Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Томский государственный университет систем управления и радиоэлектроники»

Факультет систем управления (ФСУ)

Кафедра автоматизированных систем управления (АСУ)

**РЕШЕНИЕ ЗАДАЧ ЛИНЕЙНОЙ АЛГЕБРЫ**

**Отчет по лабораторной работе №2**

**По дисциплине**

**«Численные методы»**

Обучающийся гр. 431-3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Андреев Д.П.

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

Руководитель: ст. Преподаватель каф. АСУ

Косова А. Е.

« » 2023 г.

Томск 2023

Оглавление

[1 Цель и Задачи 3](#_Toc136900793)

[2 Теория 4](#_Toc136900794)

[3.Листинг программы 6](#_Toc136900795)

[4.Пример решения 13](#_Toc136900796)

[5.Вывод 15](#_Toc136900797)

# 1 Цель и Задачи

В данной практической работе необходимо реализовать один из трех обязательных точных методов (в зависимости от номера варианта):

1. Метод Гаусса;

2. Метод декомпозиции;

3. Метод ортогонализации (схема №1).

Дополнительно можно реализовать еще один итерационный метод – Зейделя или простой итерации.

При помощи данных методов необходимо реализовать решение следующих задач:

1. Решение СЛАУ.

2. Поиск определителя матрицы (только для методов Гаусса и декомпозиции).

3. Поиск обратной матрицы.

# 2 Теория

Метод Декомпозиции

Сначала исходная матрица A раскладывается на две треугольные матрицы B и C таким образом, что A = BC. Формулы для получения элементов матриц B и C:

Диагональные элементы матрицы C равны 1, остальные элементы матриц B и C нулевые:

Важен порядок вычисления элементов матриц B и C. Сначала вычисляется первый столбец матрицы B, затем первая строка матрицы C, затем второй столбец B, затем вторая строка C и т.д. После этого сначала решается СЛАУ By = d, а затем – СЛАУ Cx = y. По аналогии для решения этих систем можно записать

Определитель исходной матрицы A можно вычислить по формуле

Метод простой итерации

Преобразуем исходную систему к виду

x = β + αx,

где α – матрица размера n×n, β – вектор размера n:

Полагая в качестве начального приближения решения = β, строим итерационный процесс по формулам

Итерации заканчиваются, когда выполняется условие

где ε – требуемая точность решения.

# 3.Листинг программы

using System;

using System.IO;

namespace ЧМ\_ЛБ\_2

{

class Program

{

static double[] VecNev(double[,] \_MatrixA, double[] \_Vector, double[] \_x, int n)

{

double[] VectorEps = new double[n];

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

{

VectorEps[i] = 0;

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

{

VectorEps[i] += \_MatrixA[i, j] \* \_x[j];

}

VectorEps[i] = \_Vector[i] - VectorEps[i];

}

return VectorEps;

}

static double[,] MatNev(double[,] \_MatrixA, double[,] \_MatrixE, double[,] \_MatrixX, int n)

{

double[,] MatrixEps = new double[n, n];

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

{

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

{

MatrixEps[i, j] = 0;

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

{

MatrixEps[i, j] += \_MatrixA[i, k] \* \_MatrixX[k, j];

}

MatrixEps[i, j] = MatrixEps[i, j] - \_MatrixE[i, j];

}

}

return MatrixEps;

}

static double NormVecNev(double[] \_VectorEps, int n)

{

double NormalVectorEps = 0;

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

{

NormalVectorEps += \_VectorEps[i] \* \_VectorEps[i];

}

NormalVectorEps = Math.Sqrt(NormalVectorEps);

return NormalVectorEps;

}

static double NormMatNev(double[,] \_MatrixEps, int n)

{

double NormalMatrixEps = 0;

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

{

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

{

NormalMatrixEps += \_MatrixEps[i, j] \* \_MatrixEps[i, j];

}

}

NormalMatrixEps = Math.Sqrt(NormalMatrixEps);

return NormalMatrixEps;

}

static double[] Dec(double[,] \_MatrixB, double[,] \_MatrixC, double[] \_Vector, double[,] \_MatrixE, int n)

{

double[] y = new double[n];

double sum = 0;

y[0] = \_Vector[0] / \_MatrixB[0, 0];

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

{

sum = 0;

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

{

sum += \_MatrixB[i, k] \* y[k];

}

y[i] = (\_Vector[i] - sum) / \_MatrixB[i, i];

}

double[] x = new double[n];

x[n - 1] = y[n - 1];

for (int i = n - 2; i >= 0; i--)

{

sum = 0;

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

{

sum += \_MatrixC[i, k] \* x[k];

}

x[i] = y[i] - sum;

}

return x;

}

static double[] Iter(double[,] \_MatrixA, double[] \_Vector, int n)

{

double eps = 0.0001;

double sum = 0;

double[] x = new double[n];

double[] b = new double[n];

double[,] a = new double[n, n];

double[] Xn = new double[n];

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

{

x[i] = \_Vector[i] / \_MatrixA[i, i];

b[i] = \_Vector[i] / \_MatrixA[i, i];

}

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

{

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

{

if (i == j)

{

a[i, i] = 0;

continue;

}

a[i, j] = -(\_MatrixA[i, j] / \_MatrixA[i, i]);

}

}

do

{

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

{

sum = 0;

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

{

sum += a[i, j] \* x[j];

}

Xn[i] = sum + b[i];

}

bool flag = true;

for (int i = 0; i < n - 1; i++)

{

if (Math.Abs(Xn[i] - x[i]) > eps)

{

flag = false;

break;

}

}

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

{

x[i] = Xn[i];

}

if (flag)

{

break;

}

} while (true);

return x;

}

static void Main(string[] args)

{

FileStream fileIn = new FileStream("IN.txt", FileMode.Open);

StreamReader reader = new StreamReader(fileIn);

int n = Convert.ToInt32(reader.ReadLine());

double[,] MatrixA = new double[n, n];//Входныя матрица

double[,] MatrixB = new double[n, n];//Нижний треугольник

double[,] MatrixC = new double[n, n];//Верхний треугольник

double[,] MatrixX = new double[n, n];//Обратная матрица

double[,] MatrixY = new double[n, n];

double[,] MatrixE = new double[n, n];//Еденичная матрица

double[] Vector = new double[n];//Входной вектор

double sum = 0;

double[] x = new double[n];//Вектор x

double[] y = new double[n];//Вектор y

//Заполняем входные матрицу и вектор

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

{

string s = reader.ReadLine();

string[] s1 = s.Split(' ');

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

{

MatrixA[i, j] = Convert.ToDouble(s1[j]);

Console.Write(" {0}", MatrixA[i, j]);

MatrixB[i, j] = 0;

MatrixC[i, j] = 0;

if (i == j)

{

MatrixC[i, j] = 1;

MatrixE[i, j] = 1;

}

}

Vector[i] = Convert.ToDouble(s1[n]);

Console.Write(" {0}", Vector[i]);

Console.WriteLine(" ");

}

Console.WriteLine(" ");

reader.Close();

fileIn.Close();

Console.WriteLine("1-Метод Декомпозиции");

Console.WriteLine("2-Метод Простой итерации");

int m = Convert.ToInt32(Console.ReadLine());

switch (m)

{

case 1:

//находим первый столбец B и первую строку C

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

{

MatrixB[i, 0] = MatrixA[i, 0];

MatrixC[0, i] = MatrixA[0, i] / MatrixB[0, 0];

}

//Вычисляем B и C по формуле

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

{

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

{

if (i >= j)//нижний треугольник

{

sum = 0;

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

{

sum += MatrixB[i, k] \* MatrixC[k, j];

}

MatrixB[i, j] = MatrixA[i, j] - sum;

}

else//верхний

{

sum = 0;

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

{

sum += MatrixB[i, k] \* MatrixC[k, j];

}

MatrixC[i, j] = (MatrixA[i, j] - sum) / MatrixB[i, i];

}

}

}

//Нахождение определителя

double det = 1;

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

{

det \*= MatrixB[i, i];

}

//вычисляем x и y

y[0] = Vector[0] / MatrixB[0, 0];

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

{

sum = 0;

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

{

sum += MatrixB[i, k] \* y[k];

}

y[i] = (Vector[i] - sum) / MatrixB[i, i];

}

Console.WriteLine();

x[n - 1] = Math.Round(y[n - 1],3);

for (int i = n - 2; i >= 0; i--)

{

sum = 0;

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

{

sum += MatrixC[i, k] \* x[k];

}

x[i] = Math.Round(y[i] - sum,3);

}

//Обратная матрица

double[] buf = new double[n];

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

{

//Копируем первую строчку

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

{

buf[j] = MatrixE[j, i];

}

//Передаем в итерационный метод

buf = Dec(MatrixB, MatrixC, buf, MatrixE, n);

//Заполняем обратную матрицу

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

{

MatrixX[j, i] = Math.Round(buf[j], 3);

}

}

//Матрица невязки

double[,] MatrixEps = new double[n, n];

MatrixEps = MatNev(MatrixA, MatrixE, MatrixX, n);

//Норма матрицы невязки

double NormalMatrixEps = NormMatNev(MatrixEps, n);

//Вектор невязки

double[] VectorEps = new double[n];

VectorEps = VecNev(MatrixA, Vector, x, n);

//Норма вектора невязки

double NormalVectorEps = NormVecNev(VectorEps, n);

//записываем результат

FileStream fileOut1 = new FileStream("OUT.txt", FileMode.Create);

StreamWriter writer1 = new StreamWriter(fileOut1);

writer1.WriteLine("Матрица B");

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

{

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

{

writer1.Write("{0} ", Math.Round(MatrixB[i, j], 3));

}

writer1.WriteLine();

}

writer1.WriteLine();

writer1.WriteLine("Матрица C");

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

{

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

{

writer1.Write("{0} ", Math.Round(MatrixC[i, j], 3));

}

writer1.WriteLine();

}

writer1.WriteLine();

writer1.Write("Вектор y: ");

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

{

writer1.Write("y{0}= {1}, ", i + 1, Math.Round(y[i], 3));

}

writer1.WriteLine();

writer1.Write("Вектор x: ");

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

{

writer1.Write("x{0}= {1}, ", i + 1, Math.Round(x[i], 3));

}

writer1.WriteLine();

writer1.Write("Вектор невязки:");

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

{

writer1.Write("e{0}= {1}, ", i + 1, VectorEps[i].ToString("e4"));

}

writer1.WriteLine();

writer1.WriteLine("Норма вектора невязки = {0}", NormalVectorEps.ToString("e4"));

writer1.WriteLine();

writer1.WriteLine("Определитель матрицы= {0}", det);

writer1.WriteLine();

writer1.WriteLine("Обратная матрица:");

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

{

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

{

writer1.Write("{0} ", MatrixX[i, j]);

}

writer1.WriteLine();

}

writer1.WriteLine();

writer1.WriteLine("Матрица невязки:");

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

{

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

{

writer1.Write("{0} ", MatrixEps[i, j].ToString("e4"));

}

writer1.WriteLine();

}

writer1.WriteLine();

writer1.WriteLine("Норма матрицы невязки = {0}", NormalMatrixEps.ToString("e4"));

writer1.WriteLine();

writer1.Close();

fileOut1.Close();

break;

case 2:

double[,] a = new double[n, n];

double[] b = new double[n];

x = Iter(MatrixA, Vector, n);

//Преобразование матрицы a

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

{

b[i] = Vector[i] / MatrixA[i, i];

}

//Преобразование вектора b

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

{

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

{

if (i == j)

{

a[i, i] = 0;

continue;

}

a[i, j] = -(MatrixA[i, j] / MatrixA[i, i]);

Math.Round(a[i, j], 3);

}

}

//Вектор невязки

VectorEps = VecNev(MatrixA, Vector, x, n);

//Норма вектора невязки

NormalVectorEps = NormVecNev(VectorEps, n);

//Обратная матрица

buf = new double[n];

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

{

//Копируем первую строчку

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

{

buf[j] = MatrixE[j, i];

}

//Передаем в итерационный метод

buf = Iter(MatrixA, buf, n);

//Заполняем обратную матрицу

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

{

MatrixX[j, i] = Math.Round(buf[j], 3);

}

}

//Матрица невязки

MatrixEps = MatNev(MatrixA, MatrixE, MatrixX, n);

//Норма матрицы невязки

NormalMatrixEps = NormMatNev(MatrixEps, n);

FileStream fileOut2 = new FileStream("OUT.txt", FileMode.Create);

StreamWriter writer2 = new StreamWriter(fileOut2);

writer2.WriteLine("Матрица a:");

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

{

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

{

writer2.Write("{0} ", Math.Round(a[i, j], 3));

}

writer2.WriteLine();

}

writer2.WriteLine();

writer2.Write("Вектор b: ");

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

{

writer2.Write("b{0}={1}, ", i + 1, Math.Round(b[i], 3));

}

writer2.WriteLine();

writer2.Write("Вектор x: ");

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

{

writer2.Write("x{0}={1}, ", i + 1, Math.Round(x[i], 3));

}

writer2.WriteLine();

writer2.Write("Вектор невязки: ");

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

{

writer2.Write("e{0}={1}, ", i + 1, VectorEps[i].ToString("e4"));

}

writer2.WriteLine();

writer2.WriteLine("Норма вектора невязки = {0}", NormalVectorEps.ToString("e4"));

writer2.WriteLine();

writer2.WriteLine("Обратная матрица:");

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

{

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

{

writer2.Write("{0} ", MatrixX[i, j]);

}

writer2.WriteLine();

}

writer2.WriteLine();

writer2.WriteLine("Матрица невязки:");

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

{

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

{

writer2.Write("{0} ", MatrixEps[i, j].ToString("e4"));

}

writer2.WriteLine();

}

writer2.WriteLine();

writer2.WriteLine("Норма матрицы невязки = {0}", NormalMatrixEps.ToString("e4"));

writer2.WriteLine();

writer2.Close();

fileOut2.Close();

break;

}

Console.ReadKey();

}

# 4.Пример решения

Входной файл IN.txt для обоих методов на рисунке 4.1.

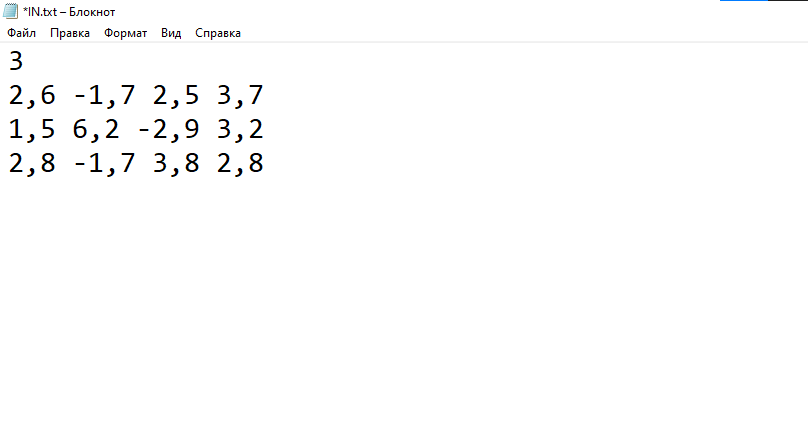


Рисунок 4.1— Входные данные из файла

Результат работы программы, записанный в файл OUT.txt при использовании метода Декомпозиции на рисунке 4.2

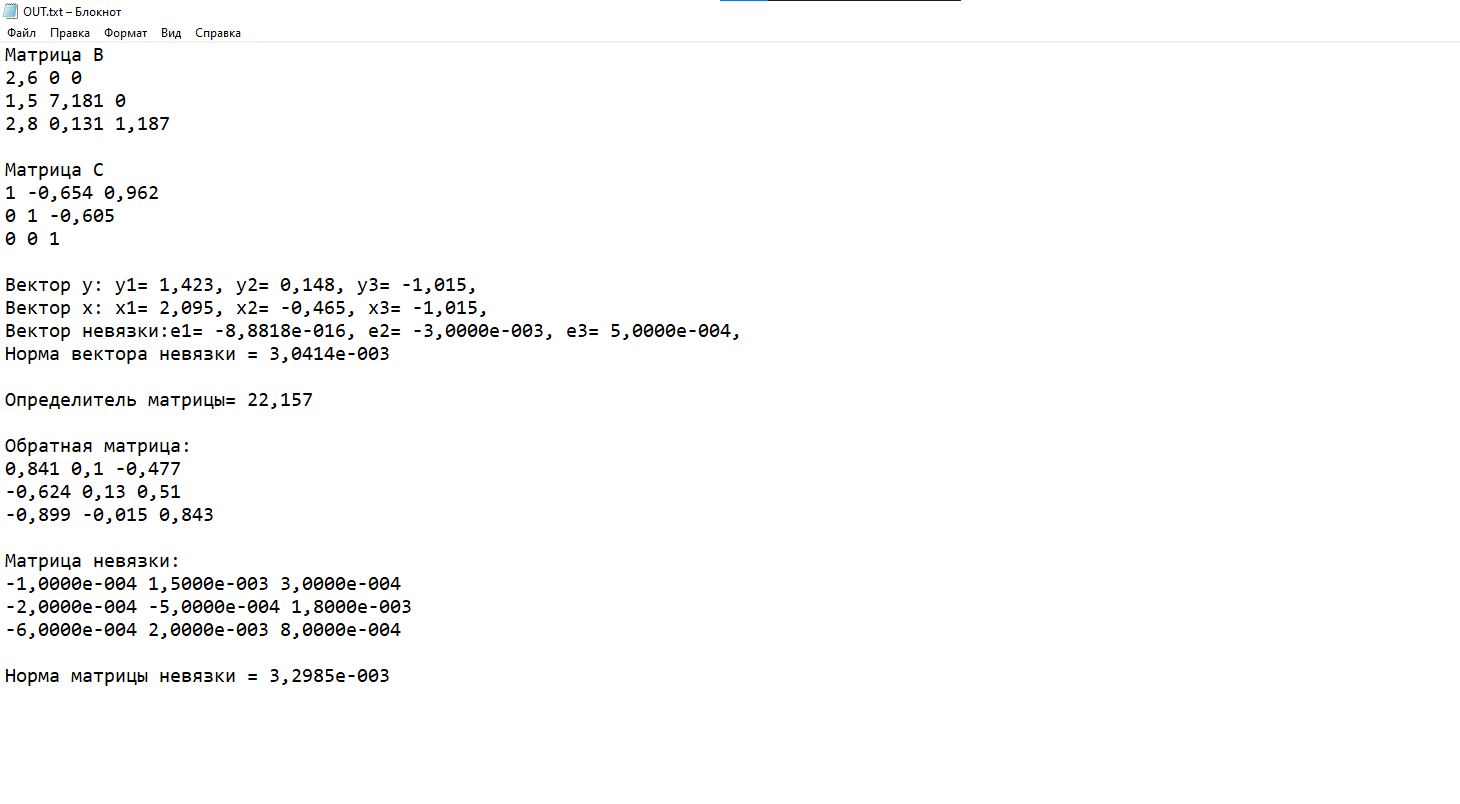


Рисунок 4.2 —Результат работы программы метода декомпозиции

Результат работы программы при использовании метода простой итерации на рисунке 4.3

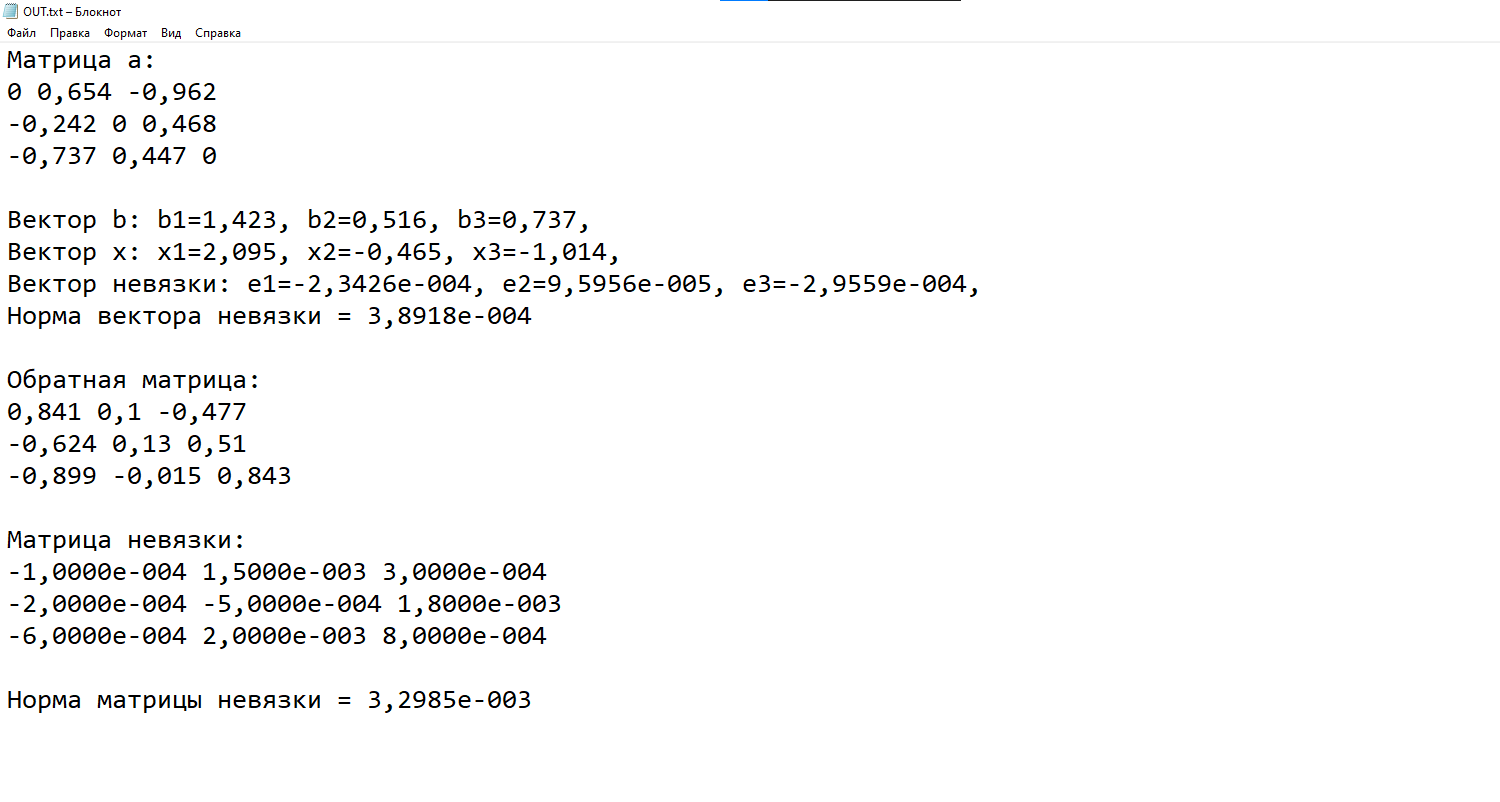


Рисунок 4.3 —Результат работы программы метода простой итерации

# 5.Вывод

В результате лабораторной работы были изучены методы решения систем линейных алгебраических уравнений (СЛАУ) и вычисление различных характеристик матриц – определителей, обратных матриц и собственных чисел, и векторов. Были изучены такие методы как метод декомпозиции и простой итерации.