Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 1

«Решение системы линейных алгебраических уравнений»

По дисциплине «Вычислительная математика»

Вариант 12

Выполнила:

Студентка группы P3217

Русакова Е.Д.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2024

Оглавление

Цель работы:

Изучить прямые и итерационные методы решения систем линейных алгебраических уравнений, выполнить программную реализацию методов.

Задание:

В программе реализуемый численный метод решения системы линейных алгебраических уравнений (СЛАУ) должен быть реализован в виде отдельного класса /метода/функции, в который исходные/выходные данные передаются в качестве параметров.

Задавать размерность матрицы ($n \leq 20$) из файла или с клавиатуры – по выбору конечного пользователя.

Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

Сформировать не менее 3 файлов (тестов) с различным набором данных.

Программа должна быть протестирована на различных наборах данных, в том числе и некорректных.

Для итерационных методов должно быть реализовано:

1. Начальные приближения, точность, максимальное число итераций задаются с клавиатуры/файла;

2. Проверка достаточного условия сходимости метода. В случае если диагональное преобладание в исходной матрице отсутствует, делать перестановку строк до тех пор, пока преобладание не будет достигнуто. Выводить исходную и преобразованную матрицы;
3. Если достигнуть диагонального преобладания невозможно – выводить соответствующее сообщение. При этом попытаться решить систему линейных уравнений, ограничив итерационный процесс заданным максимальным числом итераций;
4. Вывод вектора неизвестных: $x_1,\ x_2,\ \dots, x_n$;
5. Вывод количества итераций, за которое было найдено решение;
6. Вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$.

Задание для варианта 12:

Метод Гаусса-Зейделя

Описание метода Гаусса-Зейделя:

Метод Гаусса-Зейделя является модификацией метода простой итерации и обеспечивает более быструю сходимость к решению систем уравнений. Строится эквивалентная СЛАУ и за начальное приближение выбирается вектор (обычно это вектор правых частей, но может быть нулевой или единичный). Дальше при вычислении элемента x_i^{k+1} (k+1) итерации используются элементы $x_1^{k+1}, \dots, x_{i-1}^{k+1}$ уже вычисленные на этой итерации и элементы $x_{i+1}^{k}, \dots, x_n^{k+1}$ с предыдущей итерации.

Основная формула метода Гаусса-Зейделя:

$$x_i^{k+1} = \frac{b_i}{a_{ii}} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{k+1} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{k} = d_i - \sum_{i=1}^{i-1} c_{ij} x_j^{k+1} - \sum_{j=i+1}^{n} c_{ij} x_j^{k},$$

где $d_i = \frac{b_i}{a_{ii}}$; $c_{ij} = \begin{cases} 0, & \text{при } i = j \\ -\frac{a_{ij}}{a_{ii}}, & \text{при } i \neq j \end{cases}$; и $i = 1,2, \dots, n$

Условие окончания итерационного процесса:

$$\max_{1 \le i \le n} \left| x_i^k - x_i^{k-1} \right| \le \varepsilon$$

Описание разработанной программы:

Разработанная программа позволяет найти решение СЛАУ, вводимую с клавиатуры или из файла по выбору пользователя. Также пользователь может выбрать нужно ли выводить все итерации решения или только ответ. Программа производит проверку на нулевой детерминант, диагональное преобладание, пытается добиться диагонального преобладание. Еще пользователь может задать максимальное количество итераций.

Исходный код программы:

Полный код программы выложен на Github и доступен по ссылке [lenapochemy/comp-math-lab1: вычмат лаба 1 (github.com)]

Далее приведен код класса GaussSeidelMethod, который отвечает за решение СЛАУ.

GaussSeidelMethod.java

```
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.Arrays;

public class GaussSeidelMethod {

    private final int n;
    private double[][] matrix;
    private final double[] bi;
    private double[] xik;
    private double[] xik_next;
    private final double[] di;
    private final double[][] cij;
    private final double eps;
    private final int m;
    private double[] error_vector;
```

```
    private boolean outputMode;

    public void setOutputMode(boolean outputMode){
        this.outputMode = outputMode;
    }

    public GaussSeidelMethod(int n, double[][] matrix, double[] bi, double
eps, int m){
        this.n = n;
        this.matrix = matrix;
        this.eps = eps;
        this.bi = bi;
        this.m = m;
        this.xik = new double[n];
        this.xik_next = new double[n];
        this.error_vector = new double[n];
        this.di = new double[n];
        this.cij = new double[n][n];
    }

    public void solve(){
        System.out.println("----------------------------------------");
        System.out.println("Матрица получена:");
        for (int i = 0; i < n; i++){
            System.out.println(Arrays.toString(matrix[i]) + " * x = " + bi[i]
);
        }
        if(Det.determinant(matrix) == 0) {
            System.out.println("Данная матрица имеет детерминант, равный
нулю, " +
                    "следовательно СЛАУ или не имеет решений, или имеет
бесконечно много решений");
            System.exit(0);
        }
        System.out.println("----------------------------------------");
        System.out.print("Проверяем диагональное преобладание: ");
        if(checkDiagonalDominance()) System.out.println("выполнено");
        else {
            System.out.println("невыполнено");
            System.out.println("Переставляем строки, чтобы достичь
диагональное преобладание");
            rearrangeLines();
            System.out.println("Диагональное преобладание достигнуто,
приступаем к решению СЛАУ");
            for (int i = 0; i < n; i++){
                System.out.println(Arrays.toString(matrix[i]) + " * x = " +
bi[i] );
            }
        }
        System.out.println("----------------------------------------");


        for(int i = 0; i < n; i++){
            di[i] = bi[i] / matrix[i][i];
        }

        for(int i = 0; i < n; i++){
            for(int j = 0; j < n; j++){
                if(i == j) cij[i][j] = 0;
```

```
                else cij[i][j] = - matrix[i][j] / matrix[i][i];
            }
        }
        //первое приближение
        xik = di;
//        for(int i = 0; i < n; i++){
//            xik[i] = 1000;
//        }

        if(outputMode) {
            System.out.print("Первое приближение:");
            System.out.println(Arrays.toString(xik));
            System.out.println("----------------------------------------");
        }

        for(int i = 0; i < m; i++){
            if(outputMode) {
                System.out.println("Итерация № " + (i + 1));
            }
            iteration();
            if(endConditional()) {
                System.out.println("СЛАУ решена!");
                System.out.println("Решение: " + Arrays.toString(xik_next));
                System.out.println("Вектор погрешностей: " +
Arrays.toString(error_vector));
                System.out.println("Количество итераций: " + (i + 1));
                System.exit(0);
            } else {
                if(outputMode) {
                    System.out.println("Новое приближение:");
                    System.out.println(Arrays.toString(xik_next));
                    System.out.println("-------------------------------------
----");
                }
                xik = xik_next;
                xik_next = new double[n];
            }

        }

        System.out.println("За данное количество итераций решение СЛАУ не
было найдено, вожможно уравнение расходится и решений вообще нет");

    }

    private void iteration(){
        for(int i = 0; i < n; i++){
            double sum1 = 0, sum2 = 0;
            xik_next[i] = di[i];
            for(int j = 0; j < i ; j++){
                xik_next[i] += cij[i][j] * xik_next[j];
            }
            for(int j = i + 1; j < n; j++){
                xik_next[i] += cij[i][j] * xik[j];
            }
//            xik_next[i] = rounding(xik_next[i]);
        }
    }
```

```
    private double rounding(double number){
        BigDecimal help = new BigDecimal(number);
        help = help.setScale(4, RoundingMode.HALF_UP);
        return help.doubleValue();
    }

    private boolean checkDiagonalDominance(){
        for(int i = 0; i < n; i++){
            double sum = 0;
            for (int j = 0; j < n; j++){
                sum += Math.abs(matrix[i][j]);
            }
            sum -= Math.abs(matrix[i][i]);
            if(Math.abs(matrix[i][i]) < sum) return false;
        }
        return true;
    }

    private boolean checkLineDiagonalDominance(double[] line, int i){
        double sum = 0;
        for(int j = 0; j < n; j++){
            sum += Math.abs(line[j]);
        }
        sum -= Math.abs(line[i]);
        return Math.abs(line[i]) >= sum;
    }

    private void rearrangeLines(){
        boolean lineDominance = true;
        for(int i = 0; i < n && lineDominance; i++){
            if(!checkLineDiagonalDominance(matrix[i], i)){
                lineDominance = false;
                for(int j = i + 1; j < n; j++){
                    double[] help;
                    double bhelp;
                    if(checkLineDiagonalDominance(matrix[j], i)){
                        help = matrix[i];
                        matrix[i] = matrix[j];
                        matrix[j] = help;
                        bhelp = bi[i];
                        bi[i] = bi[j];
                        bi[j] = bhelp;
                        lineDominance = true;
                    }
                }
            }
        }
        if(!lineDominance) {
            System.out.println("Невозможно добиться диагонального
преобладания в данной матрице");
            System.exit(0);
        }
    }

    //вернет true, если условие окончания выполняется и это была последняя
итерация
    private boolean endConditional(){
        double error;
        boolean result = true;
```

```
        for(int i = 0; i < n; i++){
//              error = rounding(Math.abs(xik_next[i] - xik[i]));
            error = Math.abs(xik_next[i] - xik[i]);
            error_vector[i] = error;
            if(error > eps) result = false;
        }
        if(outputMode) {
            System.out.println("Максимальная погрешность: " +
Arrays.stream(error_vector).max().getAsDouble());
        }
        return result;
    }
}
```

Примеры работы программы:

Пример 1

```
Введите размерность матрицы: 3
Введите коэфициенты матрицы по строкам через пробел:
[2, 2, 10 ]
[10, 1, 1 ]
[2, 10, 1 ]
Введите правые части уравнений по одному:
14
12
13
Введите значение точности [0.000001; 1]: 0.01
Введите максимальное количество итераций: 30
Нужно выводить результат каждой итерации решения? (y/n)y
----------------------------------------
Матрица получена:
[2.0, 2.0, 10.0] * x = 14.0
[10.0, 1.0, 1.0] * x = 12.0
[2.0, 10.0, 1.0] * x = 13.0
----------------------------------------
Проверяем диагональное преобладание: невыполнено
Переставляем строки, чтобы достичь диагональное преобладание
Диагональное преобладание достигнуто, приступаем к решению СЛАУ
[10.0, 1.0, 1.0] * x = 12.0
[2.0, 10.0, 1.0] * x = 13.0
[2.0, 2.0, 10.0] * x = 14.0
----------------------------------------
```

```
Первое приближение:[1.2, 1.3, 1.4]
----------------------------------------
Итерация № 1
Максимальная погрешность: 0.3808
Новое приближение:
[0.9299999999999998, 0.9740000000000001, 1.0191999999999999]
----------------------------------------
Итерация № 2
Максимальная погрешность: 0.07068000000000019
Новое приближение:
[1.00068, 0.9979439999999999, 1.0002752]
----------------------------------------
Итерация № 3
Максимальная погрешность: 0.001992863999999983
СЛАУ решена!
Решение: [1.00017808, 0.9999368639999999, 0.9999770111999998]
Вектор погрешностей: [5.019200000000446E-4, 0.001992863999999983, 2.9818880000009873E-4]
Количество итераций: 3
```

Пример 2

```
Введите размерность матрицы: 3
Введите коэфициенты матрицы по строкам через пробел:
[5.5, 1.6, 1.7 ]
[0.8, 3.4, 0.9 ]
[2.4, -2, -4.5 ]
Введите правые части уравнений по одному:
1
3
-1.5
Введите значение точности [0.000001; 1]: 0.01
Введите максимальное количество итераций: 20
Нужно выводить результат каждой итерации решения? (y/n)n
----------------------------------------
Матрица получена:
[5.5, 1.6, 1.7] * x = 1.0
[0.8, 3.4, 0.9] * x = 3.0
[2.4, -2.0, -4.5] * x = -1.5
----------------------------------------
Проверяем диагональное преобладание: выполнено
----------------------------------------
СЛАУ решена!
Решение: [-0.05257871873888444, 0.9230619607897879, -0.10495841034508852]
Вектор погрешностей: [0.006909681991120117, 0.0035788124303647306, 0.0020945804262130774]
Количество итераций: 4
```

Пример 3

```
Введите размерность матрицы: 4
Введите коэфициенты матрицы по строкам через пробел:
[3, 2, 1, 1 ]
[1, -1, 4, -1 ]
[-2, -3, -2, 1 ]
[1, 1, -1, 2 ]
Введите правые части уравнений по одному:
-2
-1
9
4
Введите значение точности [0.000001; 1]: 0.01
Введите максимальное количество итераций: 20
Нужно выводить результат каждой итерации решения? (y/n)y
----------------------------------------
Матрица получена:
[3.0, 2.0, 1.0, 1.0] * x = -2.0
[1.0, -1.0, 4.0, -1.0] * x = -1.0
[-2.0, -3.0, -2.0, 1.0] * x = 9.0
[1.0, 1.0, -1.0, 2.0] * x = 4.0
----------------------------------------
Проверяем диагональное преобладание: невыполнено
Переставляем строки, чтобы достичь диагональное преобладание
Невозможно добиться диагонального преобладания в данной матрице
```

Пример 4

```
Введите размерность матрицы: 3
Введите коэфициенты матрицы по строкам через пробел:
[4, 2, 1 ]
[1, -1, 4 ]
[0, 0, 0 ]
Введите правые части уравнений по одному:
-2
-1
9
Введите значение точности [0.000001; 1]: 0.01
Введите максимальное количество итераций: 100
Нужно выводить результат каждой итерации решения? (y/n)y
----------------------------------------
Матрица получена:
[4.0, 2.0, 1.0] * x = -2.0
[1.0, -1.0, 4.0] * x = -1.0
[0.0, 0.0, 0.0] * x = 9.0
Данная матрица имеет детерминант, равный нулю, следовательно СЛАУ или не имеет решений, или имеет бесконечно много решений
```

Вывод:

При выполнение лабораторной работы я познакомилась с различными методами решения систем линейных арифметических уравнений и выполнила программную реализацию метода Гаусса-Зейделя.