Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Отчет

по лабораторной работе № 1

«**Решение систем линейных уравнений методом Гаусса**»

по дисциплине «Вычислительные алгоритмы»

Студент гр. ПИ-02

Чередов Р.А.

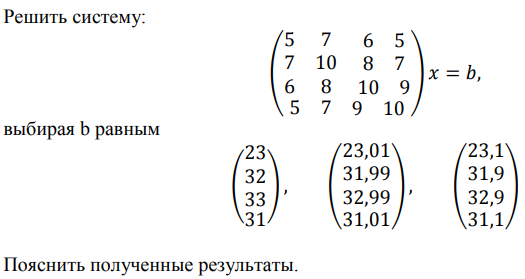
Преподаватель, к.ф.-м.н., доцент,

Проскурин А. В.

Барнаул 2023

**Задание:**

1. Составить программу для решения уравнений методом Гаусса с выбором главного элемента и нахождения определителя матрицы системы. Исходные данные — матрица системы уравнений и столбец свободных членов должны читаться из файла, а результаты расчетов помещаться в файл. В случае, когда матрица системы вырождена, выдать об этом сообщение. В противном случае вывести решение системы, невязки, величину определителя, обратную матрицу. Подобрать тестовые примеры, предусматривающие различные ситуации (матрица вырожденная, невырожденная) и провести вычисления.



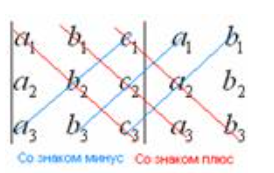
**Краткое описание алгоритма:**

В рамках реализации данного задания из текстового файла считывается размерность матрицы, создается двумерный массив, для хранения расширенной матрицы. Чтение расширенной матрицы происходит из файла input.txt поэлементно. Логика программы реализуется в функциях(функции расположены в хронологическом порядке выполнения):

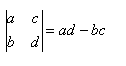
* fscanf\_s(in, "%d", &N) – функция считывания из файла размерности
* void EntryMass(int N, float\* A, float\* ASave, FILE\* in); - функция считывания матрицы из файла
* void EntryB(int N, float\* B, float\* BSave, FILE\* in); - функция считывания столбца свободных членов
* void OutPutMass(int N, float\* A) – функция вывода матрицы
* void OutPutB(int N, float\* B); - функция вывода свободных членов
* int MaxInColumn(int N, float\* A, int col); - функция определения наибольшего по модуля элемент в столбце
* int LineTrans(int N, float\* A, float\* B, float\* E, int str, int col); - функция для замены строк местами
* void Frontmove(int N, float\* A, float\* B, float\* E, int\* CountTrans) – прямой ход метода гаусса
* float CalcDet(int N, float\* A, int CountTrans) – функция для нахождения определителя матрицы
* void ReverseMove(int N, float\* A, float\* B, float\* X); - обратный ход метода гаусса
* void OutPutX(int N, float\* X, FILE\* out); - функция вывода результата вычислений и занесение их в файл
* void CaclNevyaz(int N, float\* A, float\* B, float\* X, float\* Nev); - функция вычисления невязок
* void OutPutNevyaz(int N, float\* Nev, FILE\* out); - функция вывода невязок и занесение их в файл
* void OutPutDet(float det, FILE\* out); - функция вывода определителя и занесение их в файл
* void ReverseMoveForE(int N, float\* A, float\* E, float\* InvA); - функция обратного хода Гаусса с единичной матрицей
* void OutPutRevA(int N, float\* InvA, FILE\* out); - функция вывода обратного хода Гаусса с единичной матрицей и занесение их в файл

**Объяснение вычислений:**

**Вычисление определителя:**



Числа по диагонали перемножаются



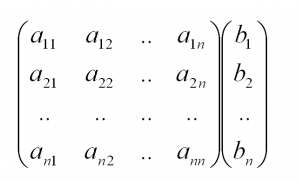
Множители, находящиеся на «красных» диагоналях входят в формулу со знаком «плюс».  
Множители, находящиеся на «синих» диагоналях входят в формулу со знаком минус:

**Вырожденная матрица:**

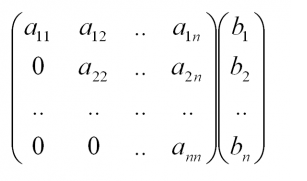
Квадратная матрица https://lfirmal.com/wp-content/uploads/2020/03/image-4770.png называется невырожденной, если определитель https://lfirmal.com/wp-content/uploads/2020/03/image-5192.png не равен нулю: https://lfirmal.com/wp-content/uploads/2020/03/image-5193.png. В противном случае (https://lfirmal.com/wp-content/uploads/2020/03/image-5194.png) матрица https://lfirmal.com/wp-content/uploads/2020/03/image-4770.png называется вырожденной.

**Прямой ход метода Гаусса:**

Сначала запишем расширенную матрицу системы. Для этого в главную матрицу добавляем столбец свободных членов.



Далее данную матрицу приводим к треугольному(ступенчатому) виду путем элементарных преобразований. По-простому под главной диагональю или над ней должны быть одни нули.

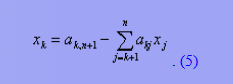


Что можно делать с матрицей:

1. Можно переставлять строки матрицы местами
2. Если в матрице есть одинаковые (или пропорциональные) строки, можно удалить их все, кроме одной
3. Можно умножать или делить строку на любое число (кроме нуля);
4. Нулевые строки удаляются;
5. Можно прибавлять к строке строку, умноженную на число, отличное от нуля.

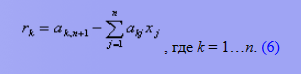
**Обратный ход метода Гаусса:**

Второй этап решения СЛАУ методом Гаусса называется обратным ходом и состоит в последовательном определении xk, начиная с xn, так как для последнего решение фактически получено. Общая формула:

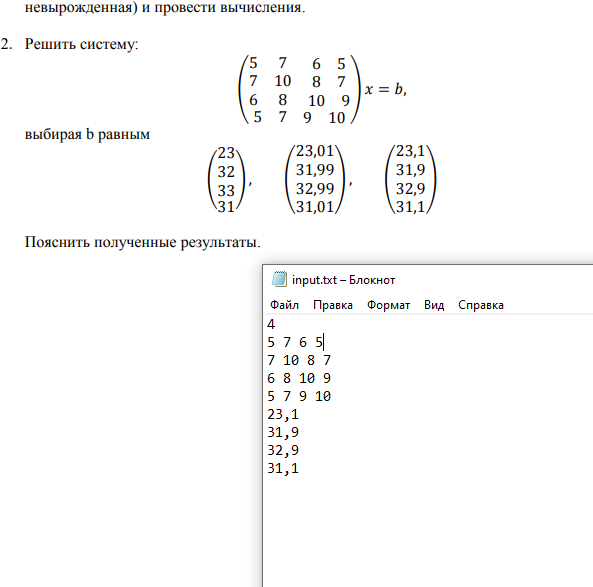


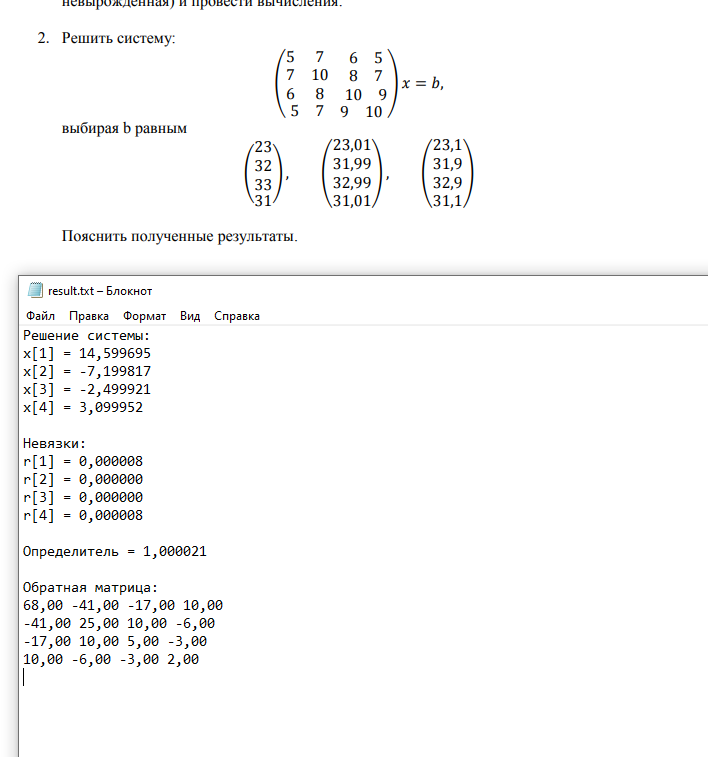
**Погрешность метода. Расчет невязок:**

Точность результатов будет определяться только точностью выполнения арифметических операций при преобразовании элементов матрицы, т.е. ошибкой округления. Контроль правильности полученного решения осуществляется подстановкой полученных значений x1…xn в исходную систему уравнений и вычислением невязок, т.е. разностей между правыми и левыми частями уравнений:



**Ход работы:**

Программа написана на C++. Входные данные хранятся в файле input.txt

Результат записывается в файл – result.txt

Код программы:

**Lastva1.cpp:**

#include "Header.h"

int main()

{

setlocale(0, "Russian");

int N; //Размерность матрицы

FILE\* in; //Файл с исходными данными

FILE\* out; //Файл для вывода результата

in = fopen("input.txt", "rt");//Файл чтения

out = fopen("result.txt", "wt");//Файл записи

fscanf\_s(in, "%d", &N); //взяли N - размерность

float\* A = new float[N \* N];//Матрица системы уравнений

float\* B = new float[N];//Столбец свободных членов

float\* ASave = new float[N \* N];//Изначальная матрица системы уравнений

float\* BSave = new float[N];//Изначальный столбец свободных членов

float\* X = new float[N];//Столбец иксов

float\* InvA = new float[N \* N];//Обратная матрица системы уравнений

float\* E = new float[N \* N]; //Единичная матрица

float\* Nev = new float[N]; //Невязки

int CountTrans = 0; //Кол-во перестановок строк

float det; //Определитель

CreateE(N, E); //Создаем единичную матрицу

EntryMass(N, A, ASave, in); //Считываем массив системы из файла

EntryB(N, B, BSave, in); //Считываем столбец свободных членов из файла

//Выводим массив и стоблец свободных членов

OutPutMass(N, A);

OutPutB(N, B);

//Прямой ход метода Гауса

Frontmove(N, A, B, E, &CountTrans);

//Вычисляем определитель матрицы

det = CalcDet(N, A, CountTrans);

if (det == 0) {

cout << endl << "Матрица вырожденная" << endl;

fprintf(out, "Матрица вырожденная");

}

else {

ReverseMove(N, A, B, X); //Обратный ход метода Гауса

OutPutX(N, X, out); //Выводим решение системы

CaclNevyaz(N, ASave, BSave, X, Nev); //Подсчитываем невязки

OutPutNevyaz(N, Nev, out);

OutPutDet(det, out); //Выводим значение определителя

//Сокращаем точность выводимых значений до 2 знаков после запятой

cout << fixed;

cout.precision(2);

//Обратный ход Гауса с единичной матрицей для получения обратной матрицы

ReverseMoveForE(N, A, E, InvA);

//Вывод обратной матрицы

OutPutRevA(N, InvA, out);

}

//Освобождаем память

delete[] A;

delete[] B;

delete[] ASave;

delete[] BSave;

delete[] InvA;

delete[] X;

delete[] Nev;

fclose(in);

fclose(out);

}

**Header.h :**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <conio.h>

#include <math.h>

#include <fstream>

using namespace std;

void EntryMass(int N, float\* A, float\* ASave, FILE\* in);//Функция считывания из файла матрицы

void EntryB(int N, float\* B, float\* BSave, FILE\* in);//Функция считывания из файла свободных членов

void OutPutMass(int N, float\* A);//Печатаем двумерный массив

void OutPutB(int N, float\* B);//Печатаем одномерный массив

void CreateE(int N, float\* E); //Создаем единичную матрицу

int MaxInColumn(int N, float\* A, int col);//Определяем наибольший по модулю элемент в столбце

int LineTrans(int N, float\* A, float\* B, float\* E, int str, int col);//Меняем строки местами

void Frontmove(int N, float\* A, float\* B, float\* E, int\* CountTrans);//Прямой ход метода Гаусса

void ReverseMove(int N, float\* A, float\* B, float\* X);//Обратный ход метода Гаусса

void ReverseMoveForE(int N, float\* A, float\* E, float\* InvA);//Обратный ход Гаусса с единичной матрицей для получения обратной матрицы

float CalcDet(int N, float\* A, int CountTrans);//Высчитываем определитель матрицы

void CaclNevyaz(int N, float\* A, float\* B, float\* X, float\* Nev);//Высчитываем невязки

void OutPutNevyaz(int N, float\* Nev, FILE\* out);//Вывод невязок и занесение их в файл

void OutPutX(int N, float\* X, FILE\* out);//Вывод результат вычислений и занесение их в файл

void OutPutDet(float det, FILE\* out); // Вывод определителя и занесение его в файл

void OutPutRevA(int N, float\* InvA, FILE\* out); // Вывод обратной матрицы и занесение ее в файл

**Source.cpp:**

#include "Header.h"

//Функции считывания из файла

void EntryMass(int N, float\* A, float\* ASave, FILE\* in)

{

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

{

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

{

fscanf(in, "%f", &(\*(A + i \* N + j)));

\*(ASave + i \* N + j) = \*(A + i \* N + j);

}

}

}//Функция считывания из файла матрицы

void EntryB(int N, float\* B, float\* BSave, FILE\* in)

{

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

{

fscanf(in, "%f", &(B[j]));

BSave[j] = B[j];

}

}

//Вывод на консоль данных из файла

void OutPutMass(int N, float\* A)

{

cout << endl << "Матрица системы уравнений:" << endl;

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

{

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

{

cout << \*(A + i \* N + j) << " ";

}

cout << std::endl;

}

}

void OutPutB(int N, float\* B)

{

cout << endl << "Столбец свободных членов:" << endl;

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

cout << B[i] << "\n";

}

}

//Создание единичной матрицы

void CreateE(int N, float\* E) {

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

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

if (i == j)

\*(E + i \* N + j) = 1;

else

\*(E + i \* N + j) = 0;

}

}

//Определение наибольшего элемента и перестановка строк

int MaxInColumn(int N, float\* A, int col)

{

float max = fabs(\*(A + col \* N + col));//Наибольший элемент в столбце

int str = col;

for (int i = col + 1; i < N; i++)

{

if (fabs(\*(A + i \* N + col)) > max)

{

max = \*(A + i \* N + col);

str = i;

}

}

return str;

}

int LineTrans(int N, float\* A, float\* B, float\* E, int str, int col)

{

if (str == col)//Если наибольшим элементом является текущий, то строки менять местами не нужно

return 0;

else

{

float buffer;

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

{

//Меняем строки матрицы

buffer = \*(A + col \* N + i);

\*(A + col \* N + i) = \*(A + str \* N + i);

\*(A + str \* N + i) = buffer;

//Меняем строки единичной матрицы

buffer = \*(E + col \* N + i);

\*(E + col \* N + i) = \*(E + str \* N + i);

\*(E + str \* N + i) = buffer;

}

//Меняем местами свободные члены

buffer = \*(B + col);

\*(B + col) = \*(B + str);

\*(B + str) = buffer;

return 1;

}

}

//Прямой ход метода Гаусса

void Frontmove(int N, float\* A, float\* B, float\* E, int\* CountTrans)

{

float d;

for (int col = 0; col < N; col++)//проход по столбцам

{

\*CountTrans += LineTrans(N, A, B, E, MaxInColumn(N, A, col), col);

for (int str = col + 1; str < N; str++)//проход по строкам ниже гл. диагонали

{

d = \*(A + str \* N + col) / \*(A + col \* N + col); //коэффициент уничтожения

//Вычитаем из строк матрицы строку текущего элемента гл. диагонали, умноженную на коэф.

for (int i = col; i < N; i++)

\*(A + str \* N + i) = \*(A + str \* N + i) - d \* \*(A + col \* N + i);

//Вычитаем из строк Единичной матрицы строку текущего элемента гл. диагонали, умноженную на коэф.

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

\*(E + str \* N + i) = \*(E + str \* N + i) - d \* \*(E + col \* N + i);

//Вычитаем из нижних значений столбца свободных членов значение,

//которое находится в строке текущего элемента гл. диагонали, умнож. на коэф.

B[str] = B[str] - d \* B[col]; // формула (3)

}

}

}

//Обратный ход для получения решения и для получения обратной матрицы

void ReverseMove(int N, float\* A, float\* B, float\* X)

{

float d, s;

d = 0;

for (int k = N - 1; k >= 0; k--) // обратный ход

{

d = 0;

//Высчитываем сумму левой части с уже известными Х

for (int j = k + 1; j < N; j++)

{

s = \*(A + k \* N + j) \* X[j];

d = d + s;

}

X[k] = (B[k] - d) / \*(A + k \* N + k); //Получаем значение Х

}

}

void ReverseMoveForE(int N, float\* A, float\* E, float\* InvA)

{

float\* d = new float[N];

float s;

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

d[i] = 0;

for (int str = N - 1; str >= 0; str--) // обратный ход

{

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

d[i] = 0;

//Высчитываем суммы левой части с уже известными значениями обратной матрицы

for (int j = str + 1; j < N; j++)

{

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

s = \*(A + str \* N + j) \* \*(InvA + j \* N + i);

d[i] += s;

}

}

//Получаем элементы обратной матрицы

for (int i = N - 1; i >= 0; i--)

\*(InvA + str \* N + i) = (\*(E + str \* N + i) - d[i]) / \*(A + str \* N + str);

}

}

//Вычисление определителя

float CalcDet(int N, float\* A, int CountTrans) {

float det;

if (CountTrans % 2 == 0)//Если четный

det = 1;

else

det = -1;

//Определитель треуг. матрицы равен произведению элементов гл. диагонали

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

det \*= \*(A + i \* N + i);

return det;

}

//Вычисление невязок

void CaclNevyaz(int N, float\* A, float\* B, float\* X, float\* Nev) {

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

float sum = 0;

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

sum += \*(A + i \* N + j) \* \*(X + j);

}

\*(Nev + i) = \*(B + i) - sum;

}

}

//Вывод результатов в консоль и запись их в файл

void OutPutNevyaz(int N, float\* Nev, FILE\* out) {

cout << endl << "Невязки: " << endl;

fprintf(out, "\nНевязки: \n");

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

{

fprintf(out, "r[%d] = %f \n", i + 1, Nev[i]);

cout << "r[" << i + 1 << "]=" << \*(Nev + i) << " " << endl;

}

}

void OutPutX(int N, float\* X, FILE\* out)

{

cout << endl << "Решение системы: " << endl;

fprintf(out, "Решение системы: \n");

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

{

fprintf(out, "x[%d] = %f \n", i + 1, X[i]);

cout << "x[" << i + 1 << "]=" << X[i] << " " << endl;

}

}

void OutPutDet(float det, FILE\* out) {

cout << endl << "Определитель = " << det << endl;

fprintf(out, "\nОпределитель = %f \n", det);

}

void OutPutRevA(int N, float\* InvA, FILE\* out) {

cout << endl << "Обратная матрица: " << endl;

fprintf(out, "\nОбратная матрица: \n");

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

{

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

fprintf(out, "%.2f ", \*(InvA + i \* N + j));

cout << \*(InvA + i \* N + j) << " ";

}

fprintf(out, "\n");

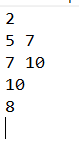
cout << endl;

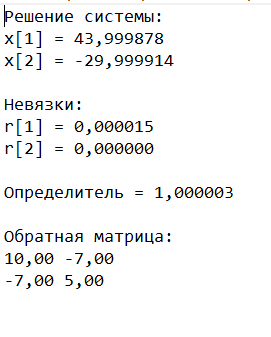
}

}

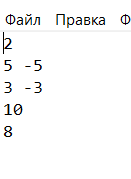
**Тесты:**

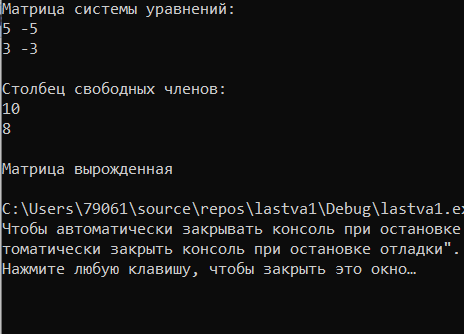
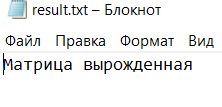
**1)**

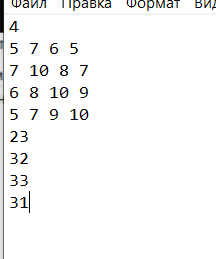


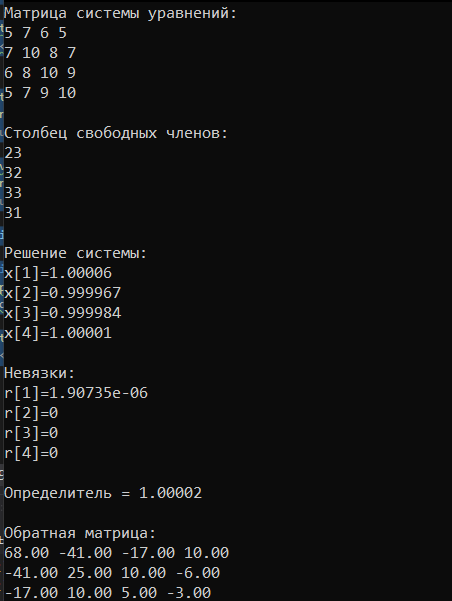
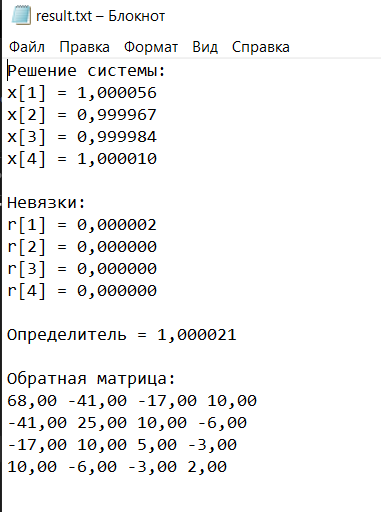
**2)**



3)



4) 