Санкт-Петербургский национальный исследовательский университет ИТМО

Вычислительная математика

Лабораторная работа №4
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

$$y = \frac{6x}{x^4 + 5} \quad \Big| \quad x \in [0, 2] \quad h = 0{,}2$$

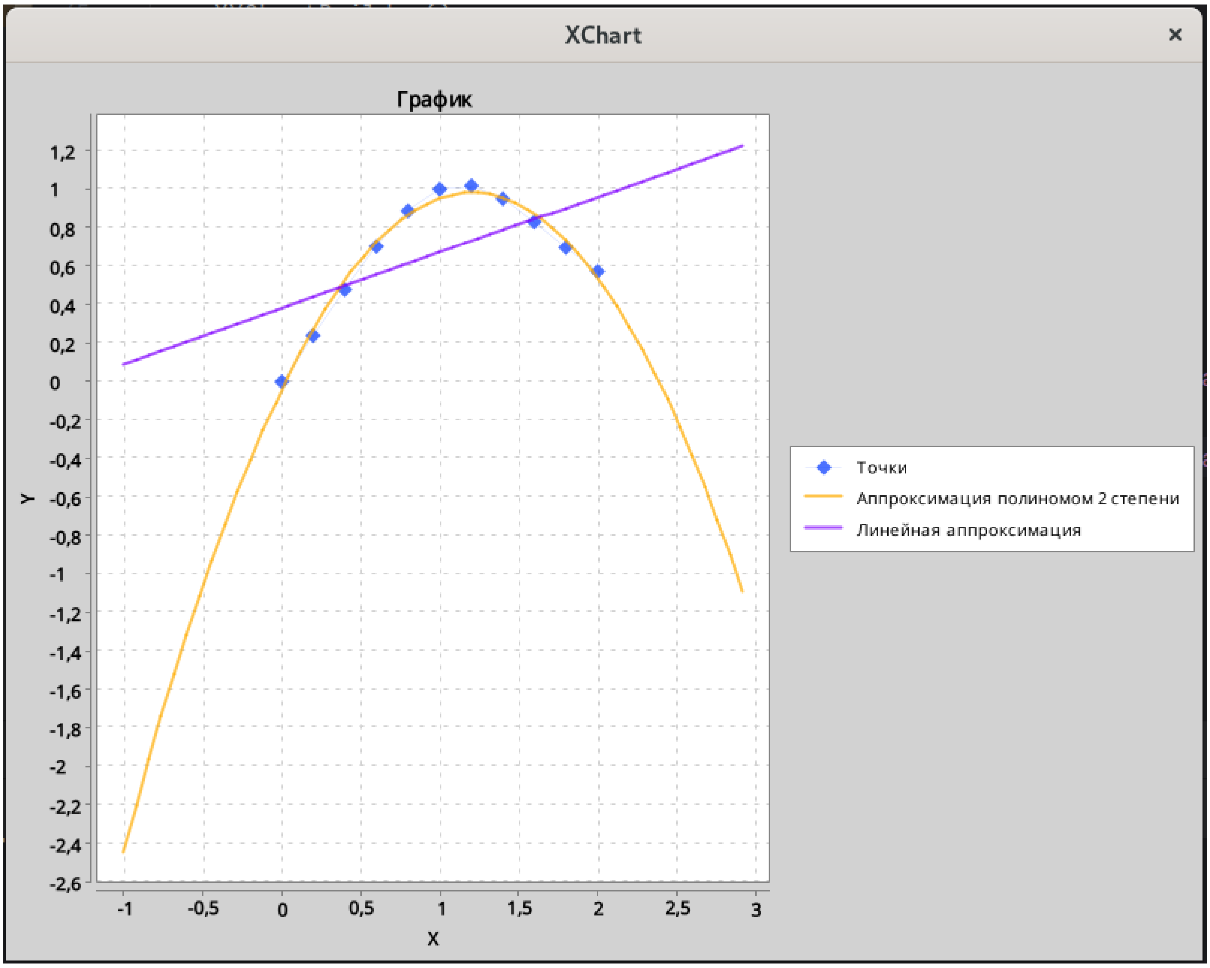
x_i	0	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
y_i	0	0.240	0,478	0,702	0,887	1	1,018	0,950	0,831	0,697	0,571

Аппроксимация полиномом 2 степени

Среднеквадратичное отклонение: **0,035**
φ(X) = { -0,042, 0,270, 0,525, 0,724, 0,866, 0,952, 0,982, 0,955, 0,872, 0,733, 0,537 }
Отклонения: { 0,002, 0,001, 0,002, 0,000, 0,000, 0,002, 0,001, 0,000, 0,002, 0,001, 0,001 }
Коэффициенты: { -0,042; 1,698; -0,704 }

Линейная аппроксимация

Среднеквадратичное отклонение: **0,251**
Коэффициент Пирсона: 0,589
φ(X) = { 0,381, 0,439, 0,497, 0,554, 0,612, 0,670, 0,728, 0,786, 0,844, 0,902, 0,960 }
Отклонения: { 0,145, 0,039, 0,000, 0,022, 0,076, 0,109, 0,084, 0,027, 0,000, 0,042, 0,151 }
Коэффициенты: { 0,290; 0,381 }



Программная реализация

На языке Kotlin

```
abstract class Approximation(x: Array<Double>, y: Array<Double>) {
   abstract val name: String
   abstract val ratios: Array<Double>

   abstract fun approximate(x: Double): Double

   val deviations by lazy {
        x.mapIndexed() { i, it ->
             (approximate(it) - y[i]).pow(2)
        }
   }

   val standardDeviation: Double by lazy {
       sqrt(deviations.sum() / x.size)
   }
}

class LinearApproximation( x: Array<Double>, y: Array<Double>) :
Approximation(x, y) {
   override val name: String = "Линейная аппроксимация"

   override val ratios: Array<Double> by lazy {
        val xSqSum = x.reduce { acc, it ->
             acc + it*it
        }
        val xy = Array(x.size) { i ->
             x[i]*y[i]
        }
        val matrix = arrayOf(
             doubleArrayOf(xSqSum, x.sum()),
             doubleArrayOf(x.sum(), x.size.toDouble())
        )
        NewtonSolveMethod(matrix, arrayOf(xy.sum(), y.sum())).solve()
   }

   val correlationRatio: Double by lazy {
        val xMean = x.sum() / x.size
        val yMean = y.sum() / y.size
        var a = 0.0
        var b = 0.0
        var c = 0.0
        for (i in x.indices) {
             a += (x[i] - xMean)*(y[i] - yMean)
             b += (x[i] - xMean).pow(2)
             c += (y[i] - yMean).pow(2)
        }
        a / sqrt(b*c)
   }
```

```
   override fun approximate(x: Double) : Double = ratios[0]*x + ratios[1]

}

class Polynomial2Approximation(x: Array<Double>, y: Array<Double>) :
Approximation(x, y) {
   override val name: String = "Аппроксимация полиномом 2 степени"

   override val ratios: Array<Double> by lazy {
       val xSqSum = x.reduce { acc, it ->
           acc + it*it
       }
       val xPow3Sum = x.reduce { acc, it ->
           acc + it*it*it
       }
       val xPow4Sum = x.reduce { acc, it ->
           acc + it*it*it*it
       }
       val xySum = Array(x.size) { i ->
           x[i]*y[i]
       }.sum()
       val xSqYSum = Array(x.size) { i ->
           x[i]*x[i]*y[i]
       }.sum()
       val xSum = x.sum()
       val matrix = arrayOf(
           doubleArrayOf(x.size.toDouble(), xSum, xSqSum),
           doubleArrayOf(xSum, xSqSum, xPow3Sum),
           doubleArrayOf(xSqSum, xPow3Sum, xPow4Sum)
       )

       NewtonSolveMethod(matrix, arrayOf(y.sum(), xySum, xSqYSum)).solve()
   }

   override fun approximate(x: Double): Double {
       return ratios[0] + ratios[1]*x + ratios[2]*x.pow(2)
   }
}
```

```
class Polynomial3Approximation(x: Array<Double>, y: Array<Double>) :
Approximation(x, y) {
```

```kotlin
    override val name: String = "Аппроксимация полиномом 3 степени"

    override val ratios: Array<Double> by lazy {
        var xSum = 0.0
        var xSqSum = 0.0
        var xPow3Sum = 0.0
        var xPow4Sum = 0.0
        var xPow5Sum = 0.0
        var xPow6Sum = 0.0
        var xySum = 0.0
        var xSqYSum = 0.0
        var xPow3YSum = 0.0
        x.forEachIndexed { index, x ->
            xSum += x
            xSqSum += x.pow(2)
            xPow3Sum += x.pow(3)
            xPow4Sum += x.pow(4)
            xPow5Sum += x.pow(5)
            xPow6Sum += x.pow(6)
            xySum += x*y[index]
            xSqYSum += x.pow(2)*y[index]
            xPow3YSum += x.pow(3)*y[index]
        }
        val lMatrix = arrayOf(
            doubleArrayOf(x.size.toDouble(), xSum, xSqSum, xPow3Sum),
            doubleArrayOf(xSum, xSqSum, xPow3Sum, xPow4Sum),
            doubleArrayOf(xSqSum, xPow3Sum, xPow4Sum, xPow5Sum),
            doubleArrayOf(xPow3Sum, xPow4Sum, xPow5Sum, xPow6Sum),
        )
        val rMatrix = arrayOf(y.sum(), xySum, xSqYSum, xPow3YSum)
        NewtonSolveMethod(lMatrix, rMatrix).solve()
    }

    override fun approximate(x: Double): Double {
        return ratios[0] + ratios[1]*x + ratios[2]*x.pow(2) +
ratios[3]*x.pow(3)
    }
}
```

```kotlin
class PowerApproximation(x: Array<Double>, y: Array<Double>) :
Approximation(x, y) {
   override val name: String = "Степенная аппроксимация"

   override val ratios: Array<Double> by lazy {
       val approx = LinearApproximation(
           x.map { ln(it) }.toTypedArray(),
           y.map { ln(it) }.toTypedArray()
       )
       arrayOf(exp(approx.ratios[0]), approx.ratios[1])
   }

   override fun approximate(x: Double): Double = ratios[0]*x.pow(ratios[1])
}


class ExpApproximation(x: Array<Double>, y: Array<Double>) : Approximation(x,
y) {
   override val name: String = "Экспоненциальная аппроксимация"

   override val ratios: Array<Double> by lazy {
       val approx = LinearApproximation(x, y.map { ln(it) }.toTypedArray())
       arrayOf(exp(approx.ratios[0]), approx.ratios[1])
   }

   override fun approximate(x: Double): Double = ratios[0]* exp(ratios[1]*x)
}

class NewtonSolveMethod(private val a: Array<DoubleArray>, private val b:
Array<Double>) {
   fun solve() : Array<Double> {
       if (a.isEmpty() || a.size != a[0].size || a.size != b.size)
           throw IllegalArgumentException("incorrect matrix size")
       for (i in 1 until a.size) {
           swapToMaxRow(i)
           for (j in i until a.size) {
               val k = -(a[j][i-1]/a[i-1][i-1])
               a[i-1] = a[i-1].map { it ->
                   it * k
               }.toDoubleArray()
               b[i-1] *= k
               a[j] = a[j].mapIndexed { ii, it ->
                   it + a[i-1][ii]
               }.toDoubleArray()
               b[j] += b[i-1]
           }
       }
       val result = Array(b.size) {0.0}
       for (i in result.indices) {
           val ii = b.size-1-i
           var s = 0.0
           result.slice(0 until i).forEachIndexed { j, it ->
               s += it * a[b.size-1-i][b.size-1-j]
```

```
            }
            result[i] = (b[ii]-s)/a[ii][ii]
        }
        result.reverse()
        return result
    }

    private fun swapToMaxRow(i: Int) {
        val maxInd = a.slice(i - 1 until a.size).foldIndexed(i - 1) { ii,
acc, it ->
            val toCompare = a[acc][i - 1]
            if (it[i - 1].absoluteValue > toCompare.absoluteValue)
                ii
            else
                acc
        }
        a.swap(maxInd, i - 1)
        b.swap(maxInd, i - 1)
    }
}
```

Примеры работы

1. $f(x) = 0.05\, ln(x)$

x = { 0,500, 1,000, 1,500, 2,000, 2,500, 3,000, 3,500, 4,000, 4,500, 5,000, 5,500 }
y = { 1,165, 1,200, 1,220, 1,235, 1,246, 1,255, 1,263, 1,269, 1,275, 1,280, 1,285 }

Аппроксимация полиномом 3 степени
Среднеквадратичное отклонение: 0,002
fi(X): { 1,168, 1,196, 1,219, 1,235, 1,248, 1,257, 1,263, 1,269, 1,274, 1,279, 1,287 }
Отклонения: { 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000 }
Коэффициенты: { 1,132, 0,080, -0,017, 0,001 }

Логарифмическая аппроксимация
Среднеквадратичное отклонение: 0,010
fi(X): { 1,144, 1,188, 1,214, 1,232, 1,246, 1,258, 1,267, 1,276, 1,283, 1,290, 1,296 }
Отклонения: { 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000 }
Коэффициенты: { 0,063, 1,188 }

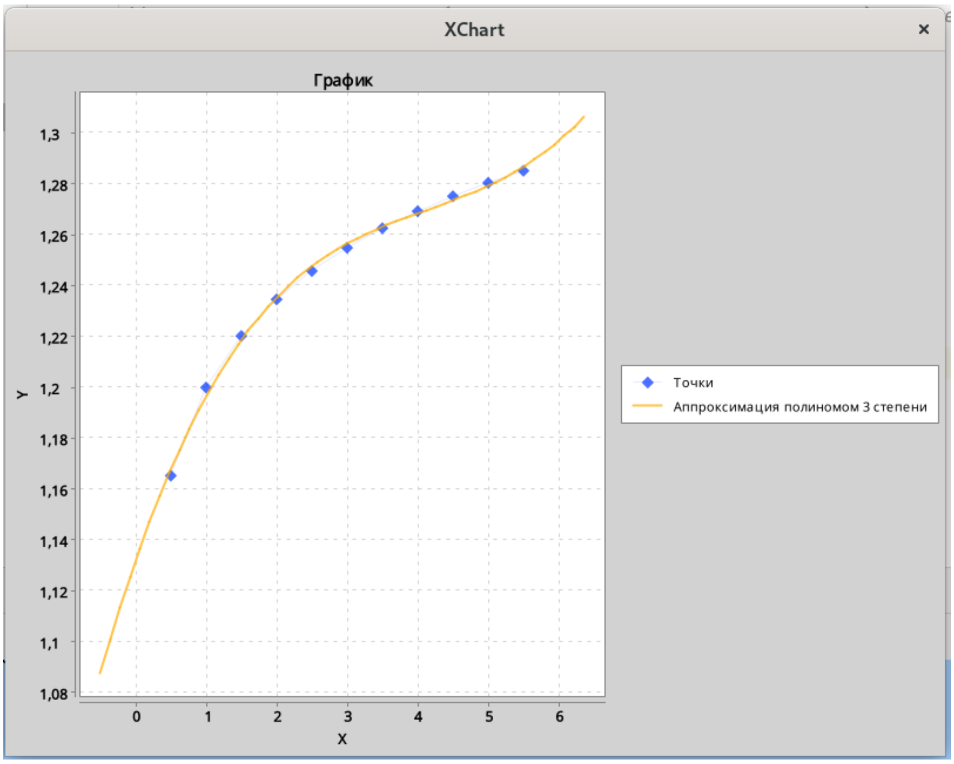
Аппроксимация полиномом 2 степени
Среднеквадратичное отклонение: 0,010
fi(X): { 1,162, 1,191, 1,216, 1,237, 1,254, 1,266, 1,274, 1,278, 1,278, 1,273, 1,265 }
Отклонения: { 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000 }
Коэффициенты: { 1,128, 0,071, -0,008 }

Линейная аппроксимация
Среднеквадратичное отклонение: 0,011

Коэффициент Пирса: 0,949
fi(X): { 1,192, 1,203, 1,213, 1,224, 1,234, 1,245, 1,255, 1,266, 1,277, 1,287, 1,298 }
Отклонения: { 0,001, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000, 0,000 }
Коэффициенты: { 0,021, 1,182 }

Степенная аппроксимация
Среднеквадратичное отклонение: 0,108
fi(X): { 0,934, 1,053, 1,129, 1,187, 1,233, 1,272, 1,307, 1,337, 1,364, 1,389, 1,412 }
Отклонения: { 0,053, 0,022, 0,008, 0,002, 0,000, 0,000, 0,002, 0,005, 0,008, 0,012, 0,016 }
Коэффициенты: { 1,053, 0,172 }

Экспоненциальная аппроксимация
Среднеквадратичное отклонение: 0,652
fi(X): { 1,106, 1,202, 1,307, 1,421, 1,545, 1,680, 1,827, 1,986, 2,160, 2,348, 2,553 }
Отклонения: { 0,004, 0,000, 0,008, 0,035, 0,090, 0,181, 0,318, 0,514, 0,782, 1,139, 1,607 }
Коэффициенты: { 1,017, 0,167 }



2. $f(x) = 0.42x^4 - e^{0.1x}$

{ 1,000, 1,500, 2,000, 2,500, 3,000, 3,500, 4,000, 4,500, 5,000, 5,500, 6,000 }
{ -0,685, 0,964, 5,499, 15,122, 32,670, 61,607, 106,028, 170,658, 260,851, 382,593, 542,498 }

Аппроксимация полиномом 3 степени
Среднеквадратичное отклонение: 1,606
fi(X): { -2,575, 2,854, 7,389, 15,437, 31,410, 59,717, 104,768, 170,973, 262,741, 384,483, 540,608 }

Отклонения: { 3,572, 3,572, 3,572, 0,099, 1,588, 3,572, 1,588, 0,099, 3,572, 3,572, 3,572 }
Коэффициенты: { -33,760, 53,554, -28,250, 5,880 }

Аппроксимация полиномом 2 степени
Среднеквадратичное отклонение: 17,491
fi(X): { 23,884, -2,437, -12,015, -4,848, 19,063, 59,717, 117,116, 191,258, 282,144, 389,775, 514,149 }
Отклонения: { 603,634, 11,572, 306,714, 398,808, 185,164, 3,572, 122,933, 424,367, 453,400, 51,578, 803,660 }
Коэффициенты: { 126,758, -136,362, 33,488 }

Линейная аппроксимация
Среднеквадратичное отклонение: 75,980
Коэффициент Пирса: 0,898
fi(X): { -101,696, -52,669, -3,643, 45,384, 94,410, 143,437, 192,463, 241,490, 290,516, 339,543, 388,569 }
Отклонения: { 10203,130, 2876,565, 83,563, 915,765, 3811,855, 6696,099, 7471,048, 5017,169, 880,022, 1853,306, 23693,955 }
Коэффициенты: { 98,053, -199,749 }

Логарифмическая аппроксимация
Среднеквадратичное отклонение: 108,155
fi(X): { -134,490, -34,221, 36,921, 92,103, 137,190, 175,311, 208,332, 237,459, 263,514, 287,084, 308,601 }
Отклонения: { 17903,679, 1237,998, 987,381, 5926,076, 10924,449, 12928,499, 10466,119, 4462,415, 7,091, 9121,994, 54707,637 }
Коэффициенты: { 247,294, -134,490 }