Министерство высшего образования и науки Российской Федерации
Национальный научно-исследовательский университет ИТМО
Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по дисциплине

«Вычислительная математика».

Вариант №1.

Работу выполнил:
Афанасьев Кирилл Александрович,
Студент группы P3206.
Преподаватель:
Рыбаков Степан Дмитриевич.

Санкт-Петербург, 2024

Оглавление

Задание

Реализовать консольное приложение, вычисляющее решение системы линейных алгебраических уравнений (СЛАУ).

1. № варианта определяется как номер в списке группы согласно ИСУ.
2. В программе численный метод должен быть реализован в виде отдельной подпрограммы/метода/класса, в который исходные/выходные данные передаются в качестве параметров.
3. Размерность матрицы n<=20 (задается из файла или с клавиатуры – по выбору конечного пользователя).
4. Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

Для прямых методов должно быть реализовано:

- Вычисление определителя
- Вывод треугольной матрицы (включая преобразованный столбец B)
- Вывод вектора неизвестных: x_1, x_2, ..., x_n
- Вывод вектора невязок: r_1 , r_2, ..., r_n

Описание метода

Вариант 1: Метод Гаусса (прямой метод).

Основан на приведении матрицы системы к треугольному виду так, чтобы ниже ее главной диагонали находились только нулевые элементы.
Прямой ход метода Гаусса состоит в последовательном исключении неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается $x1$ из всех последующих уравнений системы. Затем с помощью второго уравнения исключается $x2$ из третьего и всех последующих уравнений и т. д.
Этот процесс продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным xn , т. е. матрица системы будет приведена к треугольному виду.
Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее уравнение, находим единственное в этом уравнении неизвестное xn. Далее, используя это значение, из предыдущего уравнения вычисляем $xn-1$ и т. д. Последним найдем $x1$ из первого уравнения.
Метод имеет много различных вычислительных схем, но в каждой из них основным требованием является $\det A \neq 0$.

Блок-схема:

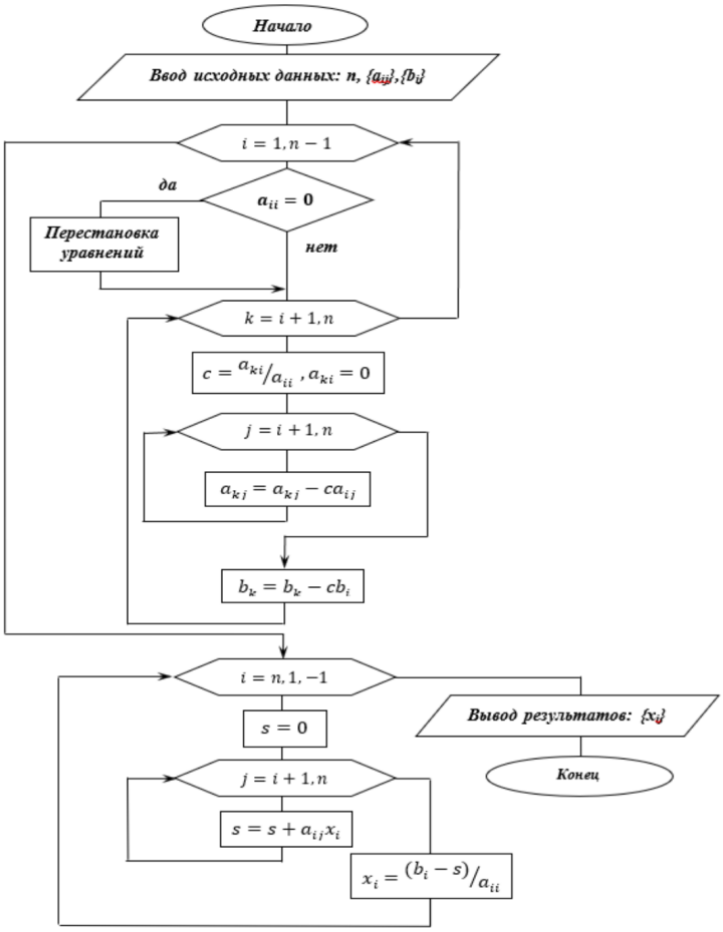

Рисунок 1. Блок-схема метода Гаусса.

Исходный код программы

Листинг вычисления определителя матрицы:

```
fun matrixDeterminant(matrix: Matrix): BigDecimal {
    val matrixClone = Matrix(matrix)
    if (!bringMatrixToValidForm(matrixClone)) return
BigDecimal.ZERO.setScale(matrix.getValueScale())
    triangleMatrix(matrixClone)
    var result = BigDecimal("1").setScale(matrix.getValueScale())
    for (i in 0..<matrixClone.getDimension()) {
        result = result.multiply(matrixClone.getMatrixElement(i, i))
            .setScale(matrixClone.getValueScale(), RoundingMode.HALF_UP)
    }
    if (matrixClone.getSwapCount() % 2 != 0) result = result.negate()
    return result
}
```

Листинг подпрограммы перестановки уравнений:

```
fun bringMatrixToValidForm(matrix: Matrix): Boolean {
    for (i in 0..<matrix.getDimension()) {
        if (matrix.getMatrixElement(i, i) ==
BigDecimal.ZERO.setScale(matrix.getValueScale())) {
            var swappedSuccess = false
            for (j in 0..<matrix.getDimension()) {
                if (matrix.getMatrixElement(j, i) !=
BigDecimal.ZERO.setScale(matrix.getValueScale())) {
```

```
                    if (matrix.getMatrixElement(i, j) !=
BigDecimal.ZERO.setScale(matrix.getValueScale())) {
                        matrix.swapRows(i, j)
                        swappedSuccess = true
                        break
                    }
                }
            }
            if (!swappedSuccess) return false
        }
    }
    return true
}
```

Листинг прямого хода:

```
fun triangleMatrix(matrix: Matrix) {
    assert(matrix.getMatrixElement(0, 0) !=
BigDecimal("0").setScale(matrix.getValueScale()))

    for (i in 0..<matrix.getDimension()) {
        for (j in i + 1..<matrix.getDimension()) {

            if (matrix.getMatrixElement(i, i) ==
BigDecimal.ZERO.setScale(matrix.getValueScale())) continue

            val firstVector = matrix.getMatrixRow(i)

            val removingCoefficient =
                matrix.getMatrixElement(j,
i).divide(matrix.getMatrixElement(i, i), RoundingMode.HALF_UP)
                    .negate()

            val modifiedVector = firstVector.map {

it.multiply(removingCoefficient).setScale(matrix.getValueScale(),
RoundingMode.HALF_UP)
            }.toTypedArray()

            matrix.applyVectorToRow(j, modifiedVector, BigDecimal::plus)
        }
    }
}
```

Листинг обратного хода:

```
private fun solveForDiagMatrix(): Array<BigDecimal> {

    val solution = Array(matrix.getDimension()) {
        BigDecimal("0").setScale(matrix.getValueScale())
    }

    for (i in matrix.getDimension() - 1 downTo 0) {
        var sum = BigDecimal("0").setScale(matrix.getValueScale())
        for (j in i + 1..<matrix.getDimension()) {
            sum = sum.add(
                matrix.getMatrixElement(i, j).multiply(solution[j])
```

```
                    .setScale(matrix.getValueScale(), RoundingMode.HALF_UP)
            )
        }

        solution[i] = matrix.getMatrixElement(i,
matrix.getDimension()).subtract(sum)
            .divide(matrix.getMatrixElement(i, i), RoundingMode.HALF_UP)
    }

    return solution
}
```

Листинг вычисления невязок:

```
fun calculateResidualVector(): Array<BigDecimal> {
    assert(slaeSolution.status == SLAESolutionStatus.OK)

    val result = Array<BigDecimal>(slaeSolution.solutionVector.size) {
BigDecimal.ZERO }

    for (i in 0..<slaeSolution.solutionVector.size) {
        val slaeVector = slaeSolution.sourceSystem.getMatrixRow(i)
        var solutionResult =
BigDecimal.ZERO.setScale(slaeSolution.sourceSystem.getValueScale())
        for (j in 0..<slaeSolution.solutionVector.size) {
            solutionResult = solutionResult.add(
                    slaeVector[j].multiply(slaeSolution.solutionVector[j])
                        .setScale(slaeSolution.sourceSystem.getValueScale(),
RoundingMode.HALF_UP)
                )
        }
        result[i] =
slaeVector[slaeSolution.solutionVector.size].subtract(solutionResult).abs()
    }

    return result
}
```

GitHub: https://github.com/Zerumi/cmath1_170224_1/

Тесты

Для программы написано 17 Unit тестов. Далее приведены некоторые выводы программы:

```
--------------------------------------------------------------------------------
SLAE Matrix:
10.00 -7.00 0.00 7.00
-3.00 2.00 6.00 4.00
5.00 -1.00 5.00 6.00
Triangle Matrix:
10.00 -7.00 0.00 7.00
0.00 -0.10 6.00 6.10
0.00 0.00 155.00 155.00
Found a solution
Solution | Residual Error:
x1: 0.000000000000000000000000000000 | 0.000000000000000000000000000000
x2: -1.000000000000000000000000000000 | 0.000000000000000000000000000000
x3: 1.000000000000000000000000000000 | 0.000000000000000000000000000000
--------------------------------------------------------------------------------
```

```
SLAE Matrix:
10.00 -7.00 0.00 7.00
-3.00 2.10 6.00 3.90
5.00 -1.00 5.00 6.00
Triangle Matrix:
10.00 -7.00 0.00 7.00
0.00 0.00 6.00 6.00
0.00 0.00 15005.00 15005.00
Found a solution
Solution | Residual Error:
x1: 0.0000000000000000000000000000000000 | 0.0000000000000000000000000000000000
x2: -1.0000000000000000000000000000000000 | 0.0000000000000000000000000000000000
x3: 1.0000000000000000000000000000000000 | 0.0000000000000000000000000000000000
--------------------------------------------------------------------------------
SLAE Matrix:
3.00 -2.00 -6.00
5.00 1.00 3.00
Triangle Matrix:
3.00 -2.00 -6.00
0.00 4.33 13.00
Found a solution
Solution | Residual Error:
x1: 0.0000000000000000000000000000000000 | 0.0000000000000000000000000000000000
x2: 3.0000000000000000000000000000000000 | 0.0000000000000000000000000000000000
--------------------------------------------------------------------------------
SLAE Matrix:
5.00 2.00 7.00
2.00 1.00 9.00
Triangle Matrix:
5.00 2.00 7.00
0.00 0.20 6.20
Found a solution
Solution | Residual Error:
x1: -11.000000000000000000000000000000000 | 0.0000000000000000000000000000000000
x2: 31.000000000000000000000000000000000 | 0.0000000000000000000000000000000000
--------------------------------------------------------------------------------
SLAE Matrix:
2.00 1.00 1.00 2.00
1.00 -1.00 0.00 -2.00
3.00 -1.00 2.00 2.00
Triangle Matrix:
2.00 1.00 1.00 2.00
0.00 -1.50 -0.50 -3.00
0.00 0.00 1.33 4.00
Found a solution
Solution | Residual Error:
x1: -1.000000000000000000000000000000000 | 0.0000000000000000000000000000000001
x2: 1.000000000000000000000000000000000 | 0.0000000000000000000000000000000000
x3: 2.999999999999999999999999999999999 | 0.0000000000000000000000000000000001
--------------------------------------------------------------------------------
SLAE Matrix:
2.00 1.00 9.00
5.00 2.00 7.00
Triangle Matrix:
2.00 1.00 9.00
0.00 -0.50 -15.50
Found a solution
Solution | Residual Error:
x1: -11.000000000000000000000000000000000 | 0.0000000000000000000000000000000000
x2: 31.000000000000000000000000000000000 | 0.0000000000000000000000000000000000
--------------------------------------------------------------------------------
```

```
SLAE Matrix:
3.00 -2.00 -1.00
1.00 3.00 7.00
Triangle Matrix:
3.00 -2.00 -1.00
0.00 3.67 7.33
Found a solution
Solution | Residual Error:
x1: 1.000000000000000000000000000000000 | 0.000000000000000000000000000000000
x2: 2.000000000000000000000000000000000 | 0.000000000000000000000000000000000
--------------------------------------------------------------------------------
SLAE Matrix:
5.00 1.00 -6.00
5.00 1.00 3.00
Triangle Matrix:
5.00 1.00 -6.00
0.00 0.00 9.00
SLAE has no solutions
--------------------------------------------------------------------------------
SLAE Matrix:
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
Triangle Matrix:
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
SLAE matrix was invalid or SLAE has uncountable amount of solutions
--------------------------------------------------------------------------------
SLAE Matrix:
1.00 2.00 3.00 4.00 0.00
1.00 2.00 3.00 4.00 0.00
1.00 2.00 3.00 4.00 0.00
1.00 2.00 3.00 4.00 0.00
Triangle Matrix:
1.00 2.00 3.00 4.00 0.00
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00
SLAE has uncountable amount of solutions
--------------------------------------------------------------------------------
SLAE Matrix:
1.00 2.00 3.00 4.00 1.00
1.00 2.00 3.00 4.00 2.00
1.00 2.00 3.00 4.00 3.00
1.00 2.00 3.00 4.00 4.00
Triangle Matrix:
1.00 2.00 3.00 4.00 1.00
0.00 0.00 0.00 0.00 1.00
0.00 0.00 0.00 0.00 2.00
0.00 0.00 0.00 0.00 3.00
SLAE has no solutions
--------------------------------------------------------------------------------
```

Вывод

Во время выполнения данной лабораторной работы я ознакомился с предметом вычислительной математики: численные методы. А для применения знаний на практике мною был запрограммирован один из таких методов: метод Гаусса для решения СЛАУ.

Также я поработал с вычислительными погрешностями при реализации алгоритма, на практике оценил их влияние на конечный результат и на последующую разработку тестов к написанной программе.