БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №3

Решение СЛАУ методом простых итераций, методом Якоби, методом Зейделя, методом Гаусса-Зейделя

Выполнил:

Студент 2 курса 5 группы ФПМИ

Дунаев Виктор

Руководитель:

Радкевич Елена Владимировна

Минск, 2016 г.

**Оглавление**

Постановка задачи……………………………………………….……………......…..3

Метод простой итерации…………………………………………………….………..3

Метод Якоби………………………………..…………………………………..…...…5

Метод Зейделя …………………………………………………………………..…….5

Метод Гаусса-Зейделя…………………………………………………………..…….7

Листинг программы …………………………………………………….…………….7

Входные данные ……………………………………………………………….…….17

Выходные данные ………………………………………….………………………..17

**1.Постановка задачи**

Дана СЛАУ, где расширенная матрица этой системы имеет следующий вид:

**А b**

0.4997 -0.0658 0.0132 0.0263 0.0921 -2.8141

0.0684 0.7824 0.0000 -0.0526 0.0526 2.4104

0.0395 0.0000 0.6286 -0.1841 0.1052 2.2828

-0.0789 0.1657 0.0000 0.6181 -0.0263 -1.6332

0.3288 0.0000 0.1184 0.0132 0.7364 1.8936

1) Найти решения системы при помощи метода простой итерации, метода Якоби, метода Зейделя, метода Гаусса-Зейделя с точностью Е=10^-5.

2) Вывести на печать:

А) Исходную матрицу A и вектор b;

Б) Матрицу B и вектор G;

В) Результаты всех методов;

Г) Количество итерация для всех методов;

Д) Вектора невязки для всех методов;

Е) Априорное количество итераций для методов Якоби и Гаусса-Зейделя.

**2.Метод простой итерации**

Простейшим итерационным методом для решения СЛАУ является метод простой итерации (МПИ), основным недостатком которого является его низкя скорость сходимости. Здесь исходная СЛАУ преобразуется к виду

http://ok-t.ru/studopediaru/baza3/351588547776.files/image031.gif(2)

где http://ok-t.ru/studopediaru/baza3/351588547776.files/image033.gif- http://ok-t.ru/studopediaru/baza3/351588547776.files/image006.gifматрица, http://ok-t.ru/studopediaru/baza3/351588547776.files/image036.gif- вектор длины http://ok-t.ru/studopediaru/baza3/351588547776.files/image010.gif, а решение СЛАУ (2) находится как предел последовательности

http://ok-t.ru/studopediaru/baza3/351588547776.files/image039.gif(3)

Ключевым вопросом при изучении итерационных методов является вопрос их сходимости (к точному решению СЛАУ). Надо выяснить, каким условиям должна удовлетворять матрица http://ok-t.ru/studopediaru/baza3/351588547776.files/image033.gifв (2) и (3), чтобы итерационный процесс (3) сходился.

**Лемма 1**. Для того, чтобы матричный ряд

http://ok-t.ru/studopediaru/baza3/351588547776.files/image041.gif

где http://ok-t.ru/studopediaru/baza3/351588547776.files/image043.gif- единичная http://ok-t.ru/studopediaru/baza3/351588547776.files/image006.gifматрица, сходился, необходимо и достаточно, чтобы все собственные значения (СЗ) http://ok-t.ru/studopediaru/baza3/351588547776.files/image006.gifматрицы http://ok-t.ru/studopediaru/baza3/351588547776.files/image033.gifпо модулю были меньше 1:

http://ok-t.ru/studopediaru/baza3/351588547776.files/image045.gif.

При выполнении этого условия матрица http://ok-t.ru/studopediaru/baza3/351588547776.files/image047.gifимеет обратную, и справедливо равенство:

http://ok-t.ru/studopediaru/baza3/351588547776.files/image049.gif

**Лемма 2.** Для того, чтобы http://ok-t.ru/studopediaru/baza3/351588547776.files/image051.gifнеобходимо и достаточно, чтобы чтобы все СЗ матрицы http://ok-t.ru/studopediaru/baza3/351588547776.files/image033.gifпо модулю были меньше 1.

**Лемма 3**. Модули собственных значений матрицы не превосходят любую из ее норм.

**Теорема 1 (критерий сходимости МПИ)**. Для того, чтобы векторная последовательность http://ok-t.ru/studopediaru/baza3/351588547776.files/image020.gif, построенная по формуле (3), сходилась при любом начальном приближении http://ok-t.ru/studopediaru/baza3/351588547776.files/image014.gifк решению СЛАУ (1) (и (2)), необходимо и достаточно, чтобы все СЗ матрицы http://ok-t.ru/studopediaru/baza3/351588547776.files/image033.gifпо модулю были меньше 1.

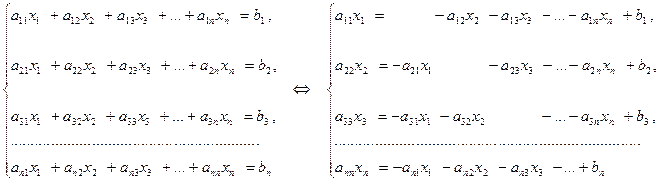
**Теорема 2**. Для того, чтобы векторная последовательность http://ok-t.ru/studopediaru/baza3/351588547776.files/image020.gif, построенная по формуле (3), сходилась к решению СЛАУ (1) (и (2)) http://ok-t.ru/studopediaru/baza3/351588547776.files/image055.gif, достаточно, чтобы какая-либо норма матрицы http://ok-t.ru/studopediaru/baza3/351588547776.files/image033.gifбыла меньше 1:

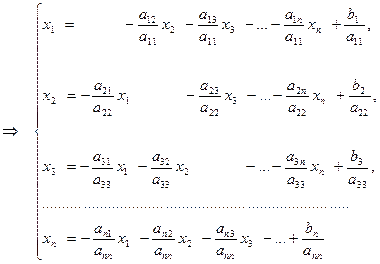
http://ok-t.ru/studopediaru/baza3/351588547776.files/image083.gif.

Пусть матрица http://ok-t.ru/studopediaru/baza3/351588547776.files/image004.gifСЛАУ (1) имеет диагональное преобладание по строкам, т.е. для ее элементов выполняются соотношения:

http://ok-t.ru/studopediaru/baza3/351588547776.files/image096.gif.

Тогда переход от вида (1) к виду (2) СЛАУ, определяющему сходящийся МПИ, можно провести следующим образом:





**3.Метод Якоби**



Запишем исходную систему в каноническом виде:



Данную систему можно записать в виде , где

, .

Метод простых итераций с приведёнными матрицей  и вектором  - метод Якоби.

**Достаточный признак сходимости метода Якоби.** Метод Якоби для исходной системы сходится, если матрица  удовлетворяет одному из следующих условий:

1. , 
2. , 
3. 

**4.Метод Зейделя**

Запишем метод простых итераций  в координатной форме.

, , 

Зафиксируем порядок вычисления координат  в порядке увеличения номеров. При вычислении последующих координат будем учитывать полученные ранее уточнения координат.

 - итерационный процесс Зейделя.

, , 

**Достаточный признак сходимости метода Зейделя.** Метод Зейделя сходится, если матрица  удовлетворяет одному из следующих условий:

1.  
2.  

Как можно выбрать матрицу , чтобы итерационный процесс заведомо сходился?











**Критерий сходимости.** Для того чтобы метод простой итерации (в т.ч. метод Зейделя) сходился при любом начальном приближении, необходимо и достаточно, чтобы все собственные значения матрицы  были по модулю меньше единицы.

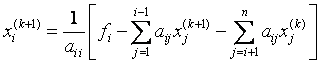
Необходимо выбрать параметр  так, чтобы матрица  удовлетворяла условиям сходимости метода. Если матрица  исходной системы вещественная, положительно определённая и симметрическая, то положив , т.е. , получим заведомо сходящийся итерационный процесс. Тогда собственные значения матрицы  могут быть вычислены по формуле , . Поскольку матрица  положительно определена, , следовательно, , т.е. итерационный процесс вида  будет сходящимся.

Если матрица  не является положительно определённой или симметрической, можно преобразовать систему к виду , где , , , . Тогда итерационный процесс будет иметь вид .

Для обоих методов итерационный процесс останавливается при выполнении условия .

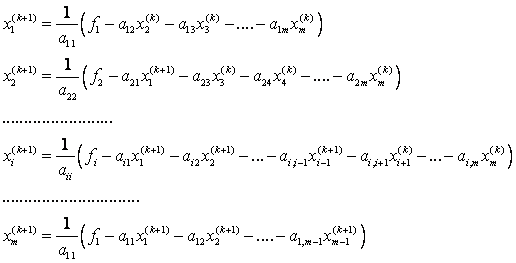
**5.Метод Гаусса-Зейделя**

Расчетные формулы имеют вид:

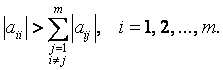


т.е. для подсчета *i*–й компоненты (*k*+1)–го приближения к искомому вектору используется уже вычисленное на этом, т.е. (*k*+1)–м шаге, новые значения первых *i*–1 компонент.

**Подробные формулы имеют вид:**



**Достаточное условие сходимости этого метода такое же, как и для метода простой итерации, т.е. диагональное преобладание:**



**Начальное приближение:**

http://pers.narod.ru/study/methods/2.files/image062.gif

**6.Листинг программы**

**Lab3\_VMA.java**

**Главный класс программы**

public class Lab3\_VMA {

public static void main(String[] args) {

int stop = 6;

NumberFormat formatter = NumberFormat.getNumberInstance();

formatter.setMaximumFractionDigits(stop);

Input I = new Input("input.txt");

Podschet P = new Podschet();

Methods M = new Methods();

double E = 0.00001;

double [][] matr = I.create2XMatrix();

double[] F = I.createVector();

System.out.println("Исходная матрица");

for(int i=0;i<matr.length;i++)

{

for(int j=0;j<matr.length;j++)

{

System.out.print(matr[i][j]+" ");

}

System.out.println();

}

System.out.println("Исходный вектор f");

for(int i=0;i<F.length;i++)

{

System.out.print(F[i]+" ");

}

System.out.println();

double[][] matrT = P.getTranspMatrix(matr);

double[][]ATA = P.getATAMatrix(matr, matrT);

double ATAnorma = P.getMatrixNorma(ATA);

double[][]B=P.getMatrixB(ATA, ATAnorma);

System.out.println("B матрица");

for(int i=0;i<B.length;i++)

{

for(int j=0;j<B.length;j++)

{

System.out.print(formatter.format(B[i][j])+" ");

}

System.out.println();

}

double[]G = P.getVectorG(F, ATAnorma,matr);

System.out.println("Вектор G");

for(int i=0;i<G.length;i++)

{

System.out.print(formatter.format(G[i])+" ");

}

System.out.println();

double[]iterX = M.iterationMethod(B, G, E);

double[]XZeid = M.methodZeidelia(B, G, E);

int max = P.getMaxIter(B, G, E);

System.out.println("Априорное количество итераций для"

+ " методов Якоби и Гаусса-Зейделя = "+ max);

double[] XYakobi = M.methodYakobi(matr, F, E);

double[] XGZ = M.methodGZ(matr, F, E);

System.out.println("Вектор невязки для метода простой итерации:");

P.vectorPrint(P.vectorNev(matr, F, iterX));

System.out.println("Вектор невязки для метода Зейделя:");

P.vectorPrint(P.vectorNev(matr, F, XZeid));

System.out.println("Вектор невязки для метода Якоби:");

P.vectorPrint(P.vectorNev(matr, F, XYakobi));

System.out.println("Вектор невязки для метода Гаусса-Зейделя:");

P.vectorPrint(P.vectorNev(matr, F, XGZ));

}

}

**Input.java**

**Класс для инициализации информации из файла**

public class Input{

String a;

int b;

FileReader fr;

BufferedReader br;

String nameFile;

**Конструктор**

Input(String name)

{

try

{

fr = new FileReader(name);

br = new BufferedReader(fr);

nameFile=name;

}

catch (IOException e)

{

System.out.println("Ошибка чтения");

}

}

**Чтение строки**

public String stringTake() throws IOException

{

a=new String();

try

{

a=br.readLine();

}

catch (IOException e)

{

System.out.println("Ошибка чтения с клавиатуры");

}

return a;

}

**Чтение размерности**

public int sizeTake() throws IOException

{

b=0;

try

{

String line = br.readLine();

b = Integer.parseInt(line);

}

catch (NumberFormatException e)

{

System.out.println("Не целое число");

}

catch (IOException e)

{

System.out.println("Ошибка чтения с клавиатуры");

}

return b;

}

**Создание вектора**

public double[] createVector()

{

double [] F;int n;

StringTokenizer st;

try{

n = Integer.parseInt(stringTake());

st=new StringTokenizer(stringTake()," ");

F = new double[n];

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

{

F[i] = Double.parseDouble(st.nextToken());

}

}

catch (IOException e)

{

System.out.println("Ошибка чтения");

st=new StringTokenizer(""," ,;");

F = new double[1];F[0]=0;

}

return F;

}

**Создание матрицы**

public double[][] create2XMatrix ( )

{

double [][]M;int n;int m;

StringTokenizer st;

try{

st = new StringTokenizer(stringTake()," ,;");

n = Integer.parseInt(st.nextToken());

m=n+1;

}

catch (IOException e)

{

System.out.println("Ошибка чтения");

st=new StringTokenizer(""," ,;");

n=0;m=0;

}

M=new double[n][m];

try{

st = new StringTokenizer(stringTake()," ,;");

}

catch (IOException e)

{

System.out.println("Ошибка чтения");

st=new StringTokenizer(""," ,;");

}

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

{

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

{

if(st.hasMoreTokens())

{

M[i][j]=Double.parseDouble(st.nextToken());

}

else

{

M[i][j]=0;

}

}

if(i<n-1)

{

try{

st = new StringTokenizer(stringTake()," ,;");

}

catch (IOException e)

{

System.out.println("Ошибка чтения");

st=new StringTokenizer(""," ,.;");

}

}

}

return M;

} }

**Podschet.java**

**Класс для вычислений**

public class Podschet {

int stop = 6; NumberFormat formatter = NumberFormat.getNumberInstance();

**Нахождение транспонированной матрицы**

public double[][] getTranspMatrix (double[][]A)

{

double[][]B = new double[A.length][A.length];

double [][]C = new double[A.length][A.length];

for(int i=0;i<C.length;i++)

{

for(int j=0;j<C.length;j++)

{

C[i][j]=0;

}

}

for(int i=0;i<A.length;i++)

{

for(int j=0;j<A.length;j++)

{

B[i][j]=A[j][i];

}

}

return B;

}

**Нахождение матрицы A(t)\*A**

public double[][] getATAMatrix(double[][]A,double[][]AT)

{

double[][]res = new double[A.length][A.length];

for(int i =0;i<A.length;i++)

{

for(int j=0;j<A.length;j++)

{

res[i][j]=0;

for(int k=0;k<A.length;k++)

{

res[i][j]+=AT[i][k]\*A[k][j];

}

}

}

return res;

}

**Нахождение нормы**

public double getMatrixNorma(double[][]A)

{

double res=0;double tempRes=0;

for(int i=0;i<A.length;i++)

{

res+=Math.abs(A[0][i]);

}

for(int i=1;i<A.length;i++)

{

tempRes=0;

for(int j=0;j<A.length;j++)

{

tempRes+=Math.abs(A[i][j]);

}

if(tempRes>res)

{

res = tempRes;

}

}

return res;

}

**Нахождение матрицы B**

public double[][]getMatrixB(double[][]ATA,double norma)

{

double[][]tempMatr = new double[ATA.length][ATA.length];

for(int i=0;i<ATA.length;i++)

{

for(int j=0;j<ATA.length;j++)

{

tempMatr[i][j]=ATA[i][j]/norma;

}

}

double[][]B = new double[ATA.length][ATA.length];

for(int i=0;i<ATA.length;i++)

{

for(int j=0;j<ATA.length;j++)

{

if(i==j)

{

B[i][j]=1-tempMatr[i][j];

}

else

{

B[i][j]=-tempMatr[i][j];

}

}

}

return B;

}

**Нахождения вектора G**

public double[] getVectorG (double[] F, double ATAnorma,double[][]matr)

{

double[] res = new double[F.length];

for(int i=0;i<F.length;i++)

{

res[i]=0;

for(int j=0;j<F.length;j++)

{

res[i]=res[i]+matr[j][i]\*F[j];

}

}

for(int i=0;i<F.length;i++)

{

res[i]=res[i]/ATAnorma;

}

return res;

}

**Число итераций**

public int getMaxIter(double[][]B,double[]g,double E)

{

double normaB;double normaG;

double tempNorma=0;int k;

for(int i=0;i<B.length;i++)

{

tempNorma = tempNorma+B[0][i];

}

normaB=tempNorma;

for(int i=1;i<B.length;i++)

{

tempNorma=0;

for(int j=0;j<B.length;j++)

{

tempNorma=tempNorma+B[i][j];

}

if(tempNorma>normaB)

{

normaB=tempNorma;

}

}

normaG=g[0];

for(int i=1;i<g.length;i++)

{

if(g[i]>normaG)

{

normaG=g[i];

}

}

double firstLogElem=E\*(1-normaB)/normaG;

k = (int)((Math.log(firstLogElem)/Math.log(normaB))+1);

return k;

}

**Вектор невязки**

public double[] vectorNev (double[][]matr,double[]F,double[]X)

{

double[] vector = new double[X.length];

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

{

vector[i]=0;

for(int j=0;j<X.length;j++)

{

vector[i]=vector[i]+matr[i][j]\*X[j];

}

vector[i]=vector[i]-F[i];

}

return vector;

}

**Печать вектора**

public void vectorPrint(double[] X)

{

formatter.setMaximumFractionDigits(stop);

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

{

System.out.print(formatter.format(X[i])+" ");

}

System.out.println();

}

}

**Methods.java**

**Класс с реализацией методов**

public class Methods {

int stop = 6;

NumberFormat formatter = NumberFormat.getNumberInstance();

**Метод простых итераций**

public double[] iterationMethod(double[][]B,double[]G,double E){

formatter.setMaximumFractionDigits(stop);

double[] X = new double[G.length];double[]Xtemp = new double[G.length];

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

{

X[i]=G[i];

}

int microAdd;int iterNum=0;

do{

microAdd=0;

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

{

Xtemp[i]=0;

for(int j=0;j<X.length;j++)

{

Xtemp[i]=Xtemp[i]+B[i][j]\*X[j];

}

Xtemp[i]=Xtemp[i]+G[i];

if(Math.abs(Xtemp[i]-X[i])<=E)

{

microAdd++;

}

}

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

{

X[i]=Xtemp[i];

}

iterNum++;

}while(microAdd<G.length);

System.out.println("Результат метода простой итерации");

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

{

System.out.print(formatter.format(X[i])+" ");

}

System.out.println();

System.out.println("Количество итераций = "+iterNum);

return X;

}

**Метод Зейделя**

public double[] methodZeidelia(double[][]B,double[]G,double E){

formatter.setMaximumFractionDigits(stop);

double[] X = new double[G.length];double[]Xtemp = new double[G.length];

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

{

X[i]=G[i];

}

int microAdd;int iterNum=0;

do{

microAdd=0;

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

{

Xtemp[i]=0;

for(int j=0;j<X.length;j++)

{

if(i<j)

{

Xtemp[i]=Xtemp[i]+B[i][j]\*Xtemp[j];

}

else

{

Xtemp[i]=Xtemp[i]+B[i][j]\*X[j];

}

}

Xtemp[i]=Xtemp[i]+G[i];

if(Math.abs(Xtemp[i]-X[i])<=E)

{

microAdd++;

}

}

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

{

X[i]=Xtemp[i];

}

iterNum++;

}while(microAdd<G.length);

System.out.println("Результат метода Зейделя");

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

{

System.out.print(formatter.format(X[i])+" ");

}

System.out.println();

System.out.println("Количество итераций = "+iterNum);

return X;

}

**Метод Якоби**

public double[] methodYakobi ( double[][]matr,double[]F,double E) {

formatter.setMaximumFractionDigits(stop);

double[]X = new double[F.length];

double[]Xtemp = new double[F.length];

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

{

X[i]=F[i];

}

int microAdd;int iterationNum=0;

do

{

microAdd=0;

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

{

Xtemp[i]=F[i]/matr[i][i];

for(int j=0;j<X.length;j++)

{

if(i!=j)

{

Xtemp[i]=Xtemp[i]+(-1)\*(matr[i][j]/matr[i][i])\*X[j];

}

}

if(Math.abs(Xtemp[i]-X[i])<E)

{

microAdd++;

}

}

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

{

X[i]=Xtemp[i];

}

iterationNum++;

}while(microAdd<X.length);

System.out.println();

System.out.println("Результат метода Якоби:");

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

{

System.out.print(formatter.format(X[i])+" ");

}

System.out.println();

System.out.println("Количество итераций = " + iterationNum);

return X;

}

**Метод Гаусса-Зейделя**

public double[] methodGZ ( double[][]matr,double[]F,double E) {

formatter.setMaximumFractionDigits(stop);

double[]X = new double[F.length];

double[]Xtemp = new double[F.length];

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

{

X[i]=F[i];

}

int microAdd;int iterationNum=0;

do

{

microAdd=0;

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

{

Xtemp[i]=F[i]/matr[i][i];

for(int j=0;j<X.length;j++)

{

if(i!=j)

{

if(i<j)

{

Xtemp[i]=Xtemp[i]+(-1)\*(matr[i][j]/matr[i][i])\*Xtemp[j];

}

else

{

Xtemp[i]=Xtemp[i]+(-1)\*(matr[i][j]/matr[i][i])\*X[j];

}

}

}

if(Math.abs(Xtemp[i]-X[i])<E)

{

microAdd++;

}

}

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

{

X[i]=Xtemp[i];

}

iterationNum++;

}while(microAdd<X.length);

System.out.println("Результат метода Гаусса-Зейделя");

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

{

System.out.print(formatter.format(X[i])+" ");

}

System.out.println();

System.out.println("Количество итераций = "+ iterationNum);

return X;

}

}

**7.Входные данные**

**А b**

0.4997 -0.0658 0.0132 0.0263 0.0921 -2.8141

0.0684 0.7824 0.0000 -0.0526 0.0526 2.4104

0.0395 0.0000 0.6286 -0.1841 0.1052 2.2828

-0.0789 0.1657 0.0000 0.6181 -0.0263 -1.6332

0.3288 0.0000 0.1184 0.0132 0.7364 1.8936

**8.Выходные данные**

Исходная матрица:

0.4997 -0.0658 0.0132 0.0263 0.0921

0.0684 0.7824 0.0 -0.0526 0.0526

0.0395 0.0 0.6286 -0.1841 0.1052

-0.0789 0.1657 0.0 0.6181 -0.0263

0.3288 0.0 0.1184 0.0132 0.7364

Исходный вектор f:

-2.8141 2.4104 2.2828 -1.6332 1.8936

Матрица B:

0,655492 -0,007036 -0,06546 0,039222 -0,277244

-0,007036 0,400873 0,000808 -0,055392 -0,028597

-0,06546 0,000808 0,619153 0,105895 -0,143781

0,039222 -0,055392 0,105895 0,609623 0,024421

-0,277244 -0,028597 -0,143781 0,024421 0,474043

Вектор G:

-0,371876 1,675159 1,509155 -1,493824 1,437639

Результат метода простой итерации:

-6,000459 3,000268 2,00031 -3,999772 5,000676

Количество итераций = 79

Результат метода Зейделя:

-6,000454 3,000268 2,000311 -3,999771 5,000672

Количество итераций = 79

Априорное количество итераций для методов Якоби и Гаусса-Зейделя = 42

Результат метода Якоби:

-6,000517 3,000267 2,000294 -3,999783 5,000721

Количество итераций = 13

Результат метода Гаусса-Зейделя:

-6,000517 3,000267 2,000294 -3,999783 5,000721

Количество итераций = 13

Вектор невязки для метода простой итерации:

0,000025 0,000001 0,000006 0,000004 -0,000013

Вектор невязки для метода Зейделя:

0,000028 0,000002 0,000006 0,000004 -0,000014

Вектор невязки для метода Якоби:

0 -0 0 0 -0,000001

Вектор невязки для метода Гаусса-Зейделя:

0 0 0 0 -0,000001