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 Проскурин А. В.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

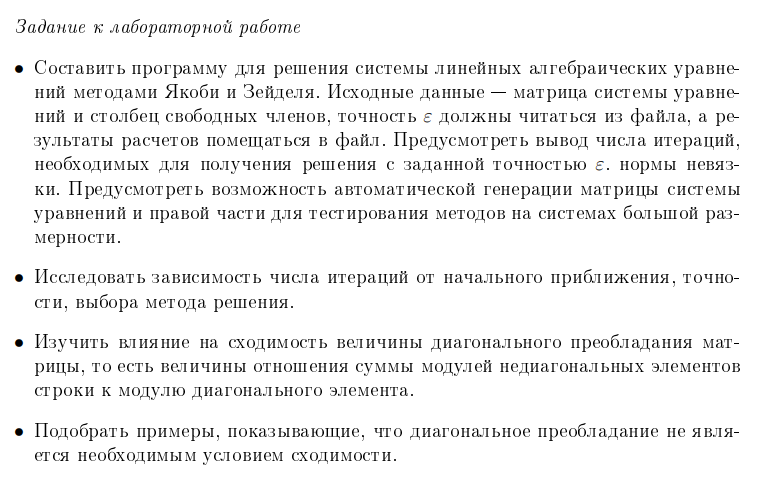
Отчет

Лабораторная работа №2

Решение систем линейных уравнений методами Якоби и Зейделя

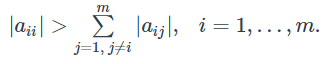
по дисциплине «Вычислительные алгоритмы»

Студент группы ПИ 92 Д. А. Савиков



Алгоритм решения

Сначала производится проверка достаточного условия для методов Якоби и Зейделя. Суть условия в том, что диагональное преобладание матрицы A должно быть меньше 1, то есть сумма модулей не диагональных элементов строки должна быть меньше модуля диагонального элемента строки в каждой строке.

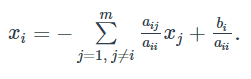


Метод Якоби

Поиск решения происходит до тех пор пока выполняется условие

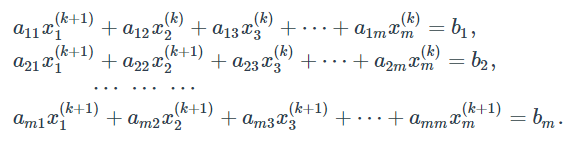
где x^(k+1) и x^(k) – решение системы на k+1 и k итерации соответственно, а e – точность.

В каждой итерации мы в каждой строке системы поставляем вместо X решение системы с прошлой итерации (в первой итерации подставляем первоначальное приближение), кроме диагональных элементов. После чего из получившихся уравнений находим x на главной диагонали.



Метод Зейделя

Аналогичен методу Якоби, кроме того что в каждой итерации мы подставляем X не с прошлой итерации, а в каждой строке вместо всех x до диагонального мы поставляем x с данной итерации, а все x после диагонального заменяем на x с прошлой итерации.



Текст программы

using System;

using System.Collections.Generic;

using System.IO;

namespace AlgLab2

{

class Program

{

//Проверка достаточного условия для решения системы

static bool Check(List<List<double>> A)

{

double test;

string r;

for (int i = 0; i < A.Count; i++) //Проверка достаточного условия сходимости

{

test = 0; //Поиск суммы модулей всех элементов строки, кроме элемента главной диагонали

for (int j = 0; j < A[i].Count; j++)

{

if (j != i)

{

test = test + Math.Abs(A[i][j]);

}

}

if (Math.Abs(test) > Math.Abs(A[i][i])) //Сравнение суммы и элемента главной диагонали этой строки

{

Console.WriteLine("Не выполняется достаточное условие. Продолжить? (y - да, n - нет)");

r = Console.ReadLine();

if (r == "y")

{

return true;

}

else return false;

}

}

return true;

}

//Поиск нормы одномерной матрицы

static double Norm(List<double> X)

{

double max;

max = 0;

for(int i = 0; i < X.Count; i++)

{

if (Math.Abs(X[i]) > Math.Abs(max)) max = X[i];

}

return Math.Abs(max);

}

//Проверка условия ||X^(k+1) - X^k|| < e

static bool IterationCheck(double e, List<double> X1, List<double> X2)

{

List<double> XNorm = new List<double>();

for(int i = 0; i < X1.Count; i++) //Разность матриц

{

XNorm.Add(X1[i] - X2[i]);

}

return e <= Norm(XNorm); //Возвращение результата сравнения

}

//Метод Якоби

static int Funk(List<List<double>> A, List<double> B, List<double> X, double e)

{

double mid;

int z;

z = 0;

List<double> XPast = new List<double>(); //X в предидущей итерации

do //Решение

{

XPast = new List<double>(X); //Сохранение предыдущей итерации

for (int i = 0; i < X.Count; i++) //Проход по всем строкам и поиск X[i]

{

mid = 0;

for (int j = 0; j < A[i].Count; j++) //Сумма всех не диагональных элементов уравнения

{

if (j != i)

{

mid = mid + A[i][j] \* XPast[j];

}

}

mid = B[i] - mid; //Разность свобочного члена и суммы не диагональных элементов

X[i] = mid / A[i][i]; //Деление на элемент A находящийся на данной строке и главной диагонали

}

z++; //Счетчик итераций

} while (IterationCheck(e, X, XPast)); //Проверка условия продолжения итераций

return z;

}

//Метод Зейделя (анологичен методу Якоби, кроме одной закоментированной строки)

static int Funk1(List<List<double>> A, List<double> B, List<double> X, double e)

{

double mid;

int z;

z = 0;

List<double> XPast = new List<double>();

do

{

XPast = new List<double>(X);

for (int i = 0; i < X.Count; i++)

{

mid = 0;

for (int j = 0; j < A[i].Count; j++)

{

if (j != i)

{

mid = mid + A[i][j] \* X[j]; //A[i][i] умножается не на XPast а на X. В X сохранены все измененные к этому моменту в этой итерации x, а все остальные x с прошлой итерации

}

}

mid = B[i] - mid;

X[i] = mid / A[i][i];

}

z++;

} while (IterationCheck(e, X, XPast));

return z;

}

//Норма невязки

static double NE(List<List<double>> A, List<double> B, List<double> X)

{

double a;

List<double> r = new List<double>();

for (int i = 0; i < A.Count; i++) //Проход по всем строкам

{

a = 0; //Поиск суммы левой части уравнения

for (int j = 0; j < A[0].Count; j++)

{

a = a + A[i][j] \* X[j];

}

r.Add(B[i] - a); //Вычитание этой суммы из правой части и сохранение результата

}

return Norm(r); //Возвращение нормы невязки

}

static void Main(string[] args)

{

string file, string1;

string[] string2;

List<List<double>> A = new List<List<double>>();

List<double> B = new List<double>();

List<double> X = new List<double>();

List<double> X1 = new List<double>();

Random rand = new Random();

double e, ne1, ne2;

int it1, it2, size, mid, left, p;

e = 1;

Console.WriteLine("Введите название файла (оставте пустым чтобы сгенерировать систему уравнений)");

file = Console.ReadLine();

if(file == "")

{

try

{

Console.WriteLine("Введите название файла для сохранения решения");

file = Console.ReadLine();

Console.WriteLine("Введите размер матрицы A");

size = Convert.ToInt32(Console.ReadLine());

StreamWriter stream1 = new StreamWriter(file);

for (int i = 0; i < size; i++) //Создание системы уравнений удовлетворяющих достаточному условию

{

mid = rand.Next(100);

left = mid;

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

{

if(j != i)

{

p = rand.Next(left);

stream1.Write(Convert.ToString(p).PadLeft(4) + " ");

left = left - p;

}

else stream1.Write(Convert.ToString(mid).PadLeft(4) + " ");

}

p = rand.Next(100);

stream1.Write(Convert.ToString(p).PadLeft(4) + " ");

stream1.WriteLine();

}

Console.WriteLine("Введите точность");

e = Convert.ToDouble(Console.ReadLine());

stream1.WriteLine("Точность");

stream1.WriteLine(e);

stream1.WriteLine("Начальное приближение");

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

{

e = rand.Next(100);

stream1.Write(Convert.ToString(e).PadLeft(4) + " ");

}

stream1.Close();

}

catch

{

Console.WriteLine("Ошибка");

return;

}

}

if (!File.Exists(file))

{

Console.WriteLine("Файл не найден");

return;

}

else

{

try //Запись матрицы в массив

{

StreamReader stream = new StreamReader(file);

string1 = stream.ReadLine();

string2 = string1.Split(' ');

while (Double.TryParse(string2[0], out e) || string2[0] == "")

{

A.Add(new List<double>());

foreach (var num in string2)

{

if (num != "")

{

A[A.Count - 1].Add(Convert.ToDouble(num));

}

}

B.Add(A[A.Count - 1][A[A.Count - 1].Count - 1]);

A[A.Count - 1].RemoveAt(A[A.Count - 1].Count - 1);

string1 = stream.ReadLine();

string2 = string1.Split(' ');

}

string1 = stream.ReadLine();

e = Convert.ToDouble(string1);

string1 = stream.ReadLine();

string1 = stream.ReadLine();

string2 = string1.Split(' ');

foreach (var num in string2)

{

if (num != "")

{

X.Add(Convert.ToDouble(num));

X1.Add(Convert.ToDouble(num));

}

}

stream.Close();

}

catch

{

Console.WriteLine("Данные введены не правильно");

return;

}

if (Check(A)) //Проверка достаточного условия

{

it1 = Funk(A, B, X, e);

it2 = Funk1(A, B, X1, e);

ne1 = NE(A, B, X);

ne2 = NE(A, B, X1);

StreamWriter stream1 = File.AppendText(file);

stream1.WriteLine();

stream1.WriteLine("Результат");

stream1.WriteLine("Метод Якоби");

Console.WriteLine();

Console.WriteLine("Результат");

Console.WriteLine("Метод Якоби");

for (int i = 0; i < X.Count; i++)

{

stream1.WriteLine("x" + i + " = " + X[i]);

Console.WriteLine("x" + i + " = " + X[i]);

}

stream1.WriteLine("Норма невязки = " + ne1);

Console.WriteLine("Норма невязки = " + ne1);

stream1.WriteLine("Кол-во итераций = " + it1);

Console.WriteLine("Кол-во итераций = " + it1);

stream1.WriteLine();

stream1.WriteLine("Метод Зейделя");

Console.WriteLine();

Console.WriteLine("Метод Зейделя");

for (int i = 0; i < X.Count; i++)

{

stream1.WriteLine("x" + i + " = " + X1[i]);

Console.WriteLine("x" + i + " = " + X1[i]);

}

stream1.WriteLine("Норма невязки = " + ne2);

Console.WriteLine("Норма невязки = " + ne2);

stream1.WriteLine("Кол-во итераций = " + it2);

Console.WriteLine("Кол-во итераций = " + it2);

stream1.Close();

}

else

{

Console.WriteLine("Решение не сходится");

return;

}

}

}

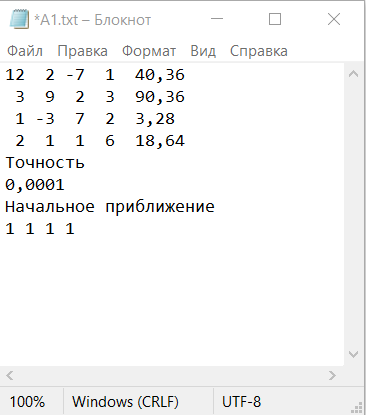
}

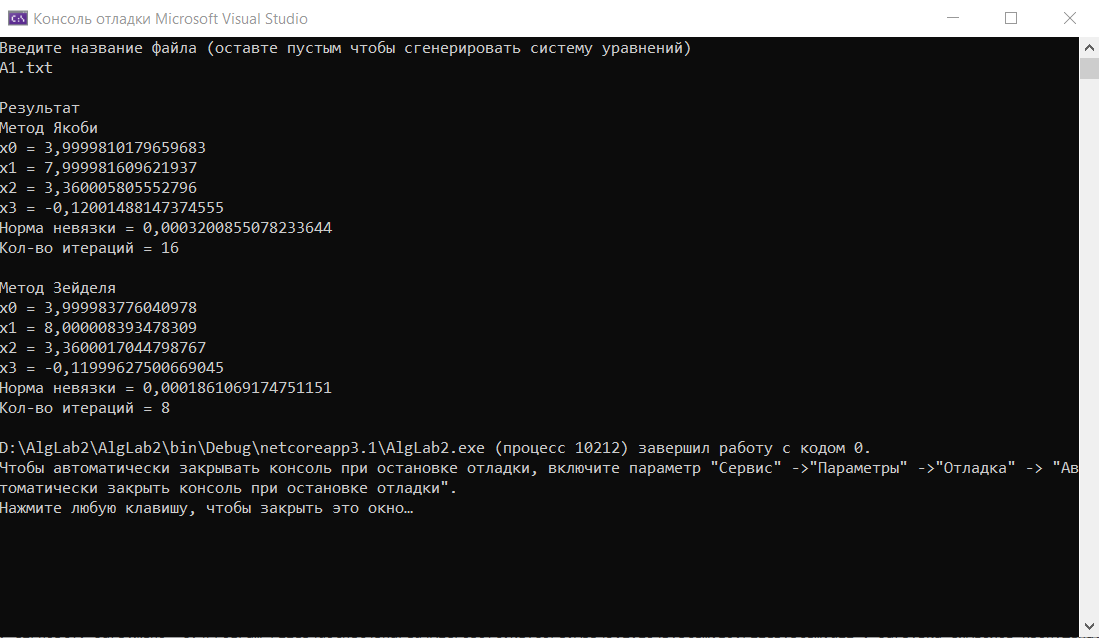
}

Тест программы

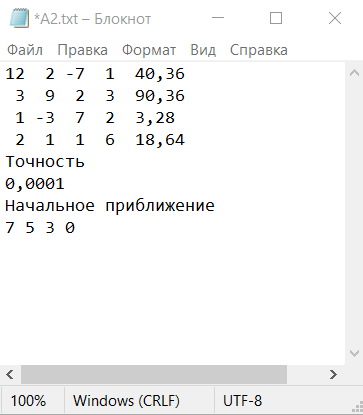
1)Проверка влияния начального приближения.

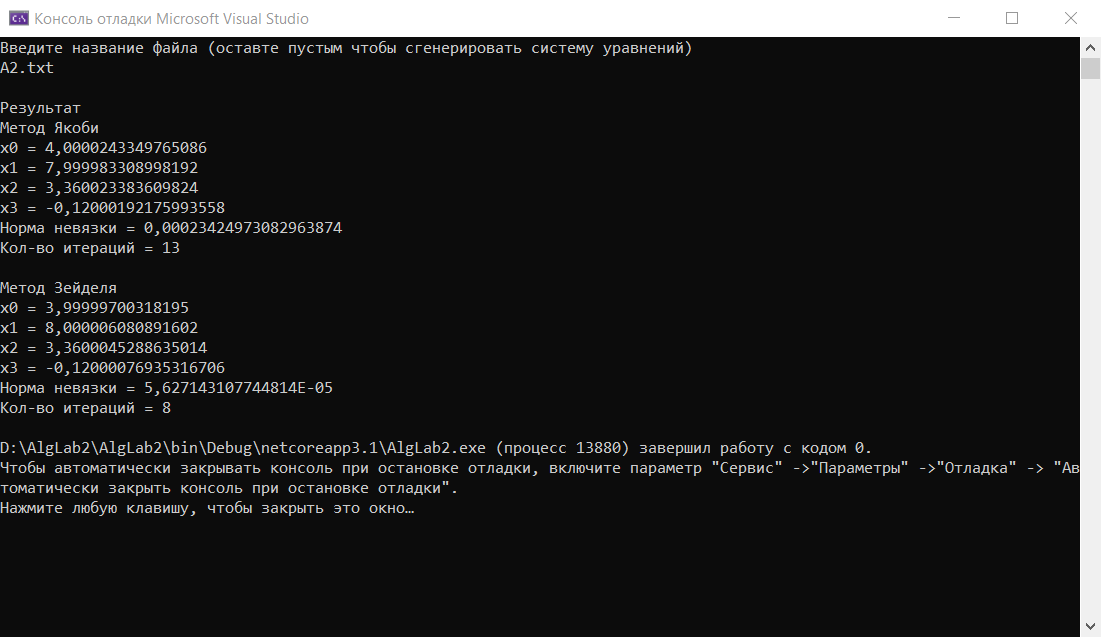
Система уравнений в каждом тесте одинаковая.



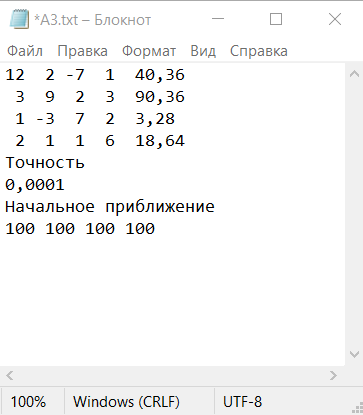


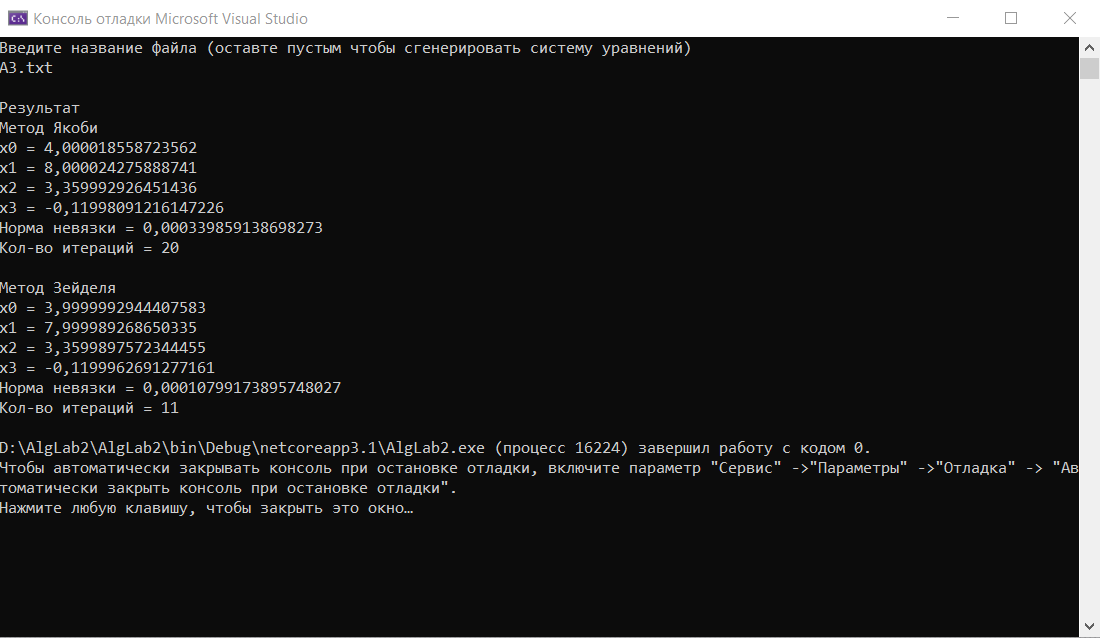
Теперь начальное приближение очень близко к решению системы.



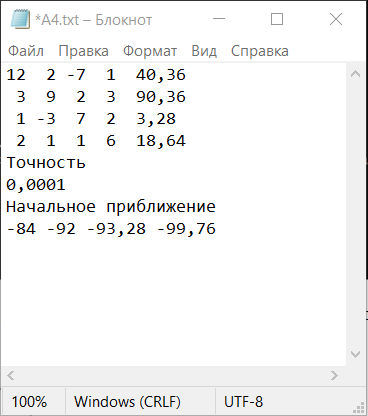


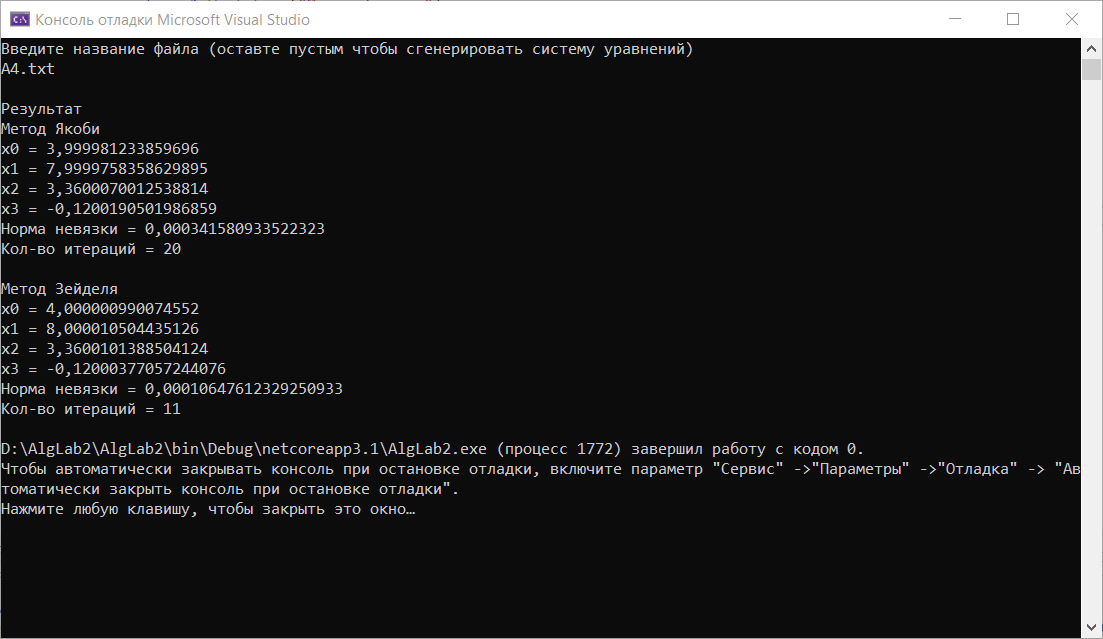
Начальное приближение очень далеко от решения системы.





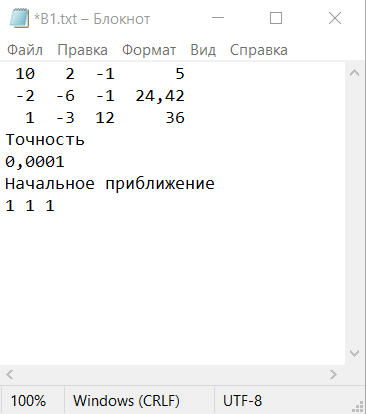
Начальное приближение настолько же далеко от решения системы как и в прошлом тесте, но теперь оно отрицательное.

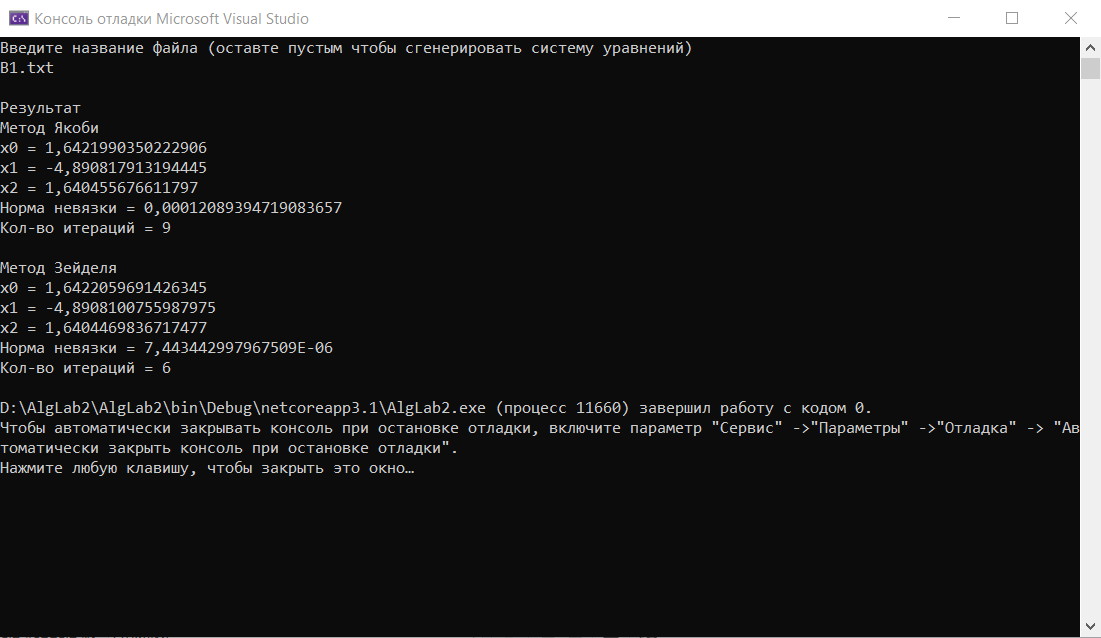


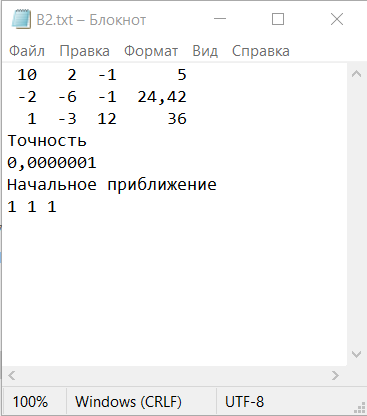


Итог: чем ближе начальное приближение к решению системы, тем меньше необходимо итераций.

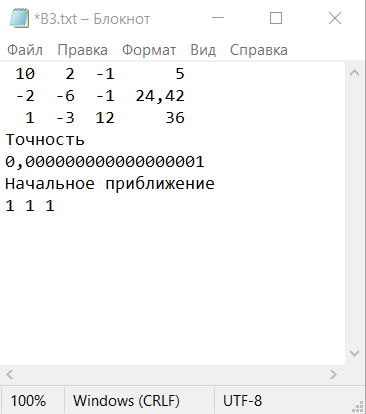
2)Проверка влияния точности.











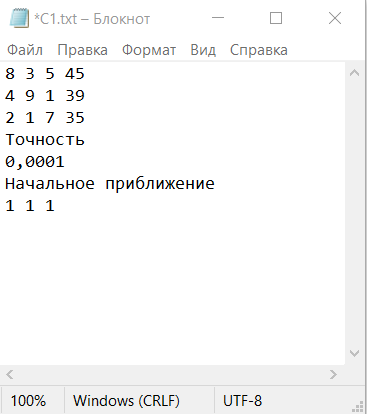


Итог: чем ниже требуемая точность, тем больше требуется итераций.

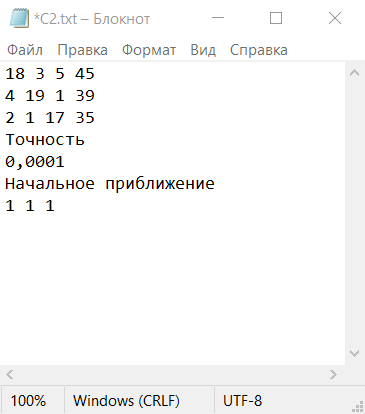
3)Сравнение методов: метод Зейделя требует меньше итераций и изменение точности и начального приближения влияет на него в меньшей степени.

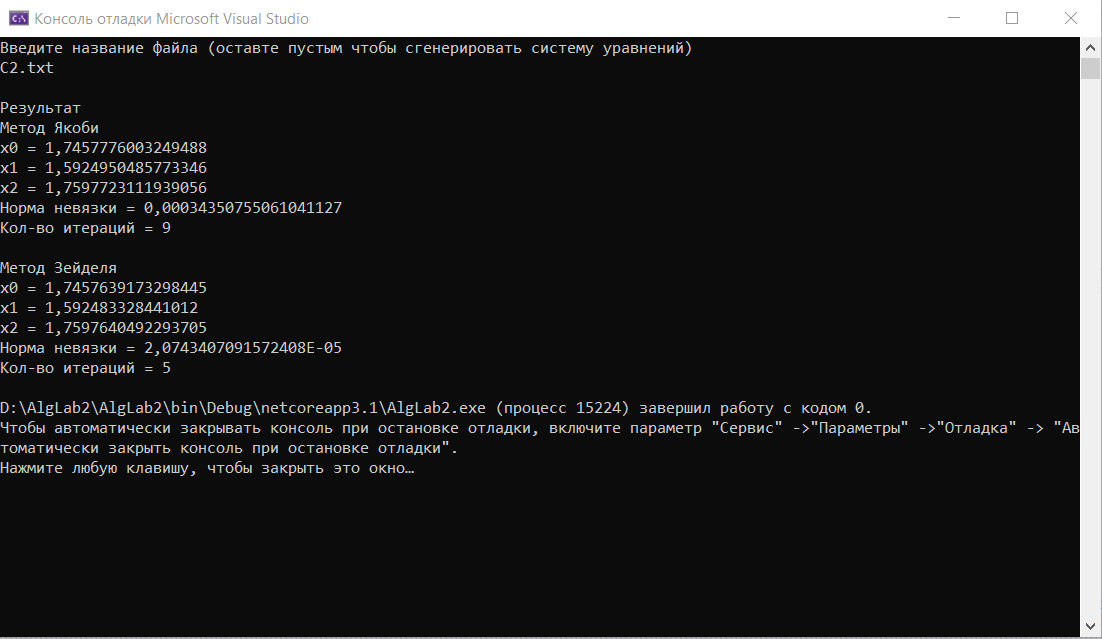
4)Проверка влияния диагонального преобладания.

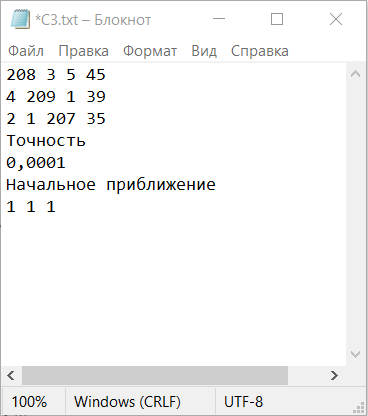
В каждом тесте увеличиваются только элементы на главной диагонали.







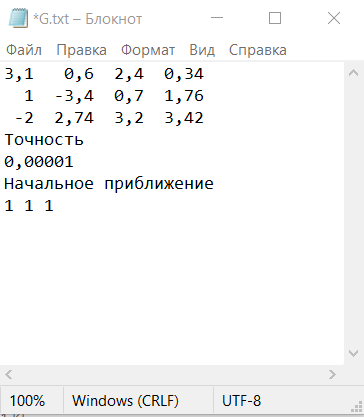


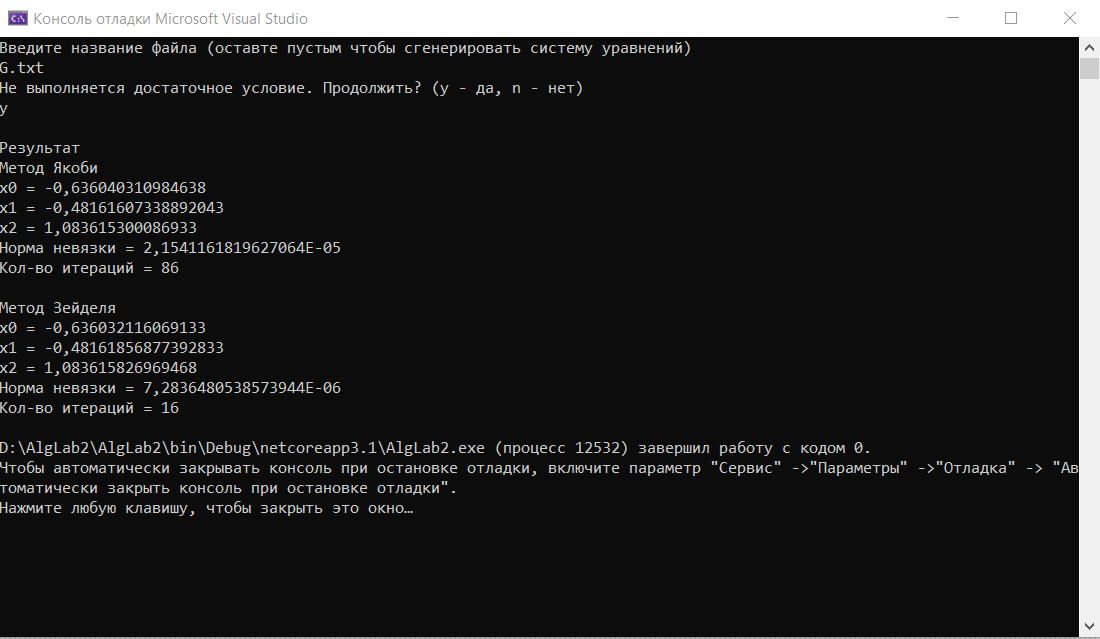




Итог: чем выше диагональное преобладание, тем меньше итераций требуется

5)Достаточное условие не выполняется, но решение все равно сходится.





Итог: диагональное преобладание не является необходимым условием сходимости решения.

Вывод: была создана программа вычисляющая решение системы уравнений методами Якоби и Зейделя. Чем ближе начальное приближение к решению системы, тем меньше итераций требуется. Чем ниже требуемая точность, тем больше итераций требуется. Метод Зейделя работает быстрее метода Якоби. Диагональное преобладание не является необходимым условием сходимости решения и чем оно выше, тем меньше итераций требуется.