Университет ИТМО
Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

«Вычислительная математика»

Выполнил:
Студент группы P32102
Гулямов Т.И.

Преподаватель:
Рыбаков С.Д.

Санкт-Петербург
2023

Цель лабораторной работы

Найти функцию, являющуюся наилучшим приближением заданной табличной функции по методу наименьших квадратов.

Порядок выполнения работы

1. Вычислительная реализация задачи
2. Программная реализация задачи

Вычислительная реализация задачи

1. Исследуемые данные
   $y = \frac{15x}{x^4+2}$; $x \in [0, 4]$; $h = 0.4$
2. Таблица табулирования заданной функции на указанном интервале

i	x_i	y_i
0	0	0
1	0.4	2.96209
2	0.8	4.98008
3	1.2	4.4187
4	1.6	2.80584
5	2.0	1.66667
6	2.4	1.02338
7	2.8	0.661776
8	3.2	0.449196
9	3.6	0.317719
10	4.0	0.232558

3. Линейная аппроксимация

$$SX = \sum_{i=0}^{n} x_i = 22$$

$$SXX = \sum_{i=0}^{n} x_i^2 = 61.6$$

$$SY = \sum_{i=0}^{n} y_i = 19.518$$

$SXY = \sum_{i=0}^{n} x_i y_i = 26.1145$

$\{aSXX + bSX = SXY;\ aSX + bn = SY\}$ – система

$\Delta = 193.6$
$\Delta_1 = -142.1365$
$\Delta_2 = 627.7898$

$a = -0.7342$
$b = 3.2427$

$\varphi_{лин}(x) = -0.7342x + 3.2427$

$\delta_{лин} = \sqrt{\frac{\sum_{i=0}^{n}(\varphi_{лин}(x_i) - y_i)^2}{n}} = \sqrt{\frac{(3.2427)^2 + ...}{11}} = \sqrt{\frac{21.474}{11}} = \sqrt{1.95218} = 1.3972$

4. Квадратичная аппроксимация

$SX = \sum_{i=0}^{n} x_i = 22$

$SXX = \sum_{i=0}^{n} x_i^2 = 61.6$

$SXXX = \sum_{i=0}^{n} x_i^3 = 193.6$

$SXXXX = \sum_{i=0}^{n} x_i^4 = 648.526$

$SY = \sum_{i=0}^{n} y_i = 19.518$

$SXY = \sum_{i=0}^{n} x_i y_i = 26.1145$

$SXXY = \sum_{i=0}^{n} x_i^2 y_i = 47.395$

$\{aSXXXX + bSXXX + cSXX = SXXY;\ aSXXX + bSXX + cSX = SXY;\ aSXX + bSX + cn = SY\}$ – система

$\Delta = 4252.6176$
$\Delta_1 = -1978.5533$
$\Delta_2 = 4792.0428$
$\Delta_3 = 9041.5029$

$a = -0.4653$

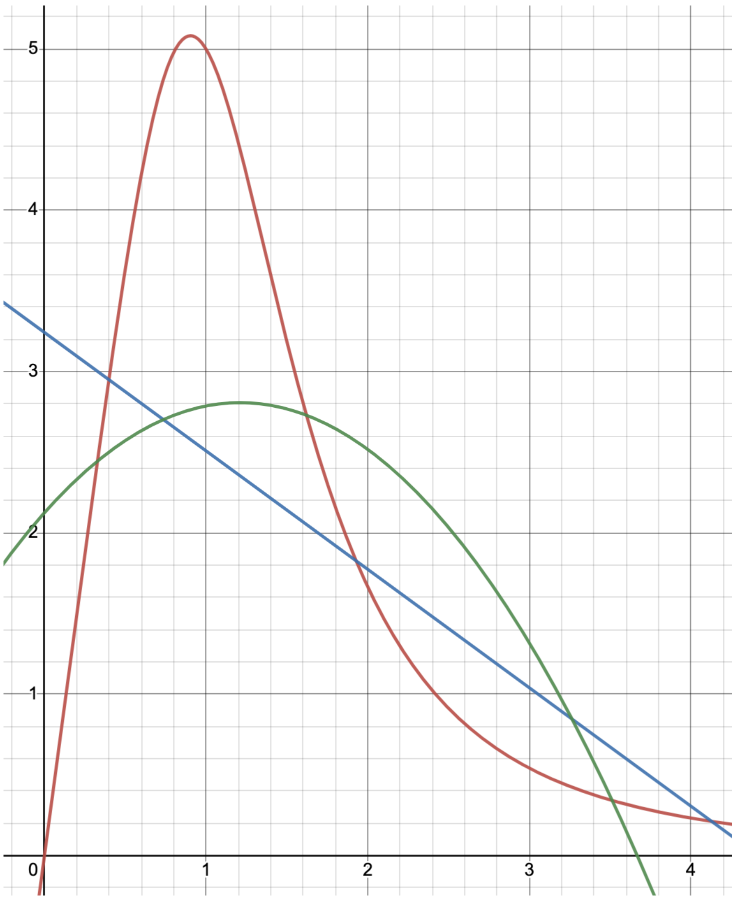
$b = 1.1268$
$c = 2.1261$

$\varphi_{квад}(x) =- 0.4653x^2 + 1.1268x + 2.1261$

$$\delta_{квад} = \sqrt{\frac{\sum_{i=0}^{n}(\varphi_{квад}(x_i)-y_i)^2}{n}} = \sqrt{\frac{(2.1261)^2+...}{11}} = \sqrt{\frac{16.7188}{11}} = \sqrt{1.5199} = 1.2328$$

5. Наилучшее приближение
   $\varphi(x) =- 0.4653x^2 + 1.1268x + 2.1261$ – квадратичная аппроксимация

6. Графики



Листинг программы

```
static func run(
  data: [Point]
) -> Output {
  [
    Function.linear(Self.linear(data: data)),
    Function.polynomial2(Self.polynomial2(data: data)),
    Function.polynomial3(Self.polynomial3(data: data)),
    Function.power(Self.power(data: data)),
```

```
    Function.exponential(Self.exponential(data: data)),
    Function.logarithmic(Self.logarithmic(data: data)),
  ].map { function in
    Output(data: data, function: function)
  }.min { lhs, rhs in
    lhs.standardDeviation < rhs.standardDeviation
  }!
}
```

```
private static func linear(
  data: [Point]
) -> Function.Linear {
  let n = Double(data.count)
  let sx = data.reduce(0.0) { $0 + $1.x }
  let sxx = data.reduce(0.0) { $0 + $1.x * $1.x }
  let sy = data.reduce(0.0) { $0 + $1.y }
  let sxy = data.reduce(0.0) { $0 + $1.x * $1.y }
  let matrix = simd_double2x2(
    [n, sx],
    [sx, sxx]
  )
  let results = SIMD2([sy, sxy])
  let coefficients = simd_mul(matrix.inverse, results)
  return .init(
    a0: coefficients.x,
    a1: coefficients.y
  )
}
```

```
private static func polynomial2(
  data: [Point]
) -> Function.Polynomial2 {
  let n = Double(data.count)
  let sx = data.reduce(0.0) { $0 + $1.x }
  let sxx = data.reduce(0.0) { $0 + $1.x * $1.x }
  let sxxx = data.reduce(0.0) { $0 + $1.x * $1.x * $1.x }
  let sxxxx = data.reduce(0.0) { $0 + $1.x * $1.x * $1.x * $1.x }
  let sy = data.reduce(0.0) { $0 + $1.y }
  let sxy = data.reduce(0.0) { $0 + $1.x * $1.y }
  let sxxy = data.reduce(0.0) { $0 + $1.x * $1.x * $1.y }
  let matrix = simd_double3x3(
    [n, sx, sxx],
    [sx, sxx, sxxx],
    [sxx, sxxx, sxxxx]
  )
  let results = SIMD3([sy, sxy, sxxy])
  let coefficients = simd_mul(matrix.inverse, results)
```

```
    return .init(
      a0: coefficients.x,
      a1: coefficients.y,
      a2: coefficients.z
    )
  }
```

```
  private static func polynomial3(
    data: [Point]
  ) -> Function.Polynomial3 {
    let n = Double(data.count)
    let sx = data.reduce(0.0) { $0 + $1.x }
    let sxx = data.reduce(0.0) { $0 + $1.x * $1.x }
    let sxxx = data.reduce(0.0) { $0 + $1.x * $1.x * $1.x }
    let sxxxx = data.reduce(0.0) { $0 + $1.x * $1.x * $1.x * $1.x }
    let sxxxxx = data.reduce(0.0) { $0 + $1.x * $1.x * $1.x * $1.x * $1.x }
    let sxxxxxx = data.reduce(0.0) { $0 + $1.x * $1.x * $1.x * $1.x * $1.x *
$1.x }
    let sy = data.reduce(0.0) { $0 + $1.y }
    let sxy = data.reduce(0.0) { $0 + $1.x * $1.y }
    let sxxy = data.reduce(0.0) { $0 + $1.x * $1.x * $1.y }
    let sxxxy = data.reduce(0.0) { $0 + $1.x * $1.x * $1.x * $1.y }
    let matrix = simd_double4x4(
      [n, sx, sxx, sxxx],
      [sx, sxx, sxxx, sxxxx],
      [sxx, sxxx, sxxxx, sxxxxx],
      [sxxx, sxxxx, sxxxxx, sxxxxxx]
    )
    let results = SIMD4([sy, sxy, sxxy, sxxxy])
    let coefficients = simd_mul(matrix.inverse, results)
    return .init(
      a0: coefficients.x,
      a1: coefficients.y,
      a2: coefficients.z,
      a3: coefficients.w
    )
  }
```

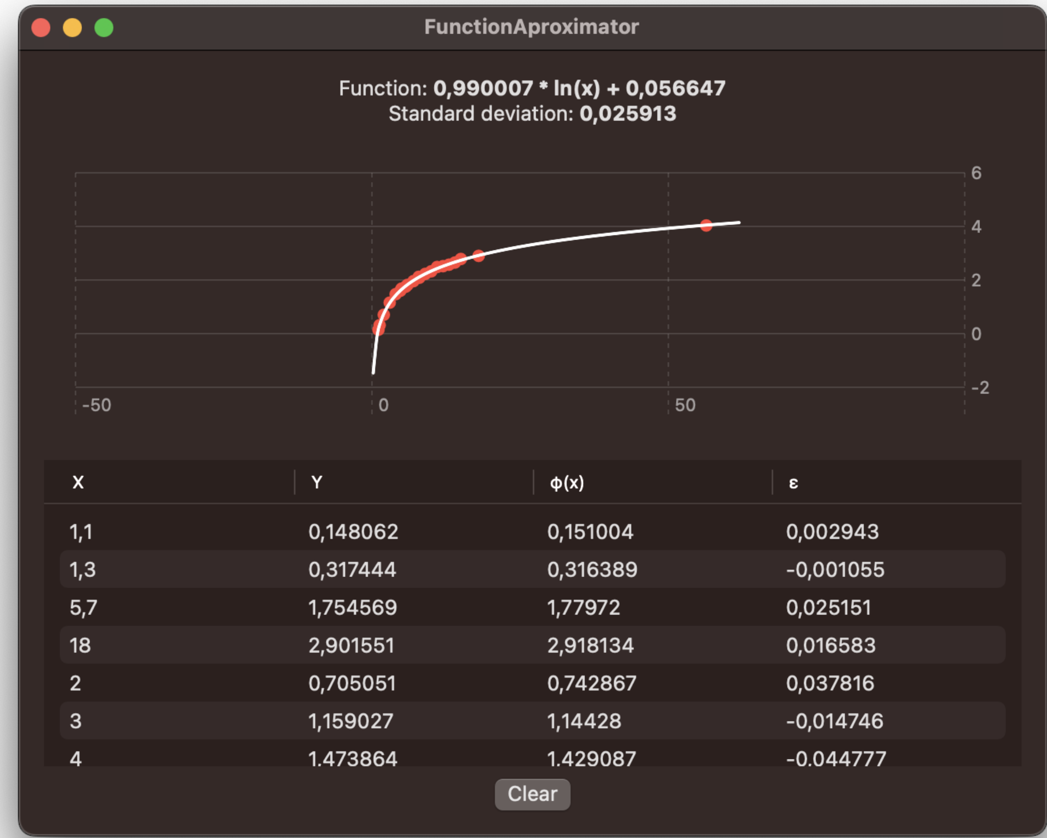
```
  private static func power(
    data: [Point]
  ) -> Function.Power {
    let linearData = data.map { Point(x: log($0.x), y: log($0.y)) }
    let linear = Self.linear(data: linearData)
    return .init(
      coefficient: exp(linear.a0),
      exponent: linear.a1
    )
```

```
}
```

```
private static func exponential(
  data: [Point]
) -> Function.Exponential {
  let linearData = data.map { Point(x: $0.x, y: log($0.y)) }
  let linear = Self.linear(data: linearData)
  return .init(
    coefficient: exp(linear.a0),
    exponentCoefficient: linear.a1
  )
}
```

```
private static func logarithmic(
  data: [Point]
) -> Function.Logarithmic {
  let powerData = data.map { Point(x: $0.x, y: exp($0.y)) }
  let power = Self.power(data: powerData)
  return .init(
    coefficient: power.exponent,
    freeCoefficient: log(power.coefficient)
  )
}
```

Результаты выполнения программы



Вывод

Во время выполнения лабораторной работы познакомился с методом аппроксимации функции. Научился использовать и реализовывать программно МНК. Получил ценные знания, которые несомненно пригодятся в будущем.