Национальный исследовательский университет информационных технологий, механики и оптики

Факультет Программной Инженерии и Компьютерной Техники

Вариант №17

Лабораторная работа №1

«Решение системы линейных алгебраических уравнений СЛАУ»

По дисциплине:

«Вычислительная математика»

Работу выполнила:

Студентка группы Р3212

Никонова Наталья Игоревна

Преподаватель:

Малышева Татьяна Алексеевна

Санкт-Петербург

2022

Цель работы

Познакомится с вычислительной математикой и численными методами на примере решения системы линейных алгебраических уравнений. Запрограммировать решение СЛАУ по методу, обозначенному в задании.

Задание

Написать программу, вычисляющую решение СЛАУ методом Гаусса с выбором главного элемента по столбцам, и, кроме того, выводящую определитель матрицы, её треугольную форму (включая столбец B, т.е. расширенную) и вектор невязок.

Описание метода, расчетные формулы

Этот метод является модификацией обычного метода Гаусса. Она заключается в том, что помимо проверки ведущего элемента на неравенство нулю, среди оставшихся уравнений ищется такое, у которого элемент, стоящий на позиции ведущего, максимален по модулю среди остальных. Если такое уравнение находится, то оно меняется местами с текущим. Далее же выполняется итерация прямого хода обычного метода Гаусса: для каждой из последующих строк рассчитывается множитель $-\frac{a_{i+k,i}}{a_{i,i}}$.На него умножается текущая строка и добавляется к последующей, в результате чего элемент в том же столбце, что и ведущий, зануляется, а остальные преобразовываются. Таким образом матрица приводится к треугольному виду.

Каждый элемент матрицы рассчитывается по формулам:

$$a_{ij}^{(1)} = a_{ij} - \frac{a_{i1}}{a_{11}} a_{1j}\ , i, j = 2,3 \dots n$$
$$b_i^{(1)} = b_i - \frac{a_{i1}}{a_{11}} b_1\ , i = 2,3 \dots n$$

Далее следует обратный ход обычного метода Гаусса без каких-либо изменений. Моя программа рассчитывает каждую неизвестную по следующей формуле:

$$x_i = \frac{b_i - (a_{i,n} * x_n + \cdots + a_{i,1} * x_1)}{a_{i,i}}$$

При этом считая, что ещё не полученные неизвестные равны нулю. Это помогает рассчитывать все неизвестные однообразно, но верно.

Листинг программы

Полный исходный код: https://github.com/nanikon/computational_math

Основной метод, в котором происходит вычисление, GaussWithMainInColumn.kt (все рассчитывается при инициализации объекта этого класса):

```kotlin
private fun compute() {
    // Выборка главного элемента по столбцам и прямой ход Гаусса
    for (i in 0 ≤ until < matrix.dim) {
        val indexRowWithMaxElem = maxElemInColumn(i)
        if (indexRowWithMaxElem != i) {
            matrix.swapRows(i, indexRowWithMaxElem)
            countReplace++
        }
        for (j in i + 1 ≤ until < matrix.dim) {
            if (matrix.getElem(i, i) == BigDecimal( val: 0)) {
                println("Максимальный по модулю ведущий элемент 0. У системы не будет единственного решения, завершение программы"
                exitProcess( status: -1)
            }
            val multiplier = -matrix.getElem(j, i).divide(matrix.getElem(i, i))
            matrix.addMultipliedFirstToSecond(i, j, multiplier)
        }
    }
    // Обратный ход Гаусса и получение решения
    for (i in matrix.dim - 1 ≥ downTo ≥ 0) {
        val x = (matrix.getElem(i, matrix.dim) - matrix.multiplyRowByVectorAndSum(i, solution)) / matrix.getElem(i, i)
        solution[i] = x
    }
    // Рассчет неувязок
    for (i in 0 ≤ until < matrix.dim) {
        discrepancies.add(matrix.getElem(i, matrix.dim) - matrix.multiplyRowByVectorAndSum(i, solution))
    }
}
```

Поиск строки с максимальным ведущим элементом и вычисление определителя (тот же файл)

```kotlin
private fun maxElemInColumn(index: Int) : Int {
    var maxElem = matrix.getElem(index, index)
    var result = index
    for (j in index + 1 ≤ until < matrix.dim) {
        val curElem = matrix.getElem(j, index)
        if (curElem.abs() > maxElem.abs()) {
            maxElem = curElem
            result = j
        }
    }
    return result
}

override fun getDeterminant(): BigDecimal {
    var result = if (countReplace % 2 == 0) { BigDecimal( val: 1) } else { -BigDecimal( val: 1) }
    for (i in 0 ≤ until < matrix.dim) { result = result.multiply(matrix.getElem(i, i)) }
    return result
}
```

```kotlin
fun swapRows(i: Int, j: Int) {
    val tmpRow = data[i].toMutableList()
    data[i] = data[j].toMutableList()
    data[j] = tmpRow
}

fun addMultipliedFirstToSecond(first: Int, second: Int, multiplier: BigDecimal) {
    for (i in 0 ≤ .. ≤ dim) {
        data[second][i] = data[second][i].add(multiplier.multiply(data[first][i]))
    }
}

fun multiplyRowByVectorAndSum(index: Int, vector: List<BigDecimal>) : BigDecimal {
    return data[index].foldIndexed(BigDecimal( val: 0)) { idx, acc, elem -> if (idx != dim) acc + elem * vector[idx] else acc }
}
```

Вспомогательные функции при работе с матрицей (файл Matrix.kx)

```
fun swapRows(i: Int, j: Int) {
    val tmpRow = data[i].toMutableList()
    data[i] = data[j].toMutableList()
    data[j] = tmpRow
}

fun addMultipliedFirstToSecond(first: Int, second: Int, multiplier: BigDecimal) {
    for (i in 0 ≤ .. ≤ dim) {
        data[second][i] = data[second][i].add(multiplier.multiply(data[first][i]))
    }
}

fun multiplyRowByVectorAndSum(index: Int, vector: List<BigDecimal>) : BigDecimal {
    return data[index].foldIndexed(BigDecimal( val: 0)) { idx, acc, elem -> if (idx != dim) acc + elem * vector[idx] else acc }
}
```

Основная функция программы

```
fun main() {
    val dimParser = choseParser( obj: "размерности матрицы")
    val n = dimParser.parseDim()
    println("Введена размерность $n")
    val matrixParser = choseParser( obj: "матрицы")
    val matrix = matrixParser.parseMatrix(n)
    println("Введена следующая матрица:")
    matrix.printMatrix()
    val computer = GaussWithMainInColumn(matrix)
    println("Определитель: ${computer.getDeterminant()}")
    println("Треугольная матрица:")
    computer.getTriangularMatrix().printMatrix()
    println("Решение:")
    println(computer.getSolution().joinToString(separator = "\n"))
    println("Невязки:")
    println(computer.getDiscrepancies().joinToString(separator = "\n"))
}
```

Результаты работы программы

```
MainKt
"C:\Program Files\Java\jdk-14.0.2\bin\java.exe" -javaagent:C:\Users\natan\AppData\Local\JetBrains\
Для ввода размерности матрицы с клавиатуры введите 1, для ввода с файла - 2
2
Введите имя файла:
input_dim.txt
7
Введена размерность 7
Для ввода матрицы с клавиатуры введите 1, для ввода с файла - 2
2
Введите имя файла:
input_matrix.txt
Введена следующая матрица:
-48,20000   36,70000    25,80000    31,40000    -37,60000   -11,50000   43,70000    -966,00000
47,60000    -42,30000   48,30000    -14,60000   50,90000    -31,20000   20,90000    493,50000
17,60000    -12,80000   -6,10000    24,00000    32,40000    41,00000    40,70000    -782,60000
-16,10000   8,40000 2,30000 -32,00000   49,40000    -46,20000   -5,50000    -2858,50000
15,10000    -10,70000   -39,80000   -23,10000   -25,50000   -31,70000   46,60000    -1287,20000
26,40000    9,70000 -33,10000   -38,50000   -1,40000    -23,50000   -31,90000   -1219,40000
44,00000    50,00000    -44,90000   -38,30000   17,90000    -24,80000   38,10000    -5620,10000
Определитель: 609051632963.583
```

```
Треугольная матрица:
-48,20000   36,70000    25,80000    31,40000    -37,60000   -11,50000   43,70000    -966,00000
0,00000 83,50207    -21,34813   -9,63610    -16,42365   -35,29793   77,99212    -6501,92573
0,00000 0,00000 72,23035    15,71017    12,57676    -45,11719   69,71320    -932,09411
0,00000 0,00000 0,00000 -41,34497   62,47222    -48,55238   -9,44298    -2930,55127
0,00000 0,00000 0,00000 0,00000 70,73513    -1,61707    44,79899    -3508,87949
0,00000 0,00000 0,00000 0,00000 0,00000 -48,17339   117,48437   -3529,44153
0,00000 0,00000 0,00000 0,00000 0,00000 0,00000 -14,87048   178,44579
Решение:
-11,00000
-50,00000
42,00000
-40,00000
-41,00000
44,00000
-12,00000
Невязки:
0,00000
0,00000
0,00000
0,00000
0,00000
0,00000
0,00000
```

Выводы

В ходе выполнения лабораторной работы я познакомилась с одним из численных методов решения систем линейных алгебраических уравнений – методом Гаусса с выбором главного элемента по столбцам. Также столкнулась с проблемами точности представления дробных чисел в компьютере и получающихся из-за этого неточностей.