**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №2

**Метод прогонки**

**Выполнил:**

Кендысь Алексей Максимович

студент 2 курса, 9 группы,

специальность

“прикладная математика”

**Преподавательница:**

Ассистентка кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2021

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2

Краткие теоретические сведения ------------------------------------------------ 2-3

Листинг программы ---------------------------------------------------------------- 3-5

Результаты --------------------------------------------------------------------------- 6

Выводы ------------------------------------------------------------------------------- 6

**Постановка задачи**

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

**Краткие теоретические сведения**

Рассматриваем систему вида

Алгоритм правой прогонки:

Прямой ход метода прогонки (прямая прогонка):

Обратный ход метода прогонки (обратная прогонка):

=

= +

Теорема (достаточное условие применения и устойчивости метода прогонки). Пусть коэффициенты системы удовлетворяют следующим условиям:

И пусть одно из последних трёх неравенств выполняется строго. Тогда метод прогонки применим и устойчив.

**Листинг программы**

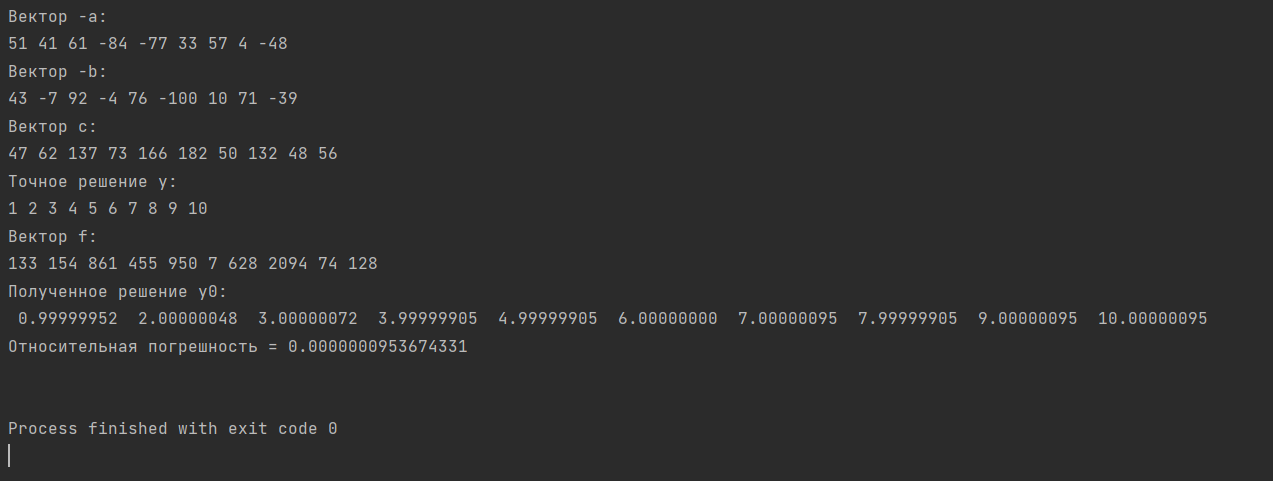
Файл Matrix.java:

import java.util.Random;  
import java.util.\*;  
public class Matrix {  
 private final static int *K* = 4;  
 private final int N;  
 private final Integer[] a;  
 private final Integer[] b;  
 private final Integer[] c;  
 private final Integer[] y;  
 private final Integer[] f;  
 private Float[] y0;  
 Matrix(int n) {  
 N = n - 1;  
  
 Random r = new Random();  
  
 //Заполнение вектора -a  
 a = new Integer[N];  
 for (int i = 0; i < N; i++) {  
 a[i] = r.nextInt(201) - 100;  
 }  
  
 //Заполнение вектора -b  
 b = new Integer[N];  
 for (int i = 0; i < N; i++) {  
 b[i] = r.nextInt(201) - 100;  
 }  
  
 //Заполнение вектора c  
 c = new Integer[N + 1];  
 c[0] = r.nextInt(*K* + 1) + (Math.*abs*(b[0]) + *K*);  
 for (int i = 1; i < N; i++) {  
 c[i] = r.nextInt(*K* + 1) + (Math.*abs*(b[i]) + Math.*abs*(a[i - 1]) + *K*);  
 }  
 c[N] = r.nextInt(*K* + 1) + (Math.*abs*(a[N - 1]) + *K*);  
  
 //Заполнение вектора y  
 y = new Integer[N + 1];  
 for (int i = 0; i < N + 1; i++) {  
 y[i] = i + 1;  
 }  
  
 //Заполнение вектора f  
 f = new Integer[N + 1];  
 f[0] = c[0] \* y[0] + b[0] \* y[1];  
 for (int i = 1; i < N; i++) {  
 f[i] = a[i - 1] \* y[i - 1] + c[i] \* y[i] + b[i] \* y[i + 1];  
 }  
 f[N] = a[N - 1] \* y[N - 1] + c[N] \* y[N];  
 }  
 public void algorithm() {  
 //Метод прогонки  
  
 //Образование массивов альфа и бета  
 float[] alpha = new float[N];  
 float[] beta = new float[N + 1];  
  
 //Прямой ход  
 alpha[0] = (float) -b[0] / c[0];  
 beta[0] = (float) f[0] / c[0];  
 float d;  
 for (int i = 1; i < N; i++) {  
 d = c[i] + a[i - 1] \* alpha[i - 1];  
 alpha[i] = -b[i] / d;  
 beta[i] = (f[i] - a[i - 1] \* beta[i - 1]) / d;  
 }  
 beta[N] = (f[N] - a[N - 1] \* beta[N - 1]) / (c[N] + a[N - 1] \* alpha [N - 1]);  
  
 //Обратный ход, заполнение вектора y0  
 y0 = new Float[N + 1];  
 y0[N] = beta[N];  
 for (int i = N - 1; i >= 0; i--) {  
 y0[i] = alpha[i] \* y0[i + 1] + beta[i];  
 }  
 }  
 public void outA() {  
 //Вывод вектора -a  
 for (int i : a) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println();  
 }  
 public void outB() {  
 //Вывод вектора -b  
 for (int i : b) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println();  
 }  
 public void outC() {  
 //Вывод вектора c  
 for (int i : c) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println();  
 }  
 public void outY() {  
 //Вывод вектора y  
 for (int i : y) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println();  
 }  
 public void outF() {  
 //Вывод вектора f  
 for (int i : f) {  
 System.*out*.print(i + " ");  
 }  
 System.*out*.println();  
 }  
 public void outY0() {  
 //Вывод вектора y0  
 Formatter fmt = new Formatter();  
 for (float i : y0) {  
 fmt.format("%- 12.8f", i);  
 }  
 System.*out*.println(fmt);  
 fmt.close();  
 }  
 public void outError() {  
 //Вывод погрешности  
 if (y0 == null) return;  
 Formatter fmt = new Formatter();  
 fmt.format("%.16f\n", mesError(y, y0));  
 System.*out*.println(fmt);  
 fmt.close();  
 }  
 private Float mesError(Integer[] x1, Float[] x2) {  
 //Подсчёт относительной погрешности  
 float max1 = 0;  
 float max2 = 0;  
 for (int i = 0; i < x1.length; i++) {  
 if (Math.*abs*(x1[i] - x2[i]) > max1) {  
 max1 = Math.*abs*(x1[i] - x2[i]);  
 }  
 if(Math.*abs*(x1[i]) > max2) {  
 max2 = Math.*abs*(x1[i]);  
 }  
 }  
 return (max1/max2);  
 }  
}

Файл Main.java:

import java.io.PrintWriter;  
public class Main {  
 public static void main(String[] args) {  
 PrintWriter out = new PrintWriter(System.*out*, true);  
 Matrix myMatrix = new Matrix(10);  
 out.println("Вектор -a:");  
 myMatrix.outA();  
 out.println("Вектор -b:");  
 myMatrix.outB();  
 out.println("Вектор c:");  
 myMatrix.outC();  
 out.println("Точное решение y:");  
 myMatrix.outY();  
 out.println("Вектор f:");  
 myMatrix.outF();  
 myMatrix.algorithm();  
 out.println("Полученное решение y0:");  
 myMatrix.outY0();  
 System.*out*.print("Относительная погрешность = ");  
 myMatrix.outError();  
 }  
}

**Результаты**

****

**Выводы**

1) Метод прогонки является частным случаем метода Гаусса, соответственно точность метода прогонки сопоставима с точностью метода Гаусса.

2) По методу прогонки требуется произвести операций умножения и деления, т.е. трудоёмкость метода линейна.

3) Также стоит отметить, что с помощью метода прогонки можно вычислить определитель матрицы:

det A =