**Лабораторная работа 1**

**«Метод Гаусса»**

Пусть дана система линейных алгебраических уравнений вида

*a*11*x*1 + *a*12*x*2 +  … + *a*1n*xn* = *b*1,

*a*21*x*2 + *a*22*x*2 +  … + *a*2n*xn* = *b*2,

. . . . . . . . . . . . . . . . . . . . . . . .

*an*1*x*1 + *an*2*x*2 +  … + *annxn* = *bn* .

**Задание 1.** Разработать программу численного решения СЛАУ методом Гаусса без выбора ведущего элемента.

Матрицу системы задать с диагональным преобладанием следующим образом:

* недиагональные элементы *ai,j*, *i≠j*, выбираются из чисел 0, –1, –2, –3, *–*4 произвольным образом;
* *ai,i=*, 2≤*i*≤*n*;
* *a*11*=*, *k*≥0.

Правую часть *b* задать умножением матрицы *A* на вектор *x=*(*m*, *m*+1, ... , *n*+*m*–1): *b=Ax*.

Для вычислений выбрать параметры:

* *m* – номер в списке студенческой группы;
* *n* – одно из чисел в пределах от 12 до 15 (12 для сдачи в конце семестра);
* *k* – рассмотреть два случая: *k=*0, *k=*(номер студенческой группы); элементы *ai,j* при фиксированных *i* и *j* в обоих случаях одни и те же (матрицы отличаются только элементом *a*11).

Программно реализовать (C или C++) вычисления для рассматриваемого примера. Для вычислений использовать тип float.

**Задание 2.** Разработать программу численного решения СЛАУ методом Гаусса с выбором ведущего элемента по столбцу.

Для заполнения матрицы *A* использовать случайные числа из диапазона от −100 до 100. Правую часть *b* задать умножением матрицы *A* на вектор *x=*(*m*, *m*+1, ... , *n*+*m*–1): *b=Ax*.

Для вычислений выбрать параметры:

* *m* – номер в списке студенческой группы;
* *n* – одно из чисел в пределах от 15 до 20 (12 для сдачи в конце семестра).

Программно реализовать вычисления для рассматриваемого примера методом Гаусса с выбором ведущего элемента и методом Гаусса без выбора ведущего элемента (система уравнений в обоих случаях одна и та же). Для вычислений использовать тип float.

**Входные данные**

**Задание 1:**

1. k=0, n=15, m=13;
2. k=3, n=15, m=13;

**Задание 2:**

n=15, m=13;

**Листинг программы**

#include <iostream>

#include <cmath>

#include <random>

#include <iomanip>

using namespace std;

//Размерность матрицы

int n = 15;

//Номер в списке группы

int m = 13;

//Номер группы или 0

int k = 0;

//Нахождение случайного числа

int Rand(int L, int R) {

static random\_device rd;

static mt19937 gen(rd());

uniform\_int\_distribution<> dis(L, R);

return dis(gen);

}

//Вывод матрицы

void PrintMatrix(float\*\* A)

{

cout << "A:" << endl;

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

{

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

{

cout << fixed << setprecision(3) << setw(10) << A[i][j] << " ";

}

cout << endl;

}

cout << endl;

}

//Вывод столбца

void PrintArray(float\* x)

{

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

cout << fixed << setprecision(7) << setw(10) << x[i] << " ";

cout << endl << endl;

}

//Умножение матрицы на столбец

void Multiplication(float\*\* A, float\* x, float\* b)

{

float s;

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

{

s = 0;

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

{

s += A[i][j] \* x[j];

}

b[i] = s;

}

}

//Нахождение решения СЛАУ без выбора ведущего элемента

void Gauss\_1(float\*\* A, float\* b)

{

float tm;

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

{

tm = A[i][i];

b[i] /= tm;

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

A[i][j] /= tm;

for (int j = i + 1; j < n; ++j)

{

tm = A[j][i];

A[j][i] = 0;

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

A[j][k] -= tm \* A[i][k];

b[j] -= tm \* b[i];

}

if (!i)

{

cout << "Преобразованная матрица A после первого шага алгоритма:" << endl;

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

{

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

{

cout << fixed << setprecision(5) << setw(10) << A[i][j] << " ";

}

cout << setw(10) << "|" << b[i] << endl;

}

cout << endl;

}

}

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

{

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

{

tm = A[j][i];

A[j][i] = 0;

b[j] -= tm \* b[i];

}

}

cout << "Вектор приближенного значения x\*:" << endl;

PrintArray(b);

}

//Нахождение решение СЛАУ с выбором ведущего элемента по столбцу

void Gauss\_2(float\*\* A, float\* b)

{

float tm;

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

{

int ind = i;

float mx = A[i][i];

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

if (fabs(A[j][i]) > mx) {

mx = fabs(A[j][i]);

ind = j;

}

}

swap(A[i], A[ind]);

swap(b[i], b[ind]);

tm = A[i][i];

b[i] /= tm;

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

A[i][j] /= tm;

for (int j = i + 1; j < n; ++j)

{

tm = A[j][i];

A[j][i] = 0;

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

A[j][k] -= tm \* A[i][k];

b[j] -= tm \* b[i];

}

if (!i)

{

cout << "Преобразованная матрица A после первого шага алгоритма:" << endl;

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

{

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

{

cout << fixed << setprecision(5) << setw(10) << A[i][j] << " ";

}

cout << setw(10) << "|" << b[i] << endl;

}

cout << endl;

}

}

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

{

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

{

tm = A[j][i];

A[j][i] = 0;

b[j] -= tm \* b[i];

}

}

cout << "Вектор приближенного значения x\*:" << endl;

PrintArray(b);

}

//Нахождение относительной погрешности

float Fault(float\* x, float\* b)

{

float\* dx = new float[n];

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

dx[i] = x[i] - b[i];

float ndx = fabs(dx[0]);

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

if (fabs(dx[i]) > ndx)

ndx = fabs(dx[i]);

float nx = fabs(x[0]);

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

if (fabs(x[i]) > nx)

nx = fabs(x[i]);

cout << fixed << setprecision(7) << "Относительная погрешность: " << 100 \* ndx / nx << "%" << endl << endl;

return ndx / nx;

}

int main()

{

setlocale(LC\_ALL, "rus");

//Задача №1

//k=0

//Создание матрицы n\*n

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

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

{

A[i] = new float[n];

}

//Инициализация

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

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

if (i != j)

A[i][j] = Rand(-4, 0);

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

{

float s = 0;

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

if (i != j)

s -= A[i][j];

A[i][i] = s;

}

A[0][0] += pow(10, -k);

//Вывод матрицы

PrintMatrix(A);

//Создание и инициализация вектора x

float\* x = new float[n];

cout << "x:" << endl;

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

x[i] = m + i;

PrintArray(x);

//Создание и нахождение вектора b

float\* b = new float[n];

Multiplication(A, x, b);

cout << "b:" << endl;

PrintArray(b);

//Нахождение решения и вычисление погрешности

Gauss\_1(A, b);

Fault(x, b);

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

//k=3

k = 3;

//Инициализация матрицы

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

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

if (i != j)

A[i][j] = Rand(-4, 0);

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

{

float s = 0;

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

if (i != j)

s -= A[i][j];

A[i][i] = s;

}

A[0][0] += pow(10, -k);

//Вывод матрицы

PrintMatrix(A);

//Создание и нахождение вектора b

Multiplication(A, x, b);

cout << "b:" << endl;

PrintArray(b);

//Нахождение решения и вычисление погрешности

Gauss\_1(A, b);

Fault(x, b);

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

//Задача №2

//Инициализация матрицы A

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

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

A[i][j] = Rand(-100, 100);

//Вывод матрицы

PrintMatrix(A);

//Нахождение вектора b

Multiplication(A, x, b);

cout << "b:" << endl;

PrintArray(b);

//Нахождение решения и вычисление погрешности

Gauss\_2(A, b);

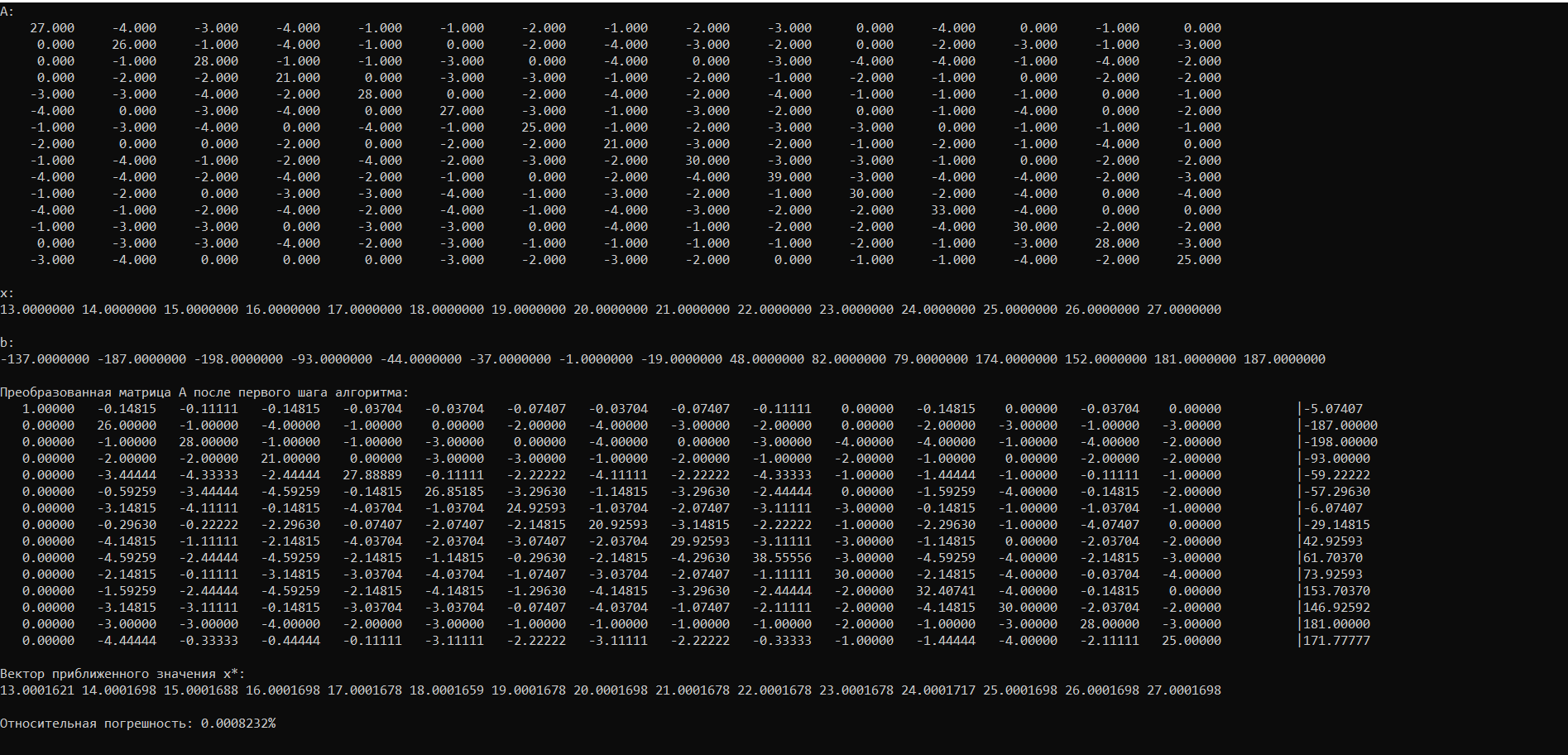
Fault(x, b);

return 0;

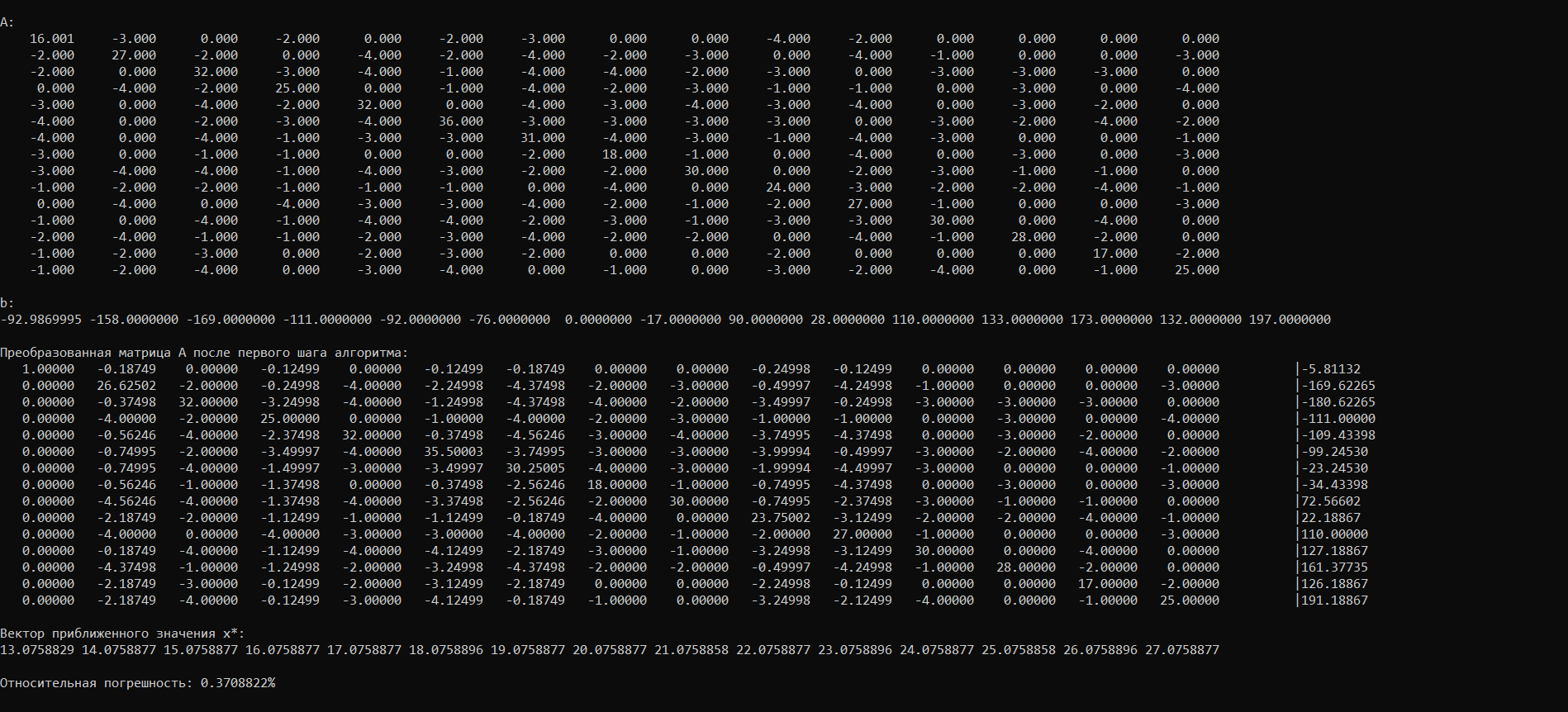
}

**Выходные данные**

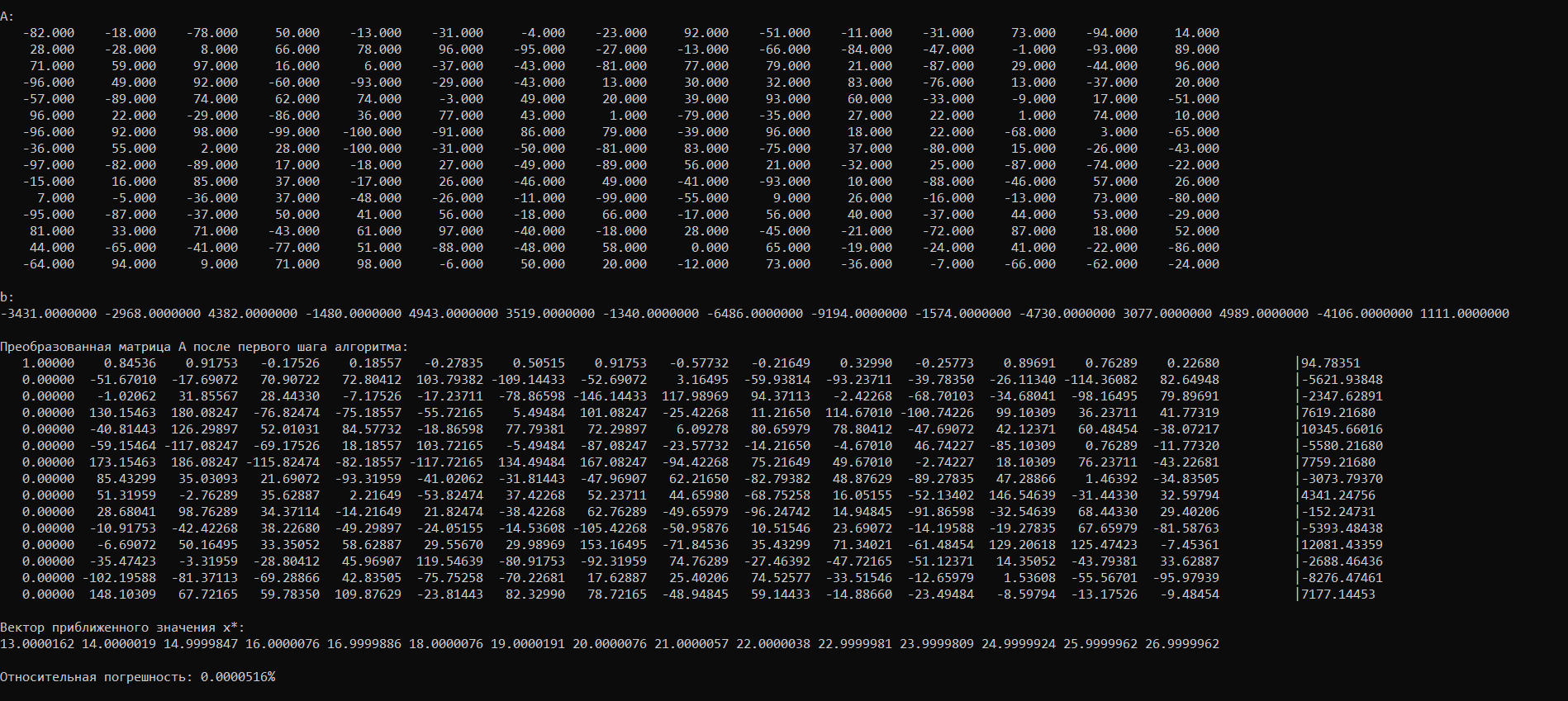
1. Метод Гаусса без выбора главного элемента (k=0)



1. Метод Гаусса без выбора главного элемента (k=3)



1. Метод Гаусса с выбором главного элемента по столбцу



**Выводы**

В первом задании при k=3 погрешность получилась в разы больше, чем при k=0, связано это с тем, что при k=3 происходит потеря значащих цифр.

Во втором задании метод Гаусса с выбором ведущего элемента является более точным, чем метод Гаусса без выбора ведущего элемента. Разница появляется из-за потери значащих цифр в методе без выбора ведущего элемента.