Национальный исследовательский университет информационных технологий, механики и оптики

Факультет Программной Инженерии и Компьютерной Техники

Вариант №12

Лабораторная работа №5

«Интерполяция функции»

По дисциплине:

«Вычислительная математика»

Работу выполнила:

Студентка группы P3212

Никонова Наталья Игоревна

Преподаватель:

Малышева Татьяна Алексеевна

Санкт-Петербург

2022

Цель

Решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек с использованием многочлена Лагранжа, Ньютона и Гаусса

Рабочие формулы

Многочлен Лагранжа

$$L_n(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x\_j}{x_i - x_j}$$

Первая интерполяционная формула Ньютона для интерполирования вперед:

$$N_n(x) = y_0 + t * \Delta y_0 + \frac{t * (t-1)}{2!}\Delta^2 y_0 + \cdots + \frac{t * (t-1) * \ldots * (t-n+1)}{n!}\Delta^n y_0,$$
$$t = \frac{x - x_0}{h}$$

Вторая интерполяционная формула Ньютона для интерполирования назад:

$$N_n(x) = y_n + t * \Delta y_{n-1} + \frac{t * (+-1)}{2!}\Delta^2 y_{n-2} + \cdots + \frac{t * (t+1) * \ldots * (t+n-1)}{n!}\Delta^n y_0$$

Первая интерполяционная формула Гаусса:

$$P_n(x) = y_0 + t\Delta y_0 + \frac{t * (t-1)}{2!}\Delta^2 y_{-1} + \frac{(t+1) * t * (t-1)}{3!}\Delta^3 y_{-1}$$
$$+ \frac{(t+1) * t * (t-1) * (t-2)}{4!}\Delta^4 y_{-2}$$
$$+ \frac{(t+2) * (t+1) * t * (t-1) * (t-2)}{5!}\Delta^5 y_{-2} + \cdots$$
$$+ \frac{(t+n-1) * \ldots * (t-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-(n-1)}$$
$$+ \frac{(t+n-1) * \ldots * (t-n)}{(2n)!}\Delta^{2n} y_{-n}$$

Вторая интерполяционная формула Гаусса:

$$P_n(x) = y_0 + t\Delta y_{-1} + \frac{t * (t+1)}{2!}\Delta^2 y_{-1} + \frac{(t+1) * t * (t-1)}{3!}\Delta^3 y_{-2}$$
$$+ \frac{(t+2) * (t+1) * t * (t-1)}{4!}\Delta^4 y_{-2} + \cdots$$
$$+ \frac{(t+n-1) * \ldots * (t-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-n}$$
$$+ \frac{(t+n) * (t+n-1) * \ldots * (t-n+1)}{(2n)!}\Delta^{2n} y_{-n}$$

Вычислительная реализация задачи

x	y
1,05	0,1213

1,15	1,1316
1,25	2,1459
1,35	3,1565
1,45	4,1571
1,55	5,1819
1,65	6,1969

Таблица конечных разностей

x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
1.05	0.1213	1.0103	0.0040	-0.0077	0.0014	0.0391	-0.1478
1.15	1.1316	1.0143	-0.0037	-0.0063	0.0405	-0.1087	
1.25	2.1459	1.0106	-0.0100	0.0342	-0.0682		
1.35	3.1565	1.0006	0.0242	-0.0340			
1.45	4.1571	1.0248	-0.0098				
1.55	5.1819	1.0150					
1.65	6.1969						

Используемые конечные разности в формуле Ньютона выделены желтым, в формуле Гаусса – зеленым

$$x_1 = 1.112$$

Так как x находится в начале интервала, используем формулу Ньютона для интерполирования вперед

$$h = 0.1 \;\; t = \frac{x - x_o}{h} = \frac{1.112 - 1.05}{0.1} = \frac{0.062}{0.1} = 0.62$$

$$t * \Delta y_0 = 0.62 * 1.0103 = 0.6264;$$

$$\frac{t * (t-1)}{2!}\Delta^2 y_0 = \frac{0.62 * (0.62 - 1)}{2} * 0.004 = -0.1178 * 0.004 = 0.0005;$$

$$\frac{t * (t-1) * (t-2)}{3!}\Delta^3 y_0 = \frac{0.62 * (0.62 - 1) * (0.62 - 2)}{3 * 2} * -0.0077 = 0.0542 * -0.0077$$
$$= -0.0004;$$

$$\frac{t * (t-1) * (t-2) * (t-3)}{4!}\Delta^4 y_0 = \frac{0.62(0.62 - 1)(0.62 - 2)(0.62 - 3)}{4 * 3 * 2} * 0.0014$$
$$= -0.0322 * 0.0014 = -0.00005;$$

$$\frac{t * (t-1) * (t-2) * (t-3) * (t-4)}{5!}\Delta^5 y_0$$
$$= \frac{0.62(0.62 - 1)(0.62 - 2)(0.62 - 3)(0.62 - 4)}{5 * 4 * 3 * 2} * 0.0391 = 0.0218 * 0.0391$$
$$= 0.00085$$

$$\frac{t * (t-1) * (t-2) * (t-3) * (t-4) * (t-5)}{6!}\Delta^6 y_0$$
$$= \frac{0.62(0.62 - 1)(0.62 - 2)(0.62 - 3)(0.62 - 4)(0.62 - 5)}{6 * 5 * 4 * 3 * 2} * -0.1478$$
$$= -0.0159 * -0.1478 = 0.0024$$

$$y(1.112) = 0.1213 + 0.6264 + 0.0005 - 0.0004 - 0.00005 + 0.00085 + 0.0024 = 0.74996$$

$$x_2 = 1.319$$

Так как x находится в середине интервала, но меньше середины, то используем вторую формулу Гаусса

$$a = 1.35 \quad t = \frac{1.319 - 1.35}{0.1} = -0.31$$

$$t * \Delta y_{-1} = -0.31 * 1.0106 = -0.31329$$

$$\frac{t * (t+1)}{2!} * \Delta^2 y_{-1} = \frac{-0.31 * (-0.31 + 1)}{2} * -0.0100 = 0.00107$$

$$\frac{t * (t+1) * (t-1)}{3!} * \Delta^3 y_{-2} = \frac{-0.31 * (-0.31 + 1) * (-0.31 - 1)}{3 * 2} * -0.0063 = -0.00029$$

$$\frac{t * (t+1) * (t-1) * (t+2)}{4!} * \Delta^4 y_{-2} = \frac{-0.31 * (-0.31 + 1) * (-0.31 - 1) * (-0.31 + 2)}{4 * 3 * 2} * 0.0405 = 0.00080$$

$$\frac{t * (t+1) * (t-1) * (t+2) * (t-2)}{5!} * \Delta^5 y_{-3} = \frac{-0.31 * (-0.31 + 1) * (-0.31 - 1) * (-0.31 + 2) * (-0.31 - 2)}{5 * 4 * 3 * 2} * 0.0391 = -0.00036$$

$$\frac{t * (t+1) * (t-1) * (t+2) * (t-2) * (t+3)}{6!} * \Delta^6 y_{-3} = \frac{-0.31 * (-0.31 + 1) * (-0.31 - 1) * (-0.31 + 2) * (-0.31 - 2) * (-0.31 + 3)}{6 * 5 * 4 * 3 * 2} = -0.00409$$

$$y(1.319) = 3.1565 - 0.31329 + 0.00107 - 0.00029 + 0.00080 - 0.00036 - 0.00409 = 2.84034$$

Программная реализация задачи

Исходный кодъ https://github.com/nanikon/computational_math/tree/lab5/interpolation

Метод Лагранжа

```
class LagrangeMethod : Method {
    nanikon
    override fun compute(x: BigDecimal, table: List<Pair<BigDecimal, BigDecimal>>): Pair<BigDecimal, BigDecimal> {
        println("\nМетод Лагранжа")
        val xs = table.map { it.first }
        val differences = mutableListOf<List<BigDecimal>>()
        println("Таблица попарных разностей")
        for (i in xs.indices) {
            val row = mutableListOf<BigDecimal>()
            for (j in 0 ≤ until < i) {
                row.add(xs[j] - xs[i])
            }
            differences.add(row)
            println(row.joinToString(separator = "\t"))
        }
        val multiply = xs.multiplyOf { number -> x - number }
        println("Общая часть числителя $multiply")

        val denominators = (xs.indices).map { i ->
            differences[i].multiplyOf { number -> -number } *
                    (i + 1 ≤ until < xs.size).map { j -> differences[j][i] }.multiplyOf { it } *
                    (x - xs[i])
        }
        println("Знаменатели: ${denominators.joinToString(separator=" ")}")
        val coef = (xs.indices).map { i ->
            multiply.divide(denominators[i], MathContext.DECIMAL32)
        }
```

```
        val y = (xs.indices).map { i -> coef[i] * table[i].second }.sumOf { it }
        coef.mapIndexed { index, number -> "$number * y_$index" } List<String>
            .joinToString(separator = " + ") String
            .also { println("Значение многочлена Лагранжа при x=$x: $it = $y") }
        return Pair(x, y)
    }
}
```

Метод Ньютона

```
class NewtonMethod : Method {
    nanikon *
    override fun compute(x: BigDecimal, table: List<Pair<BigDecimal, BigDecimal>>): Pair<BigDecimal, BigDecimal> {
        println("\nМетод Ньютона")
        val xs = table.map { it.first }
        val ys = table.map { it.second }
        val finiteDifferences = mutableListOf(ys)
        println("Таблица конечных разностей")
        println(ys.joinToString(separator = "\t"))
        for (i in 0 ≤ until < ys.size - 1) {
            val row = mutableListOf<BigDecimal>()
            for (j in 0 ≤ until < ys.size - 1 - i) {
                row.add(finiteDifferences[i][j + 1] - finiteDifferences[i][j])
            }
            finiteDifferences.add(row)
            println(row.joinToString(separator = "\t"))
        }
        val middle = (xs[0] + xs.last()).divide(BigDecimal( val: 2), MathContext.DECIMAL32)
        val h = xs[1] - xs[0]
        var y = BigDecimal.ZERO
        var stringEq = ""
        var stringDiff = ""
```

```kotlin
        if (x < middle) {
            // формула вперед
            println("Используется формула вперед")
            val a = xs.maxOf { if (it < x) it else xs[0] }
            val t = (x - a).divide(h, MathContext.DECIMAL32)
            val i = xs.indexOf(a)
            var factor = BigDecimal.ONE
            for (j in finiteDifferences[i].indices) {
                if (j != 0) { stringEq += " + " }
                stringEq += "${finiteDifferences[j][i]} * $factor"
                y += finiteDifferences[j][i] * factor
                stringDiff += "${finiteDifferences[j][i]} "
                factor *= (t - j.toBigDecimal()).divide((j + 1).toBigDecimal(), MathContext.DECIMAL32)
            }
```

```kotlin
        } else {
            // формула назад
            println("Используется формула назад")
            val a = xs.minOf { if (it > x) it else xs.last() }
            val t = (x - a).divide(h, MathContext.DECIMAL32)
            val i = xs.indexOf(a)
            var factor = BigDecimal.ONE
            for (j in i ≥ downTo ≥ 0) {
                if (j != i) { stringEq += " + " }
                val c = i - j
                stringEq += "${finiteDifferences[c][j]} * $factor"
                y += finiteDifferences[c][j] * factor
                stringDiff += "${finiteDifferences[c][j]} "
                factor *= (t + c.toBigDecimal()).divide((c + 1).toBigDecimal(), MathContext.DECIMAL32)
            }
        }
        println("Используемые конечные разности $stringDiff")
        println("\nЗначение многочлена Ньютона при x=$x: $stringEq = $y")
        return Pair(x, y)
    }
}
```

Результат выполнения программы

```
Для задания функции через таблицу введите 1, для выбора её формулы - 2:
2
Выберите функцию для интерполирования и введите её номер:
1 - sin(x)
2 - sqrt(x)
3 - ln(x)
2
Введите x для подсчета приблизительного значения функции в нем
13
Введите количество узлов интерполяции
10
Получена таблица интерполяционных узлов:
(4.0, 2) (6.0, 2.44949) (8.0, 2.828427) (10.0, 3.162278) (12.0, 3.464102) (14.0, 3.741657) (16.0, 4) (18.0, 4.242641) (20.0, 4.472136) (22.0, 4.690416)
Значение аргумента, при котором будет рассчитываться функция: 13

Метод Лагранжа
Таблица попарных разностей

-2.0
-4.0    -2.0
-6.0    -4.0    -2.0
-8.0    -6.0    -4.0    -2.0
-10.0   -8.0    -6.0    -4.0    -2.0
-12.0   -10.0   -8.0    -6.0    -4.0    -2.0
-14.0   -12.0   -10.0   -8.0    -6.0    -4.0    -2.0
-16.0   -14.0   -12.0   -10.0   -8.0    -6.0    -4.0    -2.0
-18.0   -16.0   -14.0   -12.0   -10.0   -8.0    -6.0    -4.0    -2.0
```

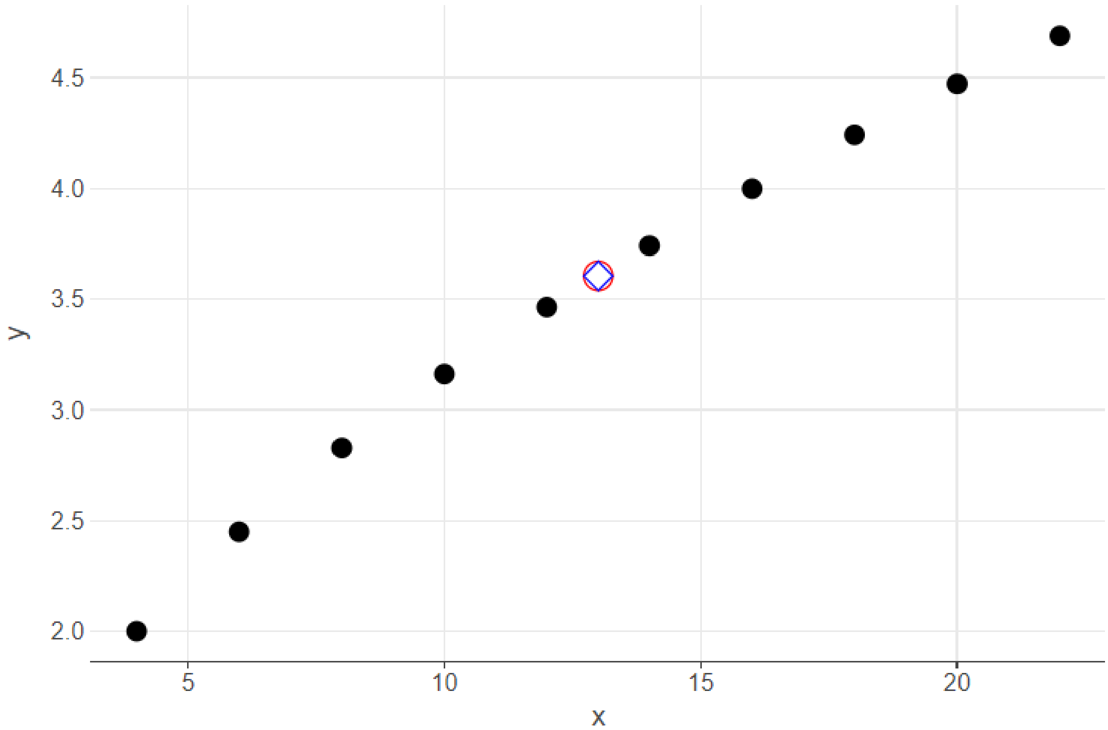
```
Общая часть числителя -893025.0000000000
Знаменатели: -1672151040.0000000000 144506880.0000000000 -25804800.0000000000 6635520.0000000000 -1474560.0000000000 -1474560.0000000000 6635520.0000000000 -25804800.0000000000
Значение многочлена Лагранжа при x=13: 0.0005340576 * y_0 + -0.006179810 * y_1 + 0.03460693 * y_2 + -0.1345825 * y_3 + 0.6056213 * y_4 + 0.6056213 * y_5 + -0.1345825 * y_6 + 0.0

Метод Ньютона
Таблица конечных разностей
2   2.44949 2.828427    3.162278    3.464102    3.741657    4   4.242641    4.472136    4.690416
0.44949 0.378937    0.333851    0.301824    0.277555    0.258343    0.242641    0.229495    0.218280
-0.070553   -0.045086   -0.032027   -0.024269   -0.019212   -0.015702   -0.013146   -0.011215
0.025467    0.013059    0.007758    0.005057    0.003510    0.002556    0.001931
-0.012408   -0.005301   -0.002701   -0.001547   -0.000954   -0.000625
0.007107    0.002600    0.001154    0.000593    0.000329
-0.004507   -0.001446   -0.000561   -0.000264
0.003061    0.000885    0.000297
-0.002176   -0.000588
0.001588
Используется формула назад
Используемые конечные разности 3.741657 0.277555 -0.024269 0.007758 -0.005301 0.007107

Значение многочлена Ньютона при x=13: 3.741657 * 1 + 0.277555 * -0.5 + -0.024269 * -0.125 + 0.007758 * -0.0625 + -0.005301 * -0.0390625 + 0.007107 * -0.02734375 = 3.60544098828
```

```
04800.0000000000 144506880.0000000000 -1672151040.0000000000
45825 * y_6 + 0.03460693 * y_7 + -0.006179810 * y_8 + 0.0005340576 * y_9 = 3.6055505433738416

= 3.60544098828125
```



Выводы

В ходе выполнения лабораторной работы я познакомилась с методами интерполяции функции и попробовала определять какой из них наиболее подходящих для конкретных входных данных.