**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

Крагель Алина Олеговна

**Метод прогонки**

Отчет по лабораторной работе №2

(«Вычислительные методы алгебры»)

Студента 2 курса 10 группы

Работа сдана \_\_\_октября 2020 г. Преподаватель:

Зачтена \_\_\_\_\_\_\_\_\_ 2020 г. Горбачева Юлия Николаевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ассистент кафедры

(Подпись преподавателя) вычислительной математики

Минск 2020

**Постановка задачи**

Написать программу, которая решает систему линейных алгебраических уравнений *Ay=f* для трехдиагональной матрицы *A* порядка *N + 1*.

Для проведения вычислительного эксперимента необходимо сгенерировать случайную трехдиагональную матрицу

*A* с диагональным преобладанием размерности *N +1 = 10*, т.е. генерируетевектор – коэффициенты матрицы *A* ниже главной диагонали; вектор– коэффициенты матрицы *A* на главной диагонали; вектор – коэффициенты матрицы *A* выше главной диагонали. Генерируете вектор точного решения *.* Вектор правой части *f* задать умножением матрицы *A* на вектор *y*: *f = Ay*.

Затем необходимо решить полученную систему с помощью вашей программы и занести в отчет результаты. В результатах выполнения тестовой задачи необходимо привести следующую информацию:

* Условие: векторы , точное решение *y.*
* Полученное решение.
* Максимум-норма невязки .
* Максимум-норма погрешности .

**Теория**

Метод прогонки — частный случай метода Гаусса решения системы линейных алгебраических уравнений для трехдиагональной матрицы. Имеем:

Рассмотрим систему: *A\*y = f*, где *А* – матрица выше, – неизвестные переменные, - столбец свободных членов.

Считаем . Воспользуемся формулами

- прямой ход.

Решение системы:

– обратный ход.

Теорема.

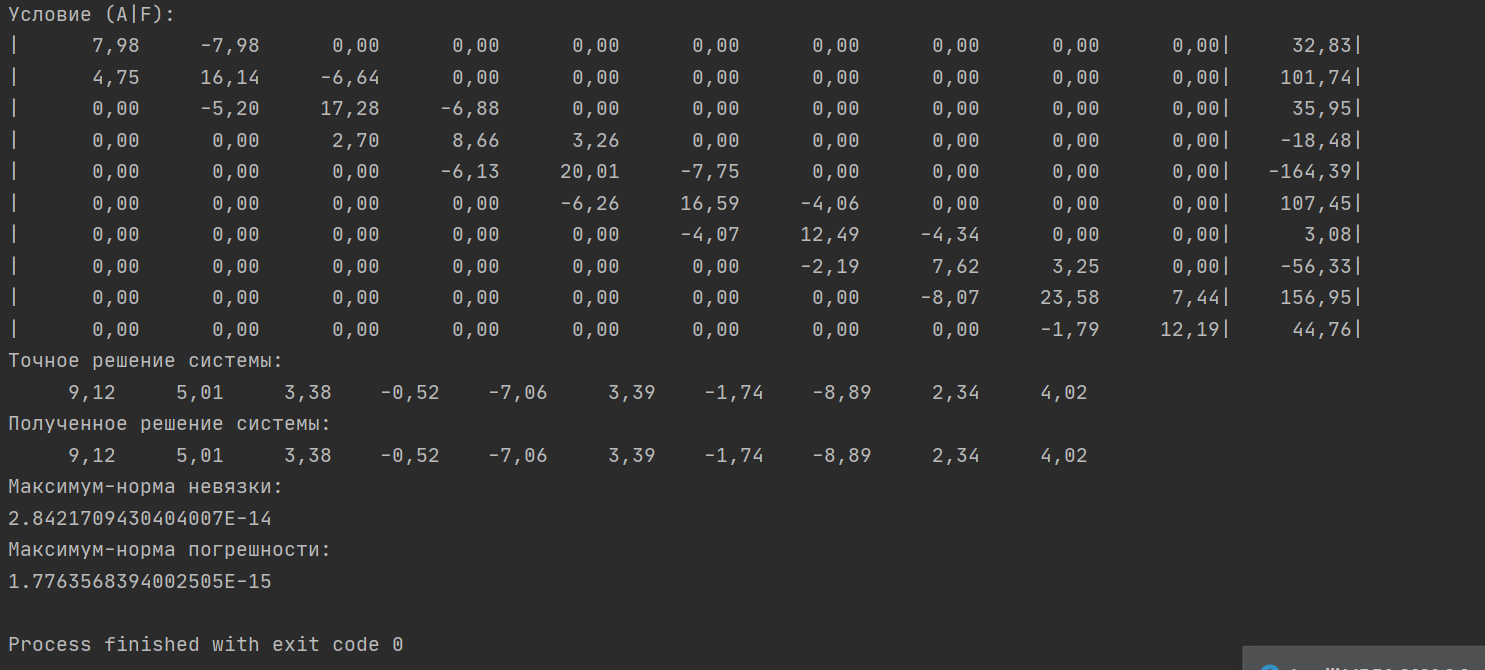
Пусть коэффициенты системы удовлетворяют условиям:

и хотя бы одно из неравенств (2)-(3) выполняется строго. Тогда метод прогонки применим и устойчив.

**Листинг**

import java.util.Random;  
import java.util.Date;  
  
public class SweepMethod {  
 //Определим основные операнды соотвественно: число N, векторы коэффицентов, точное решение, вектор свободных членок и вычисленное решение  
 public static int *N* = 9;  
 public static double[] *a* = new double[*N* + 1];  
 public static double[] *c* = new double[*N* + 1];  
 public static double[] *b* = new double[*N*];  
 public static double[] *xPrecise* = new double[*N* + 1];  
 public static double[] *f* = new double[*N* + 1];  
 public static double[] *xCalculated* = new double[*N* + 1];  
 //Генератор уравнения и его основных элементов  
 public static void EquationGeneration(){  
 Random myRand = new Random((new Date()).getTime());  
 *a*[0] = 0;  
 //Вектор a  
 for (int i = 1; i < *N* + 1; i++) {  
 *a*[i] = (myRand.nextDouble() \* (20) - 10);  
 }  
 //Вектор b  
 for (int i = 0; i < *N*; i++) {  
 *b*[i] = (myRand.nextDouble() \* (20) - 10);  
 }  
 //Вектор c, равный сумме модулей двух предыдущих векторов  
 for (int i = 0; i < *N*; i++) {  
 *c*[i] = Math.*abs*(*a*[i + 1]) + Math.*abs*(*b*[i]);  
 }  
 *c*[*N*] = Math.*abs*(*a*[1]) + Math.*abs*(*b*[*N* - 1]);  
 //Точное решение  
 for (int i = 0; i < *N* + 1; i++) {  
 *xPrecise*[i] = (myRand.nextDouble() \* (20) - 10);  
 }  
 //Вектор свободных членов  
 *f*[0] = *c*[0] \* *xPrecise*[0] + *b*[0] \* *xPrecise*[1];  
 for (int i = 1; i < *N*; i++) {  
 *f*[i] = *a*[i] \* *xPrecise*[i - 1] + *c*[i] \* *xPrecise*[i] + *b*[i] \* *xPrecise*[i + 1];  
 }  
 *f*[*N*] = *a*[*N*] \* *xPrecise*[*N* - 1] + *c*[*N*] \* *xPrecise*[*N*];  
 }  
 //Непосредственно метод прогонки  
 public static void SweepMethod(){  
 double [] alpha = new double[*N* + 1];  
 double [] beta = new double[*N* + 2];  
 alpha[0] = 0;  
 beta[0] = 0;  
 alpha[1] = (-*b*[0]) / *c*[0];  
 beta[1] = *f*[0] / *c*[0];  
 for (int i = 1; i < *N*; i++) {  
 alpha[i + 1] = (-*b*[i]) / (*c*[i] - (-*a*[i]) \* alpha[i]);  
 beta[i + 1] = (*f*[i] + (-*a*[i]) \* beta[i]) / (*c*[i] - (-*a*[i]) \* alpha[i]);  
 }  
 beta[*N* + 1] = (*f*[*N*] + (-*a*[*N*]) \* beta[*N*]) / (*c*[*N*] - (-*a*[*N*]) \* alpha[*N*]);  
  
 *xCalculated*[*N*] = beta[*N* + 1];  
 for (int i = *N* - 1; i >= 0; i--) {  
 *xCalculated*[i] = (alpha[i + 1] \* *xCalculated*[i + 1] + beta[i + 1]);  
 }  
 }  
 //Печать уравнения  
 public static void PrintingEquation() {  
 double [][] equation = new double[*N* + 1][*N* + 2];  
 equation[0][0] = *c*[0]; equation[0][1] = *b*[0];  
 equation [0][*N* + 1] = *f*[0];  
 for (int i = 1; i < *N*; i++) {  
 equation[i][i - 1] = *a*[i]; equation[i][i] = *c*[i]; equation[i][i + 1] = *b*[i];  
 equation [i][*N* + 1] = *f*[i];  
 }  
 equation[*N*][*N* - 1] = *a*[*N*]; equation[*N*][*N*] = *c*[*N*];  
 equation [*N*][*N* + 1] = *f*[*N*];  
  
 System.*out*.print("Условие (A|F): \n");  
 for (int i = 0; i < *N* + 1; i++) {  
 System.*out*.print("|");  
 for (int j = 0; j < *N* + 2; j++) {  
 System.*out*.printf("%10.2f", equation[i][j]);  
 if (j == *N* || j == *N* + 1)  
 System.*out*.print("|");  
 if (j == *N* + 1)  
 System.*out*.println();  
 }  
 }  
 }  
 //Печать вектора  
 public static void ArrayPrinting(double[] M, int m) {  
 for (int i = 0; i < m; i++) {  
 System.*out*.printf("%9.2f", M[i]);  
 }  
 System.*out*.print("\n");  
 }  
 //Вычисление максимум-нормы невязки  
 public static double MaxNormDiscrepancy() {  
 double [] fCalc = new double[*N* + 1];  
 fCalc[0] = *c*[0] \* *xCalculated*[0] + *b*[0] \* *xCalculated*[1];  
 for (int i = 1; i < *N*; i++) {  
 fCalc[i] = *a*[i] \* *xCalculated*[i - 1] + *c*[i] \* *xCalculated*[i] + *b*[i] \* *xCalculated*[i + 1];  
 }  
 fCalc[*N*] = *a*[*N*] \* *xCalculated*[*N* - 1] + *c*[*N*] \* *xCalculated*[*N*];  
 double [] norms = new double[*N* + 1];  
 for (int i = 0; i < *N* + 1; i++) {  
 norms[i] = Math.*abs*(fCalc[i] - *f*[i]);  
 }  
 double maxND = norms[0];  
 for (int i = 0; i < *N* + 1; i++)  
 {  
 if (norms[i] > maxND)  
 maxND = norms[i];  
 }  
 return maxND;  
 }  
 //Вычисление максимум-нормы погрешности  
 public static double MaxNormError() {  
 double[]errorX = new double [*N* + 1];  
 for (int i = 0; i < *N* + 1; i++) {  
 errorX[i] = 0;  
 }  
 for (int i = 0; i < *N* + 1; i++) {  
 errorX[i] = Math.*abs*(*xCalculated*[i] - *xPrecise*[i]);  
 }  
 double maxNE = errorX[0];  
 for (int i = 0; i < *N* + 1; i++)  
 {  
 if (errorX[i] > maxNE)  
 maxNE = errorX[i];  
 }  
 return maxNE;  
 }  
 //Главная функция  
 public static void main (String[]args){  
 *EquationGeneration*();  
 *PrintingEquation*();  
 *SweepMethod*();  
 System.*out*.print("Точное решение системы: \n");  
 *ArrayPrinting*(*xPrecise*, *N* + 1);  
 System.*out*.print("Полученное решение системы: \n");  
 *ArrayPrinting*(*xCalculated*, *N* + 1);  
 System.*out*.println("Максимум-норма невязки: \n" + *MaxNormDiscrepancy*());  
 System.*out*.println("Максимум-норма погрешности: \n" + *MaxNormError*());  
  
 }  
}

**Результаты**

****

**Вывод**

Таким образом, в результате поставленного эксперимента удалось выяснить, что использованный метод прогонки позволяет минимизировать погрешность при решении систем линейных алгебраических уравнений, что доказывают полученные значения максимум-норм невязки и погрешности.