БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Лабораторная работа №2

**Метод Прогонки**

**Выполнил:**

Крючков Василий

2 курс 9 группа

**Преподаватель:**

Горбачева Ю.Н.

Минск, 2021

**Постановка задачи**

Написать и отладить программу, реализующую метод прогонки для численного решения систем линейных алгебраических уравнений Ay = f с трехдиагональной матрицей A порядка N +1.

**Краткие теоретические сведения**

Алгоритм метода правой прогонки решения систем линейных алгебраических уравнений с трехдиагональной матрицей:

1. Вычисляем прогоночные коэффициенты по формулам (прямая прогонка):

; ,

; ,

2. Вычисляем решение по формулам (обратная прогонка):

*; ,*

Условия, гарантирующие корректность и устойчивость метода:

Матрица A имеет диагональное преобладание по строкам, причем хотя бы в

одном случае строгое.

Относительная погрешность вычислялась по формуле:

, где,

**Листинг программы**

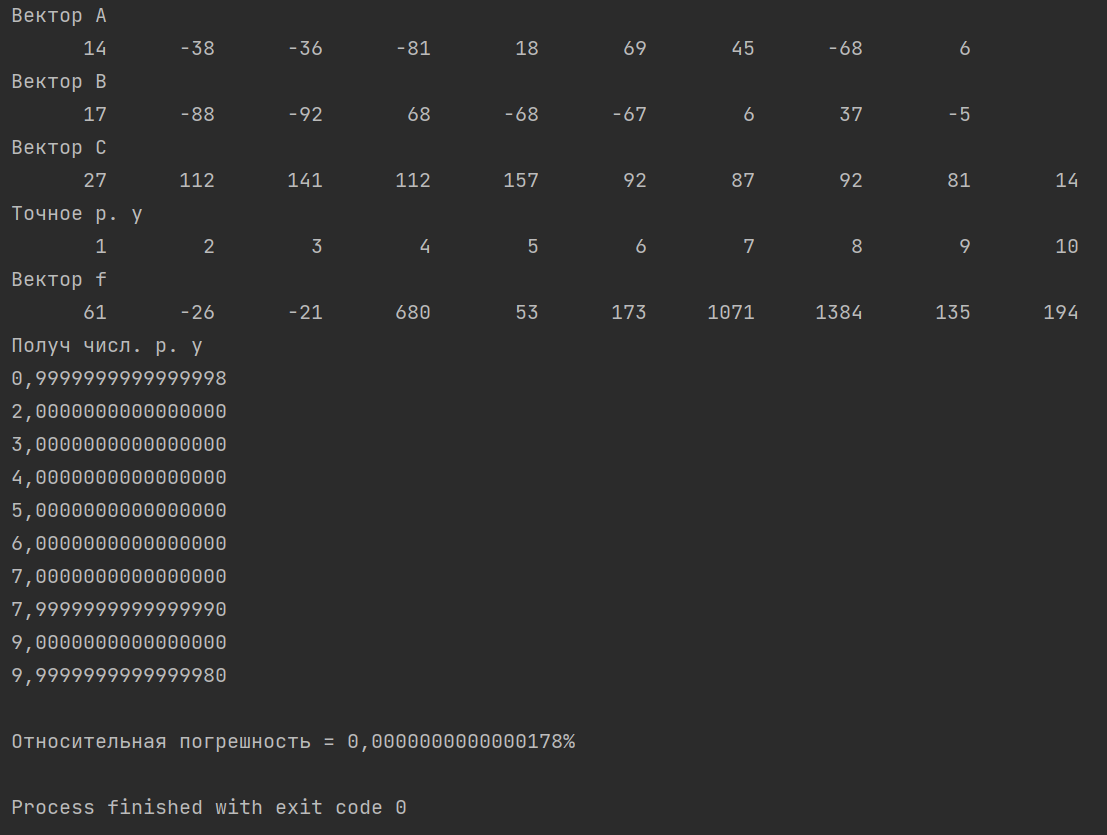
**Lab.java**

import java.util.Formatter;  
import java.util.Random;

public class Lab {  
  
 public static void runThroughMethod(int N,double [] vectorA,double [] vectorB,double [] vectorC,double [] columnF,double [] columnY){  
 double [] alpha = new double[N+2];  
 double [] betta = new double[N+2];  
 //Прямая прогонка  
 alpha[1]=vectorB[0]/vectorC[0];  
 betta[1]=columnF[0]/vectorC[0];  
 for(int i=1;i<=N-1;i++)  
 alpha[i+1]=vectorB[i]/(vectorC[i]-vectorA[i]\* alpha[i]);  
 for(int i=1;i<=N;i++)  
 betta[i+1]=(columnF[i]-vectorA[i]\*betta[i])/(vectorC[i]-vectorA[i]\* alpha[i]);  
 for(int i=0;i<=N;i++) {  
 if (alpha[i] == Double.*POSITIVE\_INFINITY* || alpha[i] == Double.*NEGATIVE\_INFINITY*)  
 throw new ArithmeticException("Деление на ноль");  
 if (betta[i] == Double.*POSITIVE\_INFINITY* || betta[i] == Double.*NEGATIVE\_INFINITY*)  
 throw new ArithmeticException("Деление на ноль");  
 }  
 //Обратная прогонка  
 columnY[N]=betta[N+1];  
 for(int i=N-1;i>=0;i--)  
 columnY[i] = (-1.0)\*alpha[i + 1] \* columnY[i + 1] + betta[i + 1];  
 }  
  
  
 public static void main(String[] args) {  
 int N = 9;  
 int k =6;  
 double [] vectorA = new double[N+1];  
 double [] vectorB = new double[N];  
 double [] vectorC = new double[N+1];  
 double [] columnY = new double[N+1];  
 double [] exactColumnY = new double[N+1];  
 double [] columnF = new double[N+1];  
  
  
 //Заполняем вектор а  
 Random rand = new Random();  
 for (int i=1; i<=N; i++)  
 vectorA[i] = -100 + rand.nextInt(200+1);  
  
 //Заполняем вектор b  
 for (int i=0; i<=N-1; i++)  
 vectorB[i] = -100 + rand.nextInt(200+1);  
  
 //Заполняем вектор c  
 for (int i=1; i<=N-1; i++)  
 vectorC[i] = Math.*abs*(vectorA[i]) + Math.*abs*(vectorB[i]) + k + rand.nextInt(k+1);  
 vectorC[0]=Math.*abs*(vectorB[0]) + k + rand.nextInt(k+1);  
 vectorC[N]=Math.*abs*(vectorA[N]) + k + rand.nextInt(k+1);

//Точное решение y  
 for (int i=0; i<=N; i++)  
 exactColumnY[i]=i+1;  
  
 //вектор правой части f  
 for(int i = 1; i <= N-1; i++)  
 columnF[i] = vectorA[i]\*exactColumnY[i-1]+vectorC[i]\*exactColumnY[i]+ vectorB[i]\*exactColumnY[i+1];  
 columnF[0] = vectorC[0]\*exactColumnY[0]+ vectorB[0]\*exactColumnY[1];  
 columnF[N] = vectorA[N]\*exactColumnY[N-1]+vectorC[N]\*exactColumnY[N];  
  
  
//Вызываем метод  
 try {  
 *runThroughMethod*(N,vectorA,vectorB,vectorC,columnF,columnY);  
 }catch(ArithmeticException e){  
 System.*out*.println(e.getMessage());  
 System.*exit*(0);  
 }  
   
  
  
 //Погрешность  
 double normXX=Math.*abs*(exactColumnY[0]-columnY[0]);  
 double normX = Math.*abs*(exactColumnY[0]);  
 for (int i=1; i<=N; i++) {  
 if(Math.*abs*(exactColumnY[i]-columnY[i])>normXX)  
 normXX=Math.*abs*(exactColumnY[i]-columnY[i]);  
 if(Math.*abs*(exactColumnY[i])>normX)  
 normX=Math.*abs*(exactColumnY[i]);  
 }  
 double accuracy=normXX/normX\*100;  
  
  
 //Ввывод результатов  
 Formatter f = new Formatter();  
 f.format("Вектор А %n");  
 for (int i=1; i<=N; i++) {  
 f.format("%8.0f ", vectorA[i]);  
 }  
 f.format("%n");  
 System.*out*.print(f);  
  
 f = new Formatter();  
 f.format("Вектор В %n");  
 for (int i=0; i<=N-1; i++) {  
 f.format("%8.0f ", vectorB[i]);  
 }  
 f.format("%n");  
 System.*out*.print(f);  
  
 f = new Formatter();  
 f.format("Вектор С %n");  
 for (int i=0; i<=N; i++) {  
 f.format("%8.0f ", vectorC[i]);  
 }  
 f.format("%n");  
 System.*out*.print(f);  
  
 f = new Formatter();  
 f.format("Точное р. y %n");  
 for (int i=0; i<=N; i++) {  
 f.format("%8.0f ", exactColumnY[i]);  
 }  
 f.format("%n");  
 System.*out*.print(f);  
  
 f = new Formatter();  
 f.format("Вектор f %n");  
 for (int i=0; i<=N; i++) {  
 f.format("%8.0f ", columnF[i]);  
 }  
 f.format("%n");  
 System.*out*.print(f);  
  
 f = new Formatter();  
 f.format("Получ числ. р. y%n");  
 for (int i=0; i<=N; i++) {  
 f.format("%16.16f%n", columnY[i]);  
 }  
 f.format("%n");  
 System.*out*.print(f);  
  
 System.*out*.print("Относительная погрешность =");  
 f = new Formatter();  
 f.format("%19.16f%%%n", accuracy);  
 System.*out*.print(f);  
 }  
}

**Результаты**

****

**Выводы**

Метод правой прогонки решения систем линейных алгебраических уравнений является эффективным алгоритмом для систем со трехдиагональными матрицами. Метод превосходно демонстрирует, как сильно учет специфики задачи позволяет построить более экономичные и в тоже время точные алгоритмы по сравнению с универсальными.