БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №2
Метод прогонки для решения систем линейных уравнений

Преподаватель: Полевиков Виктор Кузьмич
Студент: Шиляев Иван
2 курс 9 группа

Постановка задачи

Сгенерировать случайным образом вещественные элементы трёхдиагональной квадратной матрицы A размера (10x10) с точностью до двух знаков после запятой. Задать вектор X и вычислить f = AX. Далее:

1. Найти решение, используя метод прогонки;
2. Проверить условия применимости и устойчивости метода.

Сгенерированная матрица:

27,31	25,81	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
6,66	83,56	−54,88	0,00	0,00	0,00	0,00	0,00	0,00	0,00
0,00	9,94	85,11	61,71	0,00	0,00	0,00	0,00	0,00	0,00
0,00	0,00	−2,14	76,60	−15,02	0,00	0,00	0,00	0,00	0,00
0,00	0,00	0,00	42,19	100,18	18,79	0,00	0,00	0,00	0,00
0,00	0,00	0,00	0,00	21,38	−32,00	9,24	0,00	0,00	0,00
0,00	0,00	0,00	0,00	0,00	−5,36	72,77	−9,06	0,00	0,00
0,00	0,00	0,00	0,00	0,00	0,00	28,83	85,26	−11,31	0,00
0,00	0,00	0,00	0,00	0,00	0,00	0,00	−39,51	−66,83	6,20
0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	34,77	−89,25

Краткая теория

Метод прогонки или алгоритм Томаса используется для решения систем линейных уравнений вида Ax = f, где A — трёхдиагональная матрица. Метод представляет собой вариант метода последовательного исключения неизвестных.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & 0 & 0 & \dots & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & \dots & 0 \\ 0 & a_{3,2} & a_{3,3} & a_{3,4} & \dots & 0 \\ & & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & a_{n,n-1} & a_{n,n} \end{pmatrix}$$

Решение системы методом прогонки

Введём следующие обозначения: $a_{ii} = c_i$, $a_{i-1,i} = -a_i$, $a_{i+1,i} = -b_i$. Для вычисления коэффициентов системы выполним прямой ход метода прогонки:

$$\begin{cases} \alpha_{i+1} = \frac{b_i}{c_i - \alpha_i a_i}, \quad i = \overline{1,N-1}, \quad \alpha_1 = \frac{b_0}{c_0}, \\ \beta_{i+1} = \frac{f_i + \beta_i a_i}{c_i - \alpha_i a_i}, \quad i = \overline{1,N}, \quad \beta_1 = \frac{f_0}{c_0}. \end{cases}$$

В результате преобразований получим следующую систему:

$$\begin{cases} y_i - \alpha_{i+1} y_{i+1} = \beta_{i+1}, \quad i = \overline{0,N-1}, \\ -\alpha_N y_{N-1} + c_N y_N = f_N, \quad i = N. \end{cases}$$

Для нахождения решения системы выполним обратный ход метода прогонки:

$$y_N = \beta_{N+1}, \quad y_i = \alpha_{i+1} y_{i+1} + \beta_{i+1}, \quad i = N-1, N-2, \dots, 0.$$

В ходе решения воспользуемся теоремой о корректности и устойчивости метода прогонки:

$$c_0 \neq 0, \quad c_N \neq 0, \quad a_i \neq 0, \quad b_i \neq 0, \quad i = \overline{1,N-1}, \ (1)$$

$$|c_i| \geq |a_i| + |b_i|, \quad i = \overline{1,N-1}, \ (2)$$

$$|c_0| \geq |b_0|, \quad |c_N| \geq |a_N|. \ (3)$$

Согласно этой теореме метод прогонки применим и устойчив, когда хотя бы одно из последних двух неравенств (2), (3) выполняется строго.

По исходным данным видно, что строго выполняется третье условие теоремы: $|c_N| = |-89{,}25| \geq |a_N| = |34{,}77|$.

Листинг программы
(на языке программирования Java)

```
public class Column {
    private double[] column;
    protected final int n;

    public Column(Column C) {
        n = C.n;
        column = new double[n];
        for (int i = 0; i < n; ++i)
            column[i] = C.column[i];
    }

    public Column(double[] mas) {
        n = mas.length;
        column = new double[n];
        for (int i = 0; i < n; ++i)
            column[i] = mas[i];
    }

    public Column(int count, double min, double max) {
        n = count;
        column = new double[n];
        for (int i = 0; i < this.n; ++i) {
            column[i] = Solution.round(min + Math.random() * (max - min + 1),
2);
        }
    }

    public double[] getColumn () {
        return column;
    }
}


public class TridiagonalMatrix {
    private double[][] matrix;
    protected final int n;

    public TridiagonalMatrix(int count, double min, double max) {
        n = count;
        matrix = new double[n][n];
        do {
            matrix[0][0] = Solution.round(min + Math.random() * (max - min + 1),
2);
            matrix[n-1][n-1] = Solution.round(min + Math.random() * (max - min +
1), 2);
            matrix[0][1] = Solution.round(min + Math.random() * (max - min + 1),
2);
            matrix[n-1][n-2] = Solution.round(min + Math.random() * (max - min +
1), 2);
        }
        while (getC(0)==0 || getC(n-1)==0 || Math.abs(getC(0)) <
Math.abs(getB(0)) || Math.abs(getC(n-1)) <= Math.abs(getA(n-1)));
        // 1 and 3 conditions
        for (int i = 1; i < this.n - 1; ++i) {
            do {
                matrix[i][i] = Solution.round(min + Math.random() * (max - min +
1), 2);
                matrix[i][i - 1] = Solution.round(min + Math.random() * (max -
min + 1), 2);
```

```
                matrix[i][i + 1] = Solution.round(min + Math.random() * (max -
min + 1), 2);
            }
            while (getA(i) == 0 || getB(i) == 0 || Math.abs(getC(i)) <
(Math.abs(getA(i)) + Math.abs(getB(i))));
            // 1 and 2 conditions
        }
    }

    public TridiagonalMatrix(TridiagonalMatrix M) {
        n = M.n;
        matrix = new double[n][n];
        for (int i = 0; i < n; ++i)
            System.arraycopy(M.matrix[i], 0, matrix[i], 0, n);
    }

    public double getA(int idx) {
        if (idx < 1 || idx >= this.n)
            throw new IndexOutOfBoundsException("Bad value of index: " + idx +
". Index must be >=1 and <" + this.n + ".");
        return (-1)*matrix[idx][idx-1];
    }

    public double getB(int idx) {
        if (idx < 0 || idx >= this.n - 1)
            throw new IndexOutOfBoundsException("Bad value of index: " + idx +
". Index must be >=0 and <" + (this.n-1) + ".");
        return (-1)*matrix[idx][idx+1];
    }

    public double getC(int idx) {
        if (idx < 0 || idx >= this.n)
            throw new IndexOutOfBoundsException("Bad value of index: " + idx +
". Index must be >=0 and <" + this.n + ".");
        return matrix[idx][idx];
    }

    public double[][] getMatrix() {
        return matrix;
    }

    public Column multiplyMatrixWithColumn(Column C) throws Exception {
        if (C.n != this.n)
            throw new Exception("Wrong size of matrix");
        double[] ans = new double[C.n];
        for (int i = 0; i < this.n; ++i)
            for (int j = 0; j < this.n; ++j)
                ans[i] += matrix[i][j] * C.getColumn()[j];
        return new Column(ans);
    }
}


import java.util.Scanner;

public class Solution {
    public static double round(double value, int places) {
        if (places < 0) throw new IllegalArgumentException();
        long factor = (long) Math.pow(10, places);
        value = value * factor;
        long tmp = Math.round(value);
        return (double) tmp / factor;
    }

    private TridiagonalMatrix A;
```

```
    private Column exactY;
    private double[] y;
    private Column f;
    private final int N;
    private double[] alpha;
    private double[] beta;

    public Solution(int n, double min, double max) throws Exception {
        N = n;
        A = new TridiagonalMatrix(N+1, min, max);
        exactY = new Column(N+1, min, max); // random solution
        y = new double[N+1];
        alpha = new double [N+1];
        beta = new double [N+2];
        f = new Column(A.multiplyMatrixWithColumn(exactY)); // counting our f
    }

    public void directPassage() {
        alpha[1] = A.getB(0) / A.getC(0);
        beta[1] = f.getColumn()[0] / A.getC(0);
        for (int i = 1; i < N; ++i) {
            alpha[i+1] = A.getB(i) / (A.getC(i) - alpha[i] * A.getA(i));
            beta[i+1] = (f.getColumn()[i] + beta[i] * A.getA(i)) / (A.getC(i) -
alpha[i] * A.getA(i));
        }
        beta[N+1] = (f.getColumn()[N] + beta[N] * A.getA(N)) / (A.getC(N) -
alpha[N] * A.getA(N));
    }

    public void reversePassage() {
        y[N] = beta[N + 1];
        for (int i = N - 1; i >= 0; --i)
            y[i] = alpha[i + 1] * y[i + 1] + beta[i + 1];
    }

    public void printMatrix() {
        for (int i = 0; i < N + 1; ++i) {
            for (int j = 0; j < this.N + 1; ++j) {
                if (Math.abs(A.getMatrix()[i][j]) < 0.01)
                    System.out.printf("%7.2f ", 0.0);
                else System.out.printf("%7.2f ", A.getMatrix()[i][j]);
            }
            System.out.println();
        }
    }

    public void printMySolution() {
        for (int i = 0; i < y.length; ++i) {
            if (Math.abs(y[i]) < Math.pow(10, -20))
                System.out.println(0.0);
            else System.out.println(y[i]);
        }
    }

    public void printExactSolution() {
        for (int i = 0; i < N+1; ++i)
            System.out.println(exactY.getColumn()[i]);
    }

    public static void main (String[] args) {
        int size;
        double min, max;
        try {
            //Scanner in = new Scanner(System.in);
            //System.out.print("Введите размер матрицы: ");
```

```
            size = 9; //in.nextInt();
            /* if (size <= 0) {
                in.close();
                throw new Exception("Input error! [matrixCount <= 0]");
            }
            in.close(); */
            min = -100; max = 100;

            Solution mySystem = new Solution(size, min, max);
            System.out.println("Original matrix:");
            mySystem.printMatrix();
            // counting solution
            mySystem.directPassage();
            mySystem.reversePassage();
            // results
            System.out.println();
            System.out.println("Original solution:");
            mySystem.printExactSolution();
            System.out.println();
            System.out.println("My solution:");
            mySystem.printMySolution();
        }
        catch (Exception b) {
            System.out.println(b.getMessage());
        }
    }
}
```

Результат

Сгенерированная матрица:

```
 27,31   25,81    0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00
  6,66   83,56  -54,88    0,00    0,00    0,00    0,00    0,00    0,00    0,00
  0,00    9,94   85,11   61,71    0,00    0,00    0,00    0,00    0,00    0,00
  0,00    0,00   -2,14   76,60  -15,02    0,00    0,00    0,00    0,00    0,00
  0,00    0,00    0,00   42,19  100,18   18,79    0,00    0,00    0,00    0,00
  0,00    0,00    0,00    0,00   21,38  -32,00    9,24    0,00    0,00    0,00
  0,00    0,00    0,00    0,00    0,00   -5,36   72,77   -9,06    0,00    0,00
  0,00    0,00    0,00    0,00    0,00    0,00   28,83   85,26  -11,31    0,00
  0,00    0,00    0,00    0,00    0,00    0,00    0,00  -39,51  -66,83    6,20
  0,00    0,00    0,00    0,00    0,00    0,00    0,00    0,00   34,77  -89,25
```

Исходное точное решение:

```
71.2
72.99
-36.0
13.6
74.66
-22.06
-39.4
78.42
-24.96
88.34
```

Полученное решение:

```
71.20000000000002
72.99
-36.0
13.599999999999994
74.65999999999998
-22.060000000000016
-39.400000000000006
78.41999999999999
-24.959999999999994
88.34
```