МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Воронежский государственный технический университет»

(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет Информационных Технологий и Компьютерной Безопасности

Кафедра систем управления и информационных технологий в строительстве

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

по дисциплине Информатика

### Тема «решение системы линейных алгебраических уравнений (СЛАУ) методом Якоб, методом Зейделя, матричным методом, методом Гаусса»

Выполнил студент: Рыженков Евгений

Группа: бИСТ-221

Руководитель: доцент, к.т.н. Ефимова О.Е.

Работа защищена « » 2023г.

С оценкой \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

подпись

Воронеж 2023

**Лабораторная работа №4.**

**Решение систем линейных алгебраических уравнений.**

Цель работы: найти решение системы линейных алгебраических уравнений разными способами: методом Гаусса, матричным методом, методом Якоби и методом Зейделя.

**Задания на лабораторную работу**

**Задание 1**. Найти решение системы линейных алгебраических уравнений методом Гаусса и матричным методом.

**Задание 2**. Найти решение системы линейных алгебраических уравнений методом Якоби и методом Зейделя с заданной точностью.

|  |
| --- |
|  |

**Результат выполнения заданий**

|  |
| --- |
|  |
| Рисунок 1 – результат решения СЛАУ матричным методом |

|  |
| --- |
|  |
| Рисунок 2 – результат решения СЛАУ методом Гаусса |

|  |
| --- |
|  |
| Рисунок 3 – результат неудачного решения СЛАУ методом Якоби |

|  |
| --- |
|  |
| Рисунок 4 – результат неудачного решения СЛАУ методом Зейделя |

|  |
| --- |
|  |
| Рисунок 5 – результат решения СЛАУ методом Якоби |

|  |
| --- |
|  |
| Рисунок 6 – результат решения СЛАУ методом Зейделя |

**Листинг программы на языке С++**

#include <iostream>

using namespace std;

void show\_system(float\*\* A, float\* Y, int n);

float\* gauss(float\*\* A, float\* Y, int n);

float\* matrix\_method(float\*\* A, float\* Y, int n);

float determinant(float\*\* A);

float\* Jacobi\_method(float\*\* A, float\* Y, float eps);

float\* Seidel\_method(float\*\* A, float\* Y, float eps);

int main() {

setlocale(LC\_ALL, "RUS");

float\*\* A;

float\* Y, \* x, eps;

int n, v = 1;

while (true) {

cout << "Выберите действие: " << endl <<

"1) Решение СЛАУ матричным методом" << endl <<

"2) Решение СЛАУ методом Гаусса" << endl <<

"3) Решение СЛАУ методом Якоби" << endl <<

"4) Решение СЛАУ методом Зейделя" << endl <<

"0) Выход" << endl;

cin >> v;

if (v == 0) break;

cout << "Введите размер n квадратной матрицы: " << endl;

cin >> n;

A = new float\* [n];

Y = new float[n];

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

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

if (j == 0) cout << "A[" << i + 1 << "][" << j + 1 << "]\*X" << j + 1;

else cout << " + A[" << i + 1 << "][" << j + 1 << "]\*X" << j + 1;

}

cout << " = " << "Y[" << i + 1 << "]" << endl;

} cout << endl;

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

A[i] = new float[n];

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

cout << "A[" << i + 1 << "][" << j + 1 << "]= ";

cin >> A[i][j];

}

cout << "Y[" << i + 1 << "]=";

cin >> Y[i];

} cout << endl;

switch (v) {

case 1:

show\_system(A, Y, n);

x = matrix\_method(A, Y, n);

for (int i = 0; i < n; i++) cout << "x" << i + 1 << "=" << x[i] << endl;

break;

case 2:

show\_system(A, Y, n);

x = gauss(A, Y, n);

for (int i = 0; i < n; i++) cout << "x" << i + 1 << "=" << x[i] << endl;

break;

case 3:

show\_system(A, Y, n);

if (fabs(A[0][0]) <= fabs(A[0][1]) + fabs(A[0][2]) || fabs(A[1][1]) <= fabs(A[1][0]) + fabs(A[1][2]) || fabs(A[2][2]) <= fabs(A[2][1]) + fabs(A[2][0])) {

cout << "Невозможно применить метод Якоби, так как должно выполняться условие: " << endl <<

"| A[1][1] | > | A[1][2] | + | A[1][3] |" << endl <<

"| A[2][1] | > | A[2][2] | + | A[2][3] |" << endl <<

"| A[3][1] | > | A[3][2] | + | A[3][3] |" << endl;

continue;

}

cout << "Введите точность: " << endl;

cin >> eps;

x = Jacobi\_method(A, Y, eps);

for (int i = 0; i < n; i++) cout << "x" << i + 1 << "=" << x[i] << endl;

break;

case 4:

show\_system(A, Y, n);

if (fabs(A[0][0]) <= fabs(A[0][1]) + fabs(A[0][2]) || fabs(A[1][1]) <= fabs(A[1][0]) + fabs(A[1][2]) || fabs(A[2][2]) <= fabs(A[2][1]) + fabs(A[2][0])) {

cout << "Невозможно применить метод Зейделя, так как должно выполняться условие: " << endl <<

"| A[1][1] | > | A[1][2] | + | A[1][3] |" << endl <<

"| A[2][1] | > | A[2][2] | + | A[2][3] |" << endl <<

"| A[3][1] | > | A[3][2] | + | A[3][3] |" << endl;

continue;

}

cout << "Введите точность: " << endl;

cin >> eps;

x = Seidel\_method(A, Y, eps);

for (int i = 0; i < n; i++) cout << "x" << i + 1 << "=" << x[i] << endl;

break;

}

}

}

void show\_system(float\*\* A, float\* Y, int n) {

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

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

if (j == 0) {

cout << A[i][j] << "\*X" << j + 1 << " ";

continue;

}

if (A[i][j] < 0) cout << " - ";

else cout << " + ";

cout << fabs(A[i][j]) << "\*X" << j + 1 << " ";

}

cout << " = " << Y[i] << endl;

}

cout << endl;

}

float\* gauss(float\*\* A, float\* Y, int n) {

float\* x, max;

int k, index;

x = new float[n];

k = 0;

while (k < n) {

max = abs(A[k][k]);

index = k;

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

if (abs(A[i][k]) > max) {

max = abs(A[i][k]);

index = i;

}

}

if (max == 0) {

cout << "Решение получить невозможно" << endl;

return 0;

}

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

float temp = A[k][j];

A[k][j] = A[index][j];

A[index][j] = temp;

}

float temp = Y[k];

Y[k] = Y[index];

Y[index] = temp;

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

float temp = A[i][k];

if (abs(temp) == 0) continue;

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

A[i][j] = A[i][j] / temp;

Y[i] = Y[i] / temp;

if (i == k) continue;

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

A[i][j] -= A[k][j];

Y[i] = Y[i] - Y[k];

}

k++;

}

for (k = n - 1; k >= 0; k--) {

x[k] = Y[k];

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

Y[i] -= A[i][k] \* x[k];

}

return x;

}

float\* matrix\_method(float\*\* A, float\* Y, int n)

{

float detA = determinant(A);

float\* Y\_new, \*\* A\_new;

Y\_new = new float[n];

A\_new = new float\* [n];

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

A\_new[i] = new float[n];

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

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

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

A\_new[i][j] = A[i][j];

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

A\_new[j][o] = Y[j];

Y\_new[o] = determinant(A\_new) / detA;

}

return Y\_new;

}

float determinant(float\*\* A) {

return A[0][0] \* A[1][1] \* A[2][2] +

A[0][1] \* A[1][2] \* A[2][0] +

A[1][0] \* A[2][1] \* A[0][2] -

A[2][0] \* A[1][1] \* A[0][2] -

A[1][0] \* A[0][1] \* A[2][2] -

A[2][1] \* A[1][2] \* A[0][0];

}

float\* Jacobi\_method(float\*\* A, float\* Y, float eps) {

float\* X = new float[3];

float X\_old[3];

int i = 1;

X\_old[0] = (Y[0] - A[0][1] \* 0 - A[0][2] \* 0) / A[0][0];

X\_old[1] = (Y[1] - A[1][0] \* 0 - A[1][2] \* 0) / A[1][1];

X\_old[2] = (Y[2] - A[2][0] \* 0 - A[2][1] \* 0) / A[2][2];

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("| Итерация | х1 | х2 | х2 |\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

do {

if (i != 1) {

X\_old[0] = X[0];

X\_old[1] = X[1];

X\_old[2] = X[2];

}

X[0] = (Y[0] - A[0][1] \* X\_old[1] - A[0][2] \* X\_old[2]) / A[0][0];

X[1] = (Y[1] - A[1][0] \* X\_old[0] - A[1][2] \* X\_old[2]) / A[1][1];

X[2] = (Y[2] - A[2][0] \* X\_old[0] - A[2][1] \* X\_old[1]) / A[2][2];

printf("|%10d|%14f|%14f|%14f|\n", i, X\_old[0], X\_old[1], X\_old[2]);

printf("--------------------------------------------------------\n");

i++;

} while (fabs(X[0] - X\_old[0]) > eps && fabs(X[1] - X\_old[1]) > eps && fabs(X[2] - X\_old[2]) > eps);

return X;

}

float\* Seidel\_method(float\*\* A, float\* Y, float eps) {

float\* X = new float[3];

float x\_old\_prov[3];

int i = 1;

X[0] = (Y[0] - A[0][1] \* 0 - A[0][2] \* 0) / A[0][0];

X[1] = (Y[1] - A[1][0] \* 0 - A[1][2] \* 0) / A[1][1];

X[2] = (Y[2] - A[2][0] \* 0 - A[2][1] \* 0) / A[2][2];

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("| Итерация | х1 | х2 | х2 |\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

do {

x\_old\_prov[0] = X[0];

x\_old\_prov[1] = X[1];

x\_old\_prov[2] = X[2];

X[0] = (Y[0] - A[0][1] \* X[1] - A[0][2] \* X[2]) / A[0][0];

X[1] = (Y[1] - A[1][0] \* X[0] - A[1][2] \* X[2]) / A[1][1];

X[2] = (Y[2] - A[2][0] \* X[0] - A[2][1] \* X[1]) / A[2][2];

printf("|%10d|%14f|%14f|%14f|\n", i, X[0], X[1], X[2]);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

i++;

} while (fabs(X[0] - x\_old\_prov[0]) > eps && fabs(X[1] - x\_old\_prov[1]) > eps && fabs(X[2] - x\_old\_prov[2]) > eps);

return X;

}