Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление системного и прикладного программного обеспечения

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

курса «Вычислительная математика»

по теме: «Решение системы линейных алгебраических уравнений СЛАУ»

Вариант № 2 (№ 22 в списке группы)

Выполнил студент:

Тюрин Иван Николаевич

группа: P32131

Преподаватель:

Малышева Т. А.,

Бострикова Д. К.

Санкт-Петербург, 2023 г.

Содержание

Лабораторная работа № 1
Решение системы линейных алгебраических уравнений СЛАУ

1. Задание варианта № 2 (№ 22 в списке группы)

, , ,

Метод Гаусса с выбором главного элемента по столбцам

, , ,

2. Цель работы

Изучить способы численных методов решения системы линейных алгебраических уравнений и реализовать один из них.

3. Описание метода, расчетные формулы

Метод Гаусса с выбором главного элемента по столбцам — прямой метод решения СЛАУ. Метод состоит из двух этапов:

1. подготовка матрицы («прямой ход»),
2. вычисление вектора переменных («обратный ход»).

Во время первого этапа происходит приведение данной квадратной матрицы к треугольному виду с помощью последовательного исключения переменных из всех нижележащих строчек. Математически шаг с исключением переменной x_i из уравнений начиная с $i+1$ можно описать, как

$$a'_{kj} = a_{kj} - \frac{a_{ki}}{a_{ii}} a_{kj},\ \forall i, j \in [1..n],\ k \in [(i+1)..n]. \tag{1.1}$$

При этом модификация *выбора главного элемента по столбцам* заключается в том, что перед началом исключения мы, если это нужно, переставливаем две строчки в матрице так, чтобы главный элемент в текущей строке был как можно большим по абсолютному значению, т.е. $a_{ii} \geq a_{ji}\ \forall i \in [1..n],\ j \in [(i+1)..n]$. Этот действие позволяет повысить точность вычислений на компьютере, т.к. таким образом мы стараемся избавиться от погрешности от деления на близкое к нулю число. В добавок к этому, посчитав количество перестановок строчек, мы можем установить знак определителя нашей изначальной матрицы (его значение, с точностью до знака, мы можем вычислить перемножив элементы на диагонали полученной треугольной матрицы), а именно

$$\det A = (-1)^k \prod_{i=1}^{n} a_{ii}, \tag{1.2}$$

где k — число перестановок строк (или столбцов) матрицы при ее приведении к треугольному виду в соответствии с модификацией.

3.1. Листинг программы

Основную часть программной реализации на языке программировани Kotlin можно посмотреть в листинге 1.1. Весь код представлен в личном репозитории [1].

```
fun Double.approx(double: Double): Boolean {
    return (this - double).absoluteValue < 1e-6
}

/**
 * extension function on matrix to solve system of linear equations described by this and vector `b`.
 */
@OptIn(ExperimentalStdlibApi::class)
fun Matrix.solveSLE(b: DoubleArray, logMiddleResults: Boolean = false): DoubleArray? {
    val dim = elements.size // matrix must be square unless we can slice square matrix from it
    assert(elements.all { dim == it.size }) { "Matrix dimensions mismatch (not square matrix)"}
    assert(b.size == dim) { "Matrix and vector dimensions mismatch (must have same number of rows)" }

    val xs = DoubleArray(dim) { 0.0 }
    val tmp = elements.copy().toMutableMatrix()

    var nRowSwaps = 0;

    // Forward
    for (x in 0..<dim - 1) {
        if (tmp.mutateMatrixWithVector(vector = b, start = x)) {
            nRowSwaps++
        }
```

```
24
25          for (nextRow in x + 1..<dim) {
26              val mul = -tmp.elements[nextRow][x] / tmp.elements[x][x]
27              for (i in 0..<dim) {
28                  tmp.elements[nextRow][i] += tmp.elements[x][i] * mul
29              }
30              b[nextRow] += b[x] * mul
31
32              tmp.elements[nextRow][x] = 0.0 // must be 0 because of math
33          }
34      }
35
36      if (logMiddleResults) {
37          printSep()
38          println("Triangle matrix:")
39          println(tmp.pretty())
40          printSep()
41          println("Modified b:")
42          println(b.pretty())
43      }
44
45      val det: Double = if (nRowSwaps % 2 == 0) {
46          1.0
47      } else {
48          -1.0
49      } * tmp.elements.foldIndexed(1.0) { ii, acc, a ->
50          acc * a[ii]
51      }
52      if (logMiddleResults) {
53          printSep()
54          println("Det(Matrix)=$det")
55      }
56
57      if (det.approx(0.0)) {
58          println("Approximately zero determinant stands for 0 or infinite
    number of solutions")
59          return null
60      }
61
62      // Backward
63      for (x in xs.indices.reversed()) {
64          var tmpSum: Double = 0.0
65          for (i in x + 1..xs.lastIndex) {
66              tmpSum += tmp.elements[x][i] * xs[i]
67          }
68          xs[x] = (b[x] - tmpSum) / tmp.elements[x][x]
69      }
70
71      return xs
72  }
73
74  /**
75   * Mute matrix to get greater element at first position
76   */
77  @OptIn(ExperimentalStdlibApi::class)
78  private fun MutableMatrix.mutateMatrixWithVector(vector: DoubleArray,
    start: Int): Boolean {
79      var mutated = false
80      var greatest: Int = start
81      for (i in start..<elements.size) {
```

```
82          if (elements[i][start].absoluteValue > elements[greatest][start].
    absoluteValue) {
83              greatest = i
84              mutated = true
85          }
86      }
87      if (mutated) {
88          elements[start] = elements[greatest].also { elements[greatest] =
    elements[start] }
89          vector[start] = vector[greatest].also { vector[greatest] = vector[
    start] }
90      }
91      return mutated
92  }
```

Листинг 1.1: Реализация на языке программирования Kolin основной логики решания СЛАУ

4. Примеры и результаты работы программы

В утилите реализована возможность ввода данных через файл специального формата даннных, заполнение матрицы случайными числами и режим интерактивного ввода.

4.1. Листинг тестов

Для проверки работоспособности программы были написаны тесты, их содержимое представлено в листинге 1.2.

```
1
2   class Test {
3       @Test
4       fun `input data from file`() {
5           println("####### input data from file #######")
6           main(arrayOf("--file", "D:\\Projects\\itmo-comp-math\\lab-1\\src\\
    test\\resources\\file-input-test.txt"))
7       }
8
9       @Test
10      fun `input data by random dim=3`() {
11          println("####### input data by random #######")
12          main(arrayOf("--random", "3"))
13      }
14
15      @Test
16      fun `print documentation`(){
17          println("####### print documentation #######")
18          main(arrayOf("--help"))
19      }
20  }
```

Листинг 1.2: Реализация на языке программирования Kolin тестов утилиты для решения СЛАУ

4.2. Листинг результатов работы тестов

Соответственно вывод результатов тестов представлен в листинге 1.3. Текстовый файл использовавшийся при тестировании так же можно найти среди ресурсов для тестов в личном репозитории [1].

```
####### input data from file #######

Triangle matrix:
[[10.0, -7.0, 0.0],
 [0.0,  2.5,  5.0],
 [0.0,  0.0,  6.002]]

Modified b:
[7.0,
 2.5,
 6.001999999999999]

Det(Matrix)=-150.04999999999998

Result of calculating vector 'x':
[2.6645352591003756E-16,
 -0.9999999999999997,
 0.9999999999999999]

Vector of deviance 'r':
[0.0,
 -1.400999999999999,
 0.0019999999999988916]
####### print documentation #######

DESCRIPTION
    Utility for solving System of Liner Algebraic Equations

    NOTE: floating point is comma (",")

SYNOPSIS
    solve [FLAG [PARAM]]

FLAG
    -h, --help          Prints help doc of utility;

    --file      [path]  Receives ‘path‘ to the input file, works in not
   interactive mode;

    --random    [n]     Receives number of matrix dimensions ‘n‘;

####### input data by random #######
#-------------------------------------------------------#
# Your are welcome in System of Linear Equations solver. #
#-------------------------------------------------------#
Please input your matrix dimension:
( 0 < n < 21 )
 n= 3
Please, input your (n x n) values of matrix by rows of n elements
   separated with single space:

Thank you for your matrix:
[[0.47605838251574817,  0.8113645483712928, 0.26142577878135476],
 [0.19243939663868048,  0.30967754101767, 0.12289604698407952],
```

```
 [0.6040888727423883, 0.3408073537056102, 0.6314133409219941]]

Next step is input values of your system's right-hand part in one row
    separated with single space:

Thank you for your vector:
[0.14767079427067065,
 0.43526862948970035,
 0.4361731660679654]

Triangle matrix:
[[0.6040888727423883, 0.3408073537056102, 0.6314133409219941],
 [0.0,  0.5427878457115936, -0.23616592889834392],
 [0.0,  0.0,  0.009254467679650977]]

Modified b:
[0.4361731660679654,
 -0.19605990712835591,
 0.36896326648887656]

Det(Matrix)=0.003034466822004519

Result of calculating vector 'x':
[-50.532697350685176,
 16.985568739373555,
 39.868664439788894]

Vector of deviance 'r':
[0.28850237179728977,
 -0.6313285366180561,
 -0.06720989957909007]
```

Листинг 1.3: Вывод с тандартный поток тестов утилиты для решения СЛАУ

5. Вывод

В ходе выполнения данной лабораторной работы углубили понимание работы методов решения СЛАУ, реализовали на языке Kotlin требуемую утилиту для их решения.
При решении СЛАУ использовался метод Гаусса с выбором наибольшего элемента по строкам, обычный метод Гаусса оказалось достаточно легко реализовать и потом не трудно модифициировать до метода с выбором максимального элемента. Погрешности вычисления которые возникают при работе его реализации достаточно малы при не очень больших по модулю значениях, в противном случае возникает ошибка представления дробных чисел в копьютере: не хватает точности для больших по модулю чисел. Эта погрешность возникает потому что требуется перемножать много чисел во время расчетов.

Литература

[1] Ссылка на личный репозиторий GitHub: https://github.com/e1turin/itmo-comp-math/tree/main/lab-1