Санкт-Петербургский национальный исследовательский университет ИТМО

Вычислительная математика

Лабораторная работа №5
АППРОКСИМАЦИЯ ФУНКЦИИ МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ

Автор:

Ненов Владислав Александрович

Вариант 5

Группа №P32082

Преподаватель:

Екатерина Алексеевна Машина

Санкт-Петербург
2023

Цель работы

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов. Лабораторная работа состоит из двух частей: вычислительной и программной. № варианта задания лабораторной работы определяется как номер в списке группы согласно ИСУ.

Вычислительная часть

Задание

Сформировать таблицу табулирования заданной функции на указанном интервале (см. табл. 1)
2. Построить линейное и квадратичное приближения по 11 точкам заданного интервала;
3. Найти среднеквадратические отклонения для каждой аппроксимирующей функции. Ответы дать с тремя знаками после запятой;
4. Выбрать наилучшее приближение;
5. Построить графики заданной функции, а также полученные
линейное и квадратичное приближения;
6. Привести в отчете подробные вычисления.

Выполнение

Вариант 5

X	y
2,10	3,7587
2,15	4,1861
2,20	4,9218
2,25	5,3487
2,30	5,9275
2,35	6,4193
2,40	7,0839

$x_1 = 2.112\, x_2 = 2.205$

$h = const = 0.05$, *поэтому используем формулу Ньютона для равностоящих узлов.*

Таблица конечных разностей

x	y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$	$\Delta^6 y$
2.1	3.7587						
		0.4274					
2.15	4.1861		0.3083				
		0.7357		-0.6171			
2.2	4.9218		-0.3088		1.0778		
		0.4269		0.4607		-1.7774	
2.25	**5.3487**		**0.1519**		**-0.6996**		**2.9757**
		0.5788		-0.2389		1.1983	
2.3	5.9275		-0.087		0.4987		
		0.4918		0.2598			
2.35	6.4193		0.1728				
		0.6646					
2.4	7.0839						

Задание 1

Метод Ньютона для равностоящих узлов

$$p = \frac{x-a}{h} = \frac{2.112-2.25}{0.05}$$

$$y(x) = y_0 + p\Delta y_0 + \frac{p(p-1)}{2!} \cdot \Delta^2 y_0 + \frac{p(p-1)(p-2)}{3!} \cdot \Delta^3 y_0 + \frac{p(p-1)(p-2)(p-3)}{4!} \cdot \Delta^4 y_0$$

$$+ \frac{p(p-1)(p-2)(p-3)(p-4)}{5!} \cdot \Delta^5 y_0 + \frac{p(p-1)(p-2)(p-3)(p-4)(p-5)}{6!} \cdot \Delta^6 y_0$$

$$y(x_1) = y(2.112) = 3.7587 + 0.2137 - 0.0385 - 0.0386 - 0.0421 - 0.0486 - 0.061 = 3.7436$$

Задание 2

Метод Гаусса для (x<a)

$$p = \frac{x - x_0}{h} = \frac{2.205 - 2.25}{0.05} = -0.9$$

$$y_p = y_0 + p\Delta y_{-1} + \frac{(p+1)p}{2!}\cdot\Delta^2 y_{-1} + \frac{(p+1)p(p-1)}{3!}\cdot\Delta^3 y_{-2} + \frac{(p+2)(p+1)p(p-1)}{4!}\cdot\Delta^4 y_{-2}$$

$$+ \frac{(p+2)(p+1)p(p-1)(p-2)}{5!}\cdot\Delta^5 y_{-3} + \frac{(p+3)(p+2)(p+1)p(p-1)(p-2)}{6!}\cdot\Delta^6 y_{-3}$$

$$y(x_2) = y(2.205) = 5.3487 - 0.3842 - 0.0068355 + 0.01312995 - 0.005483115 + 0.008079616 - 0.0047343759 = 4.9686$$

Программная часть

Задание

Исходные данные задаются тремя способами:

- В виде набора данных (таблицы x,y), пользователь вводит значения с клавиатуры;
- В виде сформированных в файле данных (подготовить не менее трех тестовых
- На основе выбранной функции, из тех, которые предлагает программа. Пользователь выбирает уравнение, исследуемый интервал и количество точек на интервале (не менее двух функций).

2. Сформировать и вывести таблицу конечных разностей;
3. Вычислить приближенное значение функции для заданного значения аргумента, введенного с клавиатуры, указанными методами (см. табл. 5.2). Сравнить полученные значения;
4. Построить графики заданной функции с отмеченными узлами интерполяции и интерполяционного многочлена Ньютона/Гаусса (разными цветами);5. Программа должна быть протестирована на различных наборах данных, в том числе и некорректных.
6. Проанализировать результаты работы программы.

Выполнение

На ЯП Kotlin

```
package org.example.math.interpolation

class GaussInterpolation(nodes: List<Node>) : Interpolation {
   private val h = (nodes[1].x-nodes[0].x).absoluteValue
   private val nodes = if (nodes.size % 2 == 0) nodes.subList(0, nodes.size-1) else
nodes

   private val dyList: List<List<Double>> by lazy {
       val result = ArrayList<List<Double>>()
       result.add(nodes.map { it.y })
       while (result.last().size > 1) {
           result.add(
               List(result.last().size-1) { i ->
                   result.last()[i + 1] - result.last()[i]
               }
           )
```

```
        }
        result.removeFirst()
        result
    }

    val dyTable : String get() {
        val sb = StringBuilder()
        dyList.forEachIndexed { i, dy ->
            sb.append("yΔ${i+1} | ")
            dy.forEach { y ->
                sb.append(y.fixedLengthStr(6))
                sb.append(" | ")
            }
            sb.append("\n")
        }
        return sb.toString()
    }

    override fun interpolate(x: Double): Double {
        val a = nodes[nodes.size/2]
        val t = (x-a.x)/h
        var tMult = t
        var result = a.y
        var diffT = if (x < a.x) 1.0 else -1.0
        dyList.forEachIndexed { i, dy ->
            val dyCenter = getDyCenter(dy, a, x)
            result += dyCenter * tMult / (i+1).factorial()
            tMult *= (t + diffT)
            diffT *= -1
            diffT += (i%2)*(diffT/diffT.absoluteValue)
        }
        return result
    }

    private fun getDyCenter(dy: List<Double>, a: Node, x: Double) : Double {
        if (x > a.x)
            return dy[dy.size/2]
        return if (dy.size > 1)
            dy[(dy.size/2) - (1-dy.size%2)]
        else dy[0]
    }

}
```

```
package org.example.math.interpolation

class LagrangeInterpolation(private val nodes: List<Node>) : Interpolation {
    override fun interpolate(x: Double) : Double {
        var result = 0.0
        nodes.forEach { inode ->
            var num = 1.0
            var den = 1.0
            nodes.forEach { jnode ->
                if (inode != jnode) {
                    num *= (x-jnode.x)
                    den *= (inode.x-jnode.x)
                }
            }
            result += inode.y * num / den
```

```
        }
        return result
    }
}

fun main() {
    println("Введите \n1: для ввода точек таблицей с консоли, " +
            "\n2: для ввода таблицей с файла, \n3: для ввода промежутком функции")
    val input = readln()
    val x: List<Double>?
    val y: List<Double>?
    if (input == "1") {
        try {
            val (tmpX, tmpY) = readTable(System.`in`)
            x = tmpX
            y = tmpY
        } catch (e: Exception) {
            println("Что-то пошло не так")
            return
        }
    } else if (input == "2") {
        println("Введите имя файла")
        try {
            val (tmpX, tmpY) = readTable(FileInputStream(readln()))
            x = tmpX
            y = tmpY
        } catch (e: Exception) {
            println("Что-то пошло не так")
            return
        }
    } else if (input == "3") {
        println("Выберите функцию: \n1: x^3 - (x/2)^4 + 5 \n2: 2^(0.3*x) - x/3 + 2")
        val funcNum = readln()
        println("Введите границы исследуемого интервала и шаг через пробел")
        val (start, end, h) = readln().split(" ").map { it.toDoubleOrNull() }
        if (start == null || end == null || h == null || h < 0 || end < start) {
            println("Некорректные исходные данные")
            return
        }
        x = List(((end-start) / h).toInt()) { i -> start+i*h }
        if (funcNum == "1")
            y = List(x.size) { i -> x[i].pow(3) - (x[i]/2).pow(4) + 5 }
        else if (funcNum == "2")
            y = List(x.size) { i -> 2.0.pow(0.3*x[i])-x[i]/3+2 }
        else {
            println("Некорректные исходные данные")
            return
        }
    } else {
        println("некорректный ввод")
        return
    }

    val table = x.mapIndexed { i, it ->  Node(it, y[i]) }
    val lagrangeInterp = LagrangeInterpolation(table)
    val gaussInterp = GaussInterpolation(table)

    println(x)
    println("Таблица конечных разностей")
```

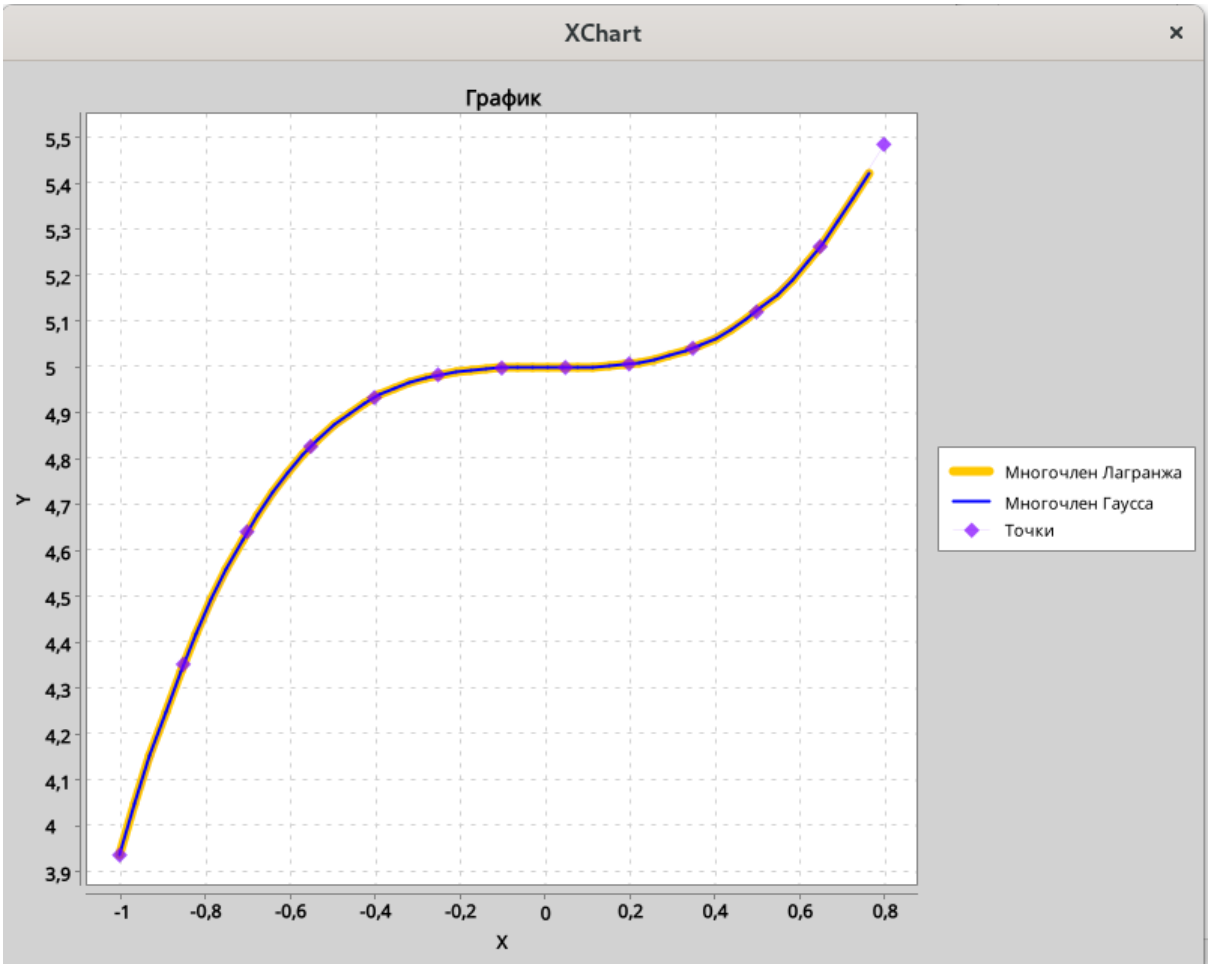
```
    println(gaussInterp.dyTable)

    XYChartBuilder()
        .width(800)
        .height(600)
        .title("График")
        .xAxisTitle("X")
        .yAxisTitle("Y")
        .build()
        .drawFunction(x.first(), x.last(), 50, "Многочлен Лагранжа",
            6f, Color.ORANGE) { lagrangeInterp.interpolate(it) }
        .drawFunction(x.first(), x.last(), 50, "Многочлен Гаусса",
            2f, Color.BLUE
        ) { gaussInterp.interpolate(it) }
        .drawSeries(Series(x, y), "Точки")
        .show()
}

fun readTable(stream: InputStream): Pair<List<Double>, List<Double>> {
    print("Введите координаты x: ")
    val scanner = Scanner(stream)
    val xStr = scanner.nextLine().split(" ", ", ")
    val x = xStr.mapNotNull { it.toDoubleOrNull() }.toList()
    print("Введите координаты y: ")
    val yStr = scanner.nextLine().split(" ", ", ")
    val y = yStr.mapNotNull { it.toDoubleOrNull() }.toList()
    println()
    if (x.size != xStr.size || y.size != yStr.size || y.size != x.size) {
        println("Некорректные данные!")
        throw IllegalArgumentException()
    }
    return Pair(x, y)
}
```

Пример выполнения программы

```
/home/vlad/.jdks/corretto-11.0.16/bin/java ...
Введите
1: для ввода точек таблицей с консоли,
2: для ввода таблицей с файла,
3: для ввода промежутком функции
3
Выберите функцию:
1: x^3 - (x/2)^4 + 5
2: 2^(0.3*x) - x/3 + 2
1
Введите границы исследуемого интервала и шаг через пробел
-1 1 0.15
Таблица конечных разностей
yΔ1 | 0,41575 | 0,28874 | 0,18591 | 0,10649 | 0,04973 | 0,01486 | 0,00113 | 0,00778 | 0,03404 | 0,07916 | 0,14237 | 0,22293 |
yΔ2 | -0,1270 | -0,1028 | -0,0794 | -0,0568 | -0,0349 | -0,0137 | 0,00664 | 0,02626 | 0,04512 | 0,06322 | 0,08056 |
yΔ3 | 0,02417 | 0,02341 | 0,02265 | 0,02190 | 0,02114 | 0,02038 | 0,01962 | 0,01886 | 0,01810 | 0,01734 |
yΔ4 | -0,0008 | -0,0008 | -0,0008 | -0,0008 | -0,0008 | -0,0008 | -0,0008 | -0,0008 | -0,0008 |
yΔ5 | -0,0000 | 0,00000 | -0,0000 | 0,00000 | -0,0000 | 0,00000 | -0,0000 | 0,00000 |
yΔ6 | 0,00000 | -0,0000 | 0,00000 | -0,0000 | 0,00000 | -0,0000 | 0,00000 |
yΔ7 | -0,0000 | 0,00000 | -0,0000 | 0,00000 | -0,0000 | 0,00000 |
yΔ8 | 0,00000 | -0,0000 | 0,00000 | -0,0000 | 0,00000 |
yΔ9 | -0,0000 | 0,00000 | -0,0000 | 0,00000 |
yΔ10 | 0,00000 | -0,0000 | 0,00000 |
yΔ11 | -0,0000 | 0,00000 |
yΔ12 | 0,00000 |
```



Вывод

В ходе лабораторной работы была решена задача интерполяции - найдены значения функции при заданных значениях аргумента, отличных от узловых точек, также для решения этой задачи была написана программа.