Университет ИТМО, факультет ПИиКТ

Лабораторная работа №1 по

“Вычислительная математика”

Вариант: метод простых итераций

Выполнил: Яремко Р. О.

Группа: P3210

Преподаватель: Перл О.В.

Санкт-Петербург

2020г.

Задание (метод простых итераций)

Размерность n<=20 (задается из файла или с клавиатуры - по выбору конечного пользователя)

Должно быть предусмотрено чтение исходных данных как из файла, так и ввод с клавиатуры.

Должна быть реализована возможность ввода коэффициентов матрицы как с клавиатуры, так и из файла. Также предусмотреть случайные коэффициенты.

Обязательно: Тестовые данные на матрице большого размера (5*5 / 6*6...) + в отчёт с решением.

Для итерационных методов:

- Точность задается с клавиатуры/файла
- Проверка диагонального преобладания
  //В случае, если диагональное преобладание в изначальной матрице отсутствует - предлагается сделать перестановку строк/столбцов до тех пор, пока преобладание не будет достигнуто. В случае невозможности достижения диагонального преобладания - выводить сообщение.
- Столбец неизвестных
- Количество итераций, за которое было найдено решение
- Столбец погрешностей

Код программы

```
import java.io.PrintWriter;
import java.util.Locale;
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Collections;



public class Solution {
    private int size;
    private double[][] matrix;
    //user's error
    private double eps;
    PrintWriter printWriter = new PrintWriter(System.out);
    ArrayList<Double> errorList = new ArrayList<Double>();

    public static void main(String[] args) {
        Solution fixedPointIteration = new Solution();
```

```
        Scanner question;
        while (true){
                System.out.print("console/file: ");
                question = new Scanner(System.in);
                String choice = question.nextLine();
                if (choice.equals("console")){
                        fixedPointIteration.consoleWrite();
                        break;
                } else if (choice.equals("file")) {
                        try{
                                fixedPointIteration.readFile();
                                break;
                        } catch(FileNotFoundException exc){
                                System.out.println("Can not read file.");
                                System.exit(0);
                        }
                } else {
                        System.out.println("please write console or file");
                }
        }
        question.close();
        if (!fixedPointIteration.check()) {
                fixedPointIteration.shuffle();
                if (!fixedPointIteration.check()){
                        System.exit(0);
                }
        }
        fixedPointIteration.solve();
}

//make matrix satisfy the diagonal condition
private void shuffle(){
        System.out.println("shuffle the matrix");
        double[][] good = new double[size][size+1];
        for(int i = 0; i<size; i++){
                double maxi = 0;
                int num = 0;
                for (int j = 0;j<size;j++){
                        if (matrix[i][j] > maxi){
                                maxi = matrix[i][j];
                                num = j;
                        }
                }
                good[num] = matrix[i];
                maxi = 0;
                num = 0;
        }
        matrix = good;
}

//make sure that the matrix satisfy the diagonal condition
public boolean check(){
        int diag = 0;
        int other = 0;
```

```
        boolean ok = true;
        System.out.print("Checking the condition: ");
        for (int i = 0; i<size; i++){
                diag+=matrix[i][i];
                for(int j = 0; j<size; j++){
                        other+=matrix[i][j];
                }
                other = other - diag;
                if(diag < other){
                        ok = false;
                }
                diag = 0;
                other = 0;
        }
        System.out.println(ok);
        return ok;
}

//read data from file
private void readFile() throws FileNotFoundException{
        System.out.print("File name: ");
        Scanner in = new Scanner(System.in);
        in = new Scanner(new File(in.nextLine()));
        size = in.nextInt();
        matrix = new double[size][size + 1];
        eps = in.nextDouble();
        String shit = in.nextLine();

        int counter = 0;
        while(in.hasNextLine()){
                String[] line = in.nextLine().split(" ");
                for (int i = 0;i<size+1;i++){
                        matrix[counter][i] = Double.parseDouble(line[i]);
                }
                counter ++;
        }
        in.close();
}

//read data from console
private void consoleWrite(){
        Scanner scanner = new Scanner(System.in);
        scanner.useLocale(new Locale("Russian"));

        System.out.println("Input main matrix size");
        size = scanner.nextInt();

        matrix = new double[size][size + 1];

        System.out.println("Input matrix " + (size*size+size) +" numbers");
        String output = "Matrix:\n";
        for (int i = 0; i < size; i++) {
                for (int j = 0; j < size + 1; j++) {
                        matrix[i][j] = scanner.nextDouble();
```

```
                output += matrix[i][j] + " ";
            }
            output += "\n";
        }
        System.out.print(output);
        System.out.println("Input accuracy");
        eps = scanner.nextDouble();
        scanner.close();
    }

    //solve the System of linear equations
    private void solve(){
        double[] previousVariableValues = new double[size];
        for (int i = 0; i < size; i++) {
            previousVariableValues[i] = 0.0;
        }
        double error;
        while (true) {
            double[] currentVariableValues = new double[size];
            for (int i = 0; i < size; i++) {
                currentVariableValues[i] = matrix[i][size];
                for (int j = 0; j < size; j++) {
                    if (i != j) {
previousVariableValues[j];          currentVariableValues[i] -= matrix[i][j] *
                    }
                }
                currentVariableValues[i] /= matrix[i][i];
            }
            error = 0.0;
            ArrayList<Double> errors = new ArrayList<Double>();
            for (int i = 0; i < size; i++) {
                errors.add(Math.abs(currentVariableValues[i] -
                 previousVariableValues[i]));
            }
            error = Collections.max(errors);
            errorList.add(error);
            if (error < eps) { break; }
            previousVariableValues = currentVariableValues;
        }
        printWriter.print("Result:"+"\n");
        for (int i = 0; i < size; i++) {
            printWriter.print("Variable "+(i+1)+" = "+
                previousVariableValues[i]+"±"
                +error+"\n");
        }
        printWriter.print(errorList.size()+" iterations"+"\n");
        printWriter.print("Error list"+"\n");
        for (int i = 0; i < errorList.size(); i++) {
            printWriter.print(errorList.get(i)+"\n");
        }
        printWriter.close();
    }
}
```

Тестовые данные

- Тест 1 (файл, размер матрицы 6)

  Файл (sole):

  6
  0.0027
  10.0 1.0 1.0 1.0 1.0 1.0 15.0
  2.0 10.0 1.0 1.0 1.0 1.0 16.0
  2.0 2.0 10.0 1.0 1.0 1.0 17.0
  2.0 2.0 2.0 10.0 1.0 1.0 18.0
  2.0 2.0 2.0 2.0 10.0 1.0 19.0
  2.0 2.0 2.0 2.0 1.0 10.0 19.0
  Вывод программы:
  [salvoroni@salvoroni-pc ComputationalMath]$ java Solution
  console/file: file
  File name: sole
  Checking the condition: true
  Result:
  Variable 1 = 1.0009200834889882±0.0025055702028266014
  Variable 2 = 1.001034892550026±0.0025055702028266014
  Variable 3 = 1.0011640276158817±0.0025055702028266014
  Variable 4 = 1.0013092763016809±0.0025055702028266014
  Variable 5 = 1.0014726492831918±0.0025055702028266014
  Variable 6 = 1.0014726492831918±0.0025055702028266014
  20 iterations
  Error list
  1.9
  1.5099999999999998
  1.039
  0.7300999999999997
  0.5122099999999998
  0.3592010000000001
  0.25195970000000023
  0.17672285000000032
  0.12395427300000017
  0.08694190690000003
  0.060981327929999884
  0.04277250018900014
  0.030000768114500365
  0.02104263482645008
  0.014759371392641052
  0.010352270325063917
  0.007261115532722351
  0.005092969669834191
  0.003572225223676395

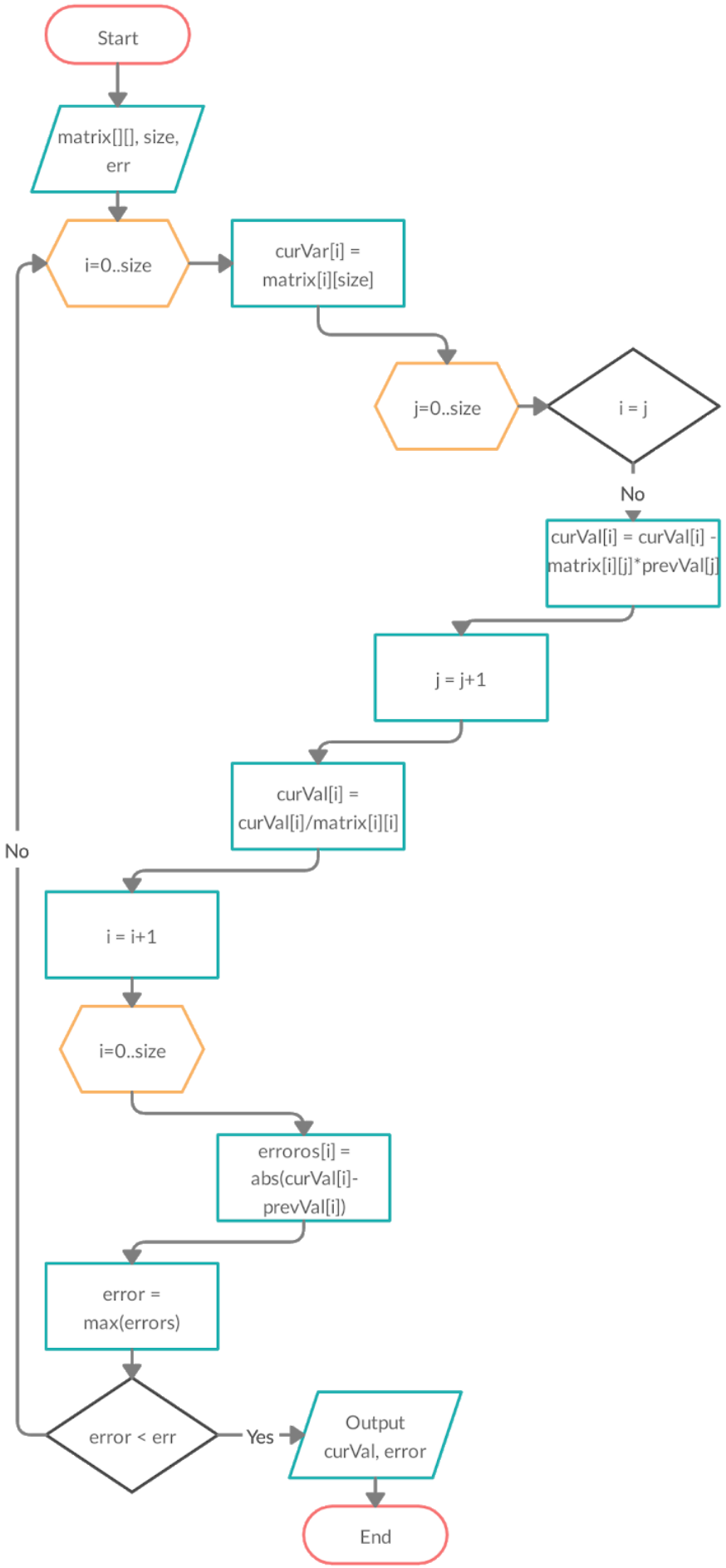
0.0025055702028266014

- Тест 2 (консоль, размер матрицы 3)

  Вывод программы:

  [salvoroni@salvoroni-pc ComputationalMath]$ java Solution
  console/file: console
  Input main matrix size
  3
  Input matrix 12 numbers
  10 1 1 12
  2 10 1 13
  2 2 10 14
  Matrix:
  10.0 1.0 1.0 12.0
  2.0 10.0 1.0 13.0
  2.0 2.0 10.0 14.0
  Input accuracy
  0.0027
  Checking the condition: true
  Result:
  Variable 1 = 0.999568±8.784000000000569E-4
  Variable 2 = 0.99946±8.784000000000569E-4
  Variable 3 = 0.999316±8.784000000000569E-4
  7 iterations
  Error list
  1.4
  0.4999999999999999
  0.13
  0.03839999999999999
  0.01079999999999992
  0.0030839999999999756
  8.784000000000569E-4

Блок-схема


Start
matrix[][], size, err
i=0..size
curVar[i] = matrix[i][size]
j=0..size
i = j
No
curVal[i] = curVal[i] - matrix[i][j]*prevVal[j]
j = j+1
curVal[i] = curVal[i]/matrix[i][i]
No
i = i+1
i=0..size
erroros[i] = abs(curVal[i]-prevVal[i])
error = max(errors)
error < err
Yes
Output curVal, error
End

Вывод

Метод прост в реализации, так же он не ресурсоемкий , так как для реализации метода не нужно хранить всю матрицу в оперативной памяти, а лишь вектора. Так же погрешности не накапливаются, поскольку точность вычислений в каждой итерации определяется разницей между результатами нынешней и прошлой итерации.

Из минусов стоит отметить, что не все СЛАУ можно решить данным методом, можно решить если имеется превосходство диагональных элементов и количество неизвестных равняется количеству уравнений.