < M; i++)

{

tmp2 = fabs(tmp[i] - x[i]);

if(tmp2 > maxx) maxx = tmp2;

}

} while(maxx > fabs((1 - q) \* eps / q));

return;

}

**Main.cpp**

#include "methods.h"

using namespace std;

int main()

{

int index, sub\_index, si = 0, sj = 0;

double eps = 0.001;

double MATRIX[M][N] = {{11.0,5.0,1.0,20.0,75.0 },

{19.0,57.0,19.0,18.0,437.0},

{6.0,10.0,33.0,16.0,117.0 },

{15.0,14.0,5.0,14.0,149.0}};

double \*\*S\_MATRIX = new double\*[M];

for (index = 0; index < N; index++) S\_MATRIX[index] = new double[N];

for(index = 0; index < M; index++)

for(sub\_index = 0; sub\_index < N; sub\_index++)

S\_MATRIX[index][sub\_index] = MATRIX[index][sub\_index];

double \*x=new double[M];

for(index = 0; index < M; index++) x[index] = 0.0;

cout<<" Gaussian Elimination Method "<<endl;

cout << " x1 " << " x2 " << " x3 " << " x4 " << endl;

cout<<"-------------------------------------------------------------------" <<endl;

gaussian\_elimination(S\_MATRIX, x, si, sj);

for (index = 0; index < M; index++) cout<<setw(14)<<x[index]<<" | ";

cout<<endl<<"-------------------------------------------------------------------"<<endl<<endl;

cout<<" Gauss-Seidel Method "<<endl;

cout << " x1 " << " x2 " << " x3 " << " x4 " << endl;

cout<<"-------------------------------------------------------------------"<<endl;

gauss\_seidel(MATRIX, x, eps);

for (index = 0; index < M; index++)

cout<<setw(14)<<x[index]<<" | ";

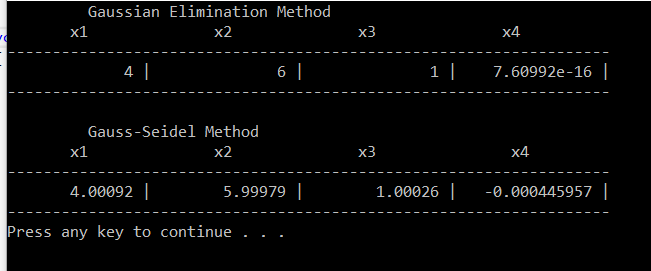
cout<<endl<<"-------------------------------------------------------------------"<<endl;

system("PAUSE");

return 0;

}

**Таблиці результатів:**

****

**Висновки:**

В ході виконання лабораторної роботи ми досліджували методи розв’язання системи лінійних алгебраїчних рівнянь, а саме один прямий метод(схема прямого поділу) та один ітераційний(метод Зейделя).

Принцип роботи методу виключення Гаусса(схема єдиного поділу) ґрунтується на ідеї еквівалентного перетворення вихідної системи до трикутного вигляду(прямий хід) та подальшого її розв’язання(зворотній хід). Цей метод має значні недоліки, а саме: якщо один з головних елементів виявиться рівним нулю, то неможливо розв’язати СЛАР, тоді як система може виявитись сумісною і мати розв’язок.

Метод Зейделя(Гаусса-Зейделя) є модифікованою версією метода простої ітерації, в якій к+1 наближення невідомого Хі враховуються вже обраховані значення Хі-1, Хі-2,…,Х1. Це дає змогу отримати кращу збіжність, ніж метод простої ітерації. Проте необхідно перед початком роботи метод звести систему до вигляду придатного до ітерації. Робиться це так. Із заданої системи виділяють рівняння з коефіцієнтами, модулі яких більші за суму модулів решти коефіцієнтів рівняння. Кожне виділене рівняння записують у такий рядок нової системи, щоб найбільший за модулем коефіцієнт став діагональним.

З решти невикористаних та виділених рівнянь системи залишають лінійно незалежні комбінації, і кожне невикористане ще рівняння потрапило б принаймні в одну лінійну комбінацію, яка є рівнянням нової системи.