Университет ИТМО
Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

«Вычислительная математика»

Выполнил:
Студент группы P32102
Гулямов Т.И.

Преподаватель:
Рыбаков С.Д.

Санкт-Петербург
2023

Цель лабораторной работы

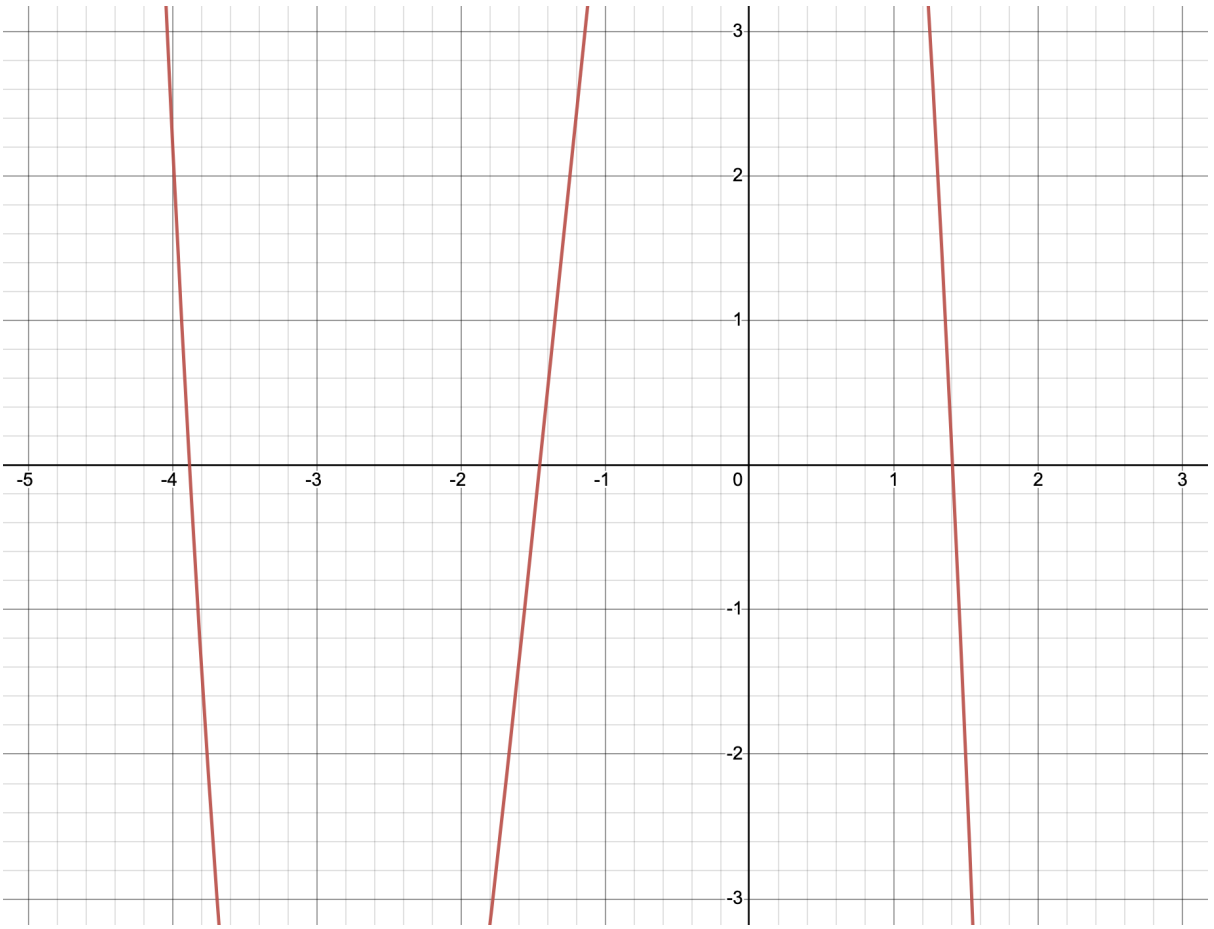
Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений, выполнить программную реализацию методов.

Порядок выполнения работы

1. Вычислительная реализация задачи
    a. Функция
    b. График функции
    c. Интервалы изоляции корней
    d. Уточнение корня уравнения методом секущих
    e. Уточнение корня уравнения методом половинного деления
    f. Уточнение корня уравнения методом простой итерации
2. Программная реализация задачи

Вычислительная реализация задачи

1. Функция: $-1,38x^3 - 5,42x^2 + 2,57x + 10,95$
2. График функции



3. Интервалы изоляции корней:
    a. Крайний левый корень: $a_1 = -4,\ b_1 = -3$

b. Центральный корень: $a_2 = -2,\ b_2 = -1$

c. Крайний правый корень: $a_3 = 1,\ b_3 = 2$

4. Уточнение корня уравнения методом секущих

№ итерации	x_{k-1}	x_k	x_{k+1}	$f(x_{k+1})$	$\lvert x_k - x_{k+1} \rvert$
1	-4	-3.5	-3.8495	-0.53870	0.34952
2	-3.5	-3.8495	-3.8893	0.15623	0.03978
3	-3.8495	-3.8893	-3.8804	-0.00293	0.00894

$x^* = x_4 \approx -3.8804$

5. Уточнение корня уравнения методом половинного деления

№ шага	a	b	x	$f(a)$	$f(b)$	$f(x)$	$\lvert a - b \rvert$
0	-2	-1	-1.5	-4.83	4.34	-0.4425	1
1	-1.5	-1	-1.25	-0.4425	4.34	1.964	0.5
2	-1.5	-1.25	-1.375	-0.4425	1.964	0.757	0.25
3	-1.5	-1.375	-1.438	-0.4425	0.757	0.155	0.125
4	-1.5	-1.438	-1.469	-0.4425	0.155	-0.144	0.0625
5	-1.469	-1.438	-1.453	-0.144	0.155	0.005	0.03125
6	-1.469	-1.453	-1.461	-0.144	0.005	-0.07	0.01563
7	-1.461	-1.453	-1.4609	-0.07	0.005	-0.07	0.00781

$x^* = x_7 \approx -1.4609$

6. Уточнение корня уравнения методом простой итерации

$f(x) = 0$

$\boldsymbol{\lambda} f(x) = 0$

$\boldsymbol{\lambda} f(x) + x = x$

$\boldsymbol{\varphi}(x) = x + \boldsymbol{\lambda} f(x)$

$\boldsymbol{\varphi}'(x) = 1 + \boldsymbol{\lambda} f'(x)$

Для высокой скорости сходимости:

$q = max_{[a,b]}(\boldsymbol{\varphi}'(x)) \approx 0 \Rightarrow \boldsymbol{\lambda} = -1/max_{[a,b]}(f'(x))$

Найдем $\boldsymbol{\lambda}$:

$f'(x) = -4.14x^2 - 10.84x + 2.57$

$f'(1) = -12.41$

$f'(2) = -35.67$
$\boldsymbol{\lambda} = 1/12.41 = 0.0806$

$\boldsymbol{\varphi}(x) = 0.882353 + 1.20709\,x - 0.436745\,x^2 - 0.111201\,x^3$

№ итерации	x_k	x_{k+1}	$\boldsymbol{\varphi}(x_{k+1})$	$f(x_{k+1})$	$\lvert x_k - x_{k+1} \rvert$
0	1	1.5415	1.29796	-3.02234	0.5415
1	1.5415	1.29796	1.47016	2.13707	0.24354
2	1.29796	1.47016	1.35965	-1.37135	0.1722
3	1.47016	1.35965	1.43668	0.955978	0.11051
4	1.35965	1.43668	1.38534	-0.637103	0.07703
5	1.43668	1.38534	1.42075	0.439435	0.05134
6	1.38534	1.42075	1.39684	-0.29671	0.03541
7	1.42075	1.39684	1.41323	0.203445	0.02391
8	1.39684	1.41323	1.40211	-0.138017	0.01639
9	1.41323	1.40211	1.40971	0.09431	0.01112
10	1.40211	1.40971		-0.0641738	0.0076

$x^* = x_{11} \approx 1.40971$

Листинг программы

Метод Ньютона

```
struct Newton {
  let function: Function
  let firstDerivative: Function
  let secondDerivative: Function
  let interval: Interval
  let tolerance: Double

  func run() throws -> Output {
    guard self.function(self.interval.from) *
          self.function(self.interval.to) < 0 else {
      throw Error.incorrectRootsCount
    }
    let x = self.function(self.interval.from) *
            self.secondDerivative(self.interval.from) > 0
      ? self.interval.from
      : self.interval.to
    return try self.run(x: x, iterationsCount: 0)
  }
```

```
  private func run(x: Double, iterationsCount: Int) throws -> Output {
    let y = self.function(x)
    guard abs(y) > self.tolerance else {
      return Output(root: x, valueInRoot: y, iterationsCount: iterationsCount)
    }
    let dy = self.firstDerivative(x)
    guard abs(dy) > .ulpOfOne else {
      throw Error.derivativeIsZero
    }
    let nextX = x - y / dy
    guard abs(nextX - x) > self.tolerance else {
      let valueInRoot = self.function(nextX)
      return Output(
        root: x,
        valueInRoot: valueInRoot,
        iterationsCount: iterationsCount + 1)
    }
    return try self.run(x: nextX, iterationsCount: iterationsCount + 1)
  }
}
```

Метод хорд

```
struct Secant {
  let function: Function
  let interval: Interval
  let tolerance: Double

  func run() throws -> Output {
    guard self.function(self.interval.from) *
          self.function(self.interval.to) < 0 else {
      throw Error.incorrectRootsCount
    }
    return self.run(
      from: self.interval.from,
      to: self.interval.to,
      iterationsCount: 0
    )
  }

  private func run(from: Double, to: Double, iterationsCount: Int) -> Output {
    let fromY = self.function(from)
    let toY = self.function(to)
    let nextX = (from * toY - to * fromY) / (toY - fromY)
    let y = self.function(nextX)
    guard abs(y) > self.tolerance else {
      return Output(root: nextX, valueInRoot: y, iterationsCount: iterationsCount)
    }
    guard abs(nextX - from) > self.tolerance, abs(nextX - to) > self.tolerance else {
      return Output(root: nextX, valueInRoot: y, iterationsCount: iterationsCount + 1)
    }
    return self.run(
      from: fromY * y < 0 ? from : nextX,
      to: toY * y < 0 ? to : nextX,
      iterationsCount: iterationsCount + 1
```

```
    )
  }
}
```

Метод простой итерации

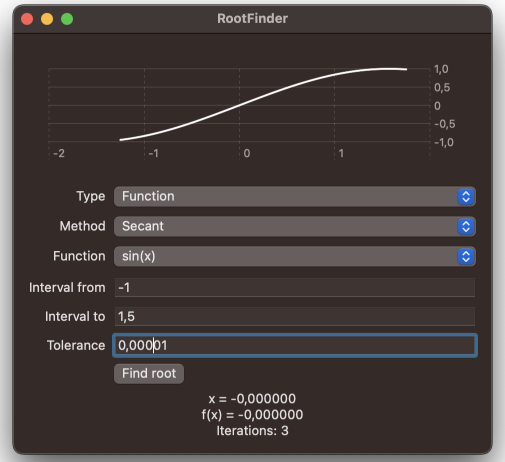
```
struct SimpleIteration {
  let function: Function
  let derivative: Function
  let interval: Interval
  let tolerance: Double

  func run() throws -> Output {
    guard self.function(self.interval.from) *
          self.function(self.interval.to) < 0 else {
      throw Error.incorrectRootsCount
    }
    let lambda = -1 / max(
      self.derivative(self.interval.from),
      self.derivative(self.interval.to)
    )
    let phi = { x in
      x + lambda * self.function(x)
    }
    let phiDerivative = { x in
      1 + lambda * self.derivative(x)
    }
    guard max(
      phiDerivative(self.interval.from),
      phiDerivative(self.interval.to)
    ) < 1 else {
      throw Error.phiDerivativeNotLessOne
    }
    return self.run(x: self.interval.from, iterationsCount: 0, phi: phi)
  }

  private func run(x: Double, iterationsCount: Int, phi: Function) -> Output {
    let nextX = phi(x)
    guard abs(nextX - x) > self.tolerance else {
      let valueInRoot = self.function(nextX)
      return Output(
        root: x,
        valueInRoot: valueInRoot,
        iterationsCount: iterationsCount + 1)
    }
    return self.run(x: nextX, iterationsCount: iterationsCount + 1, phi: phi)
  }
}
```

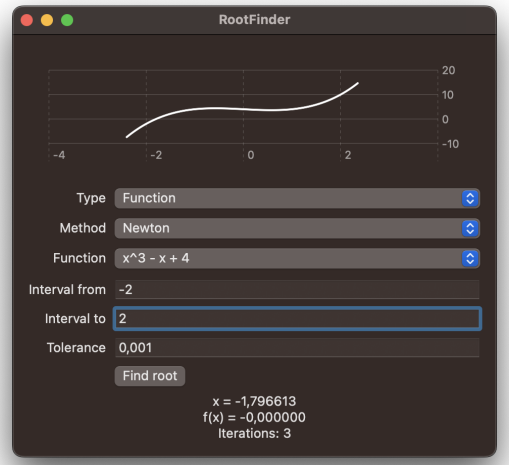
Метод простой итерации (система)

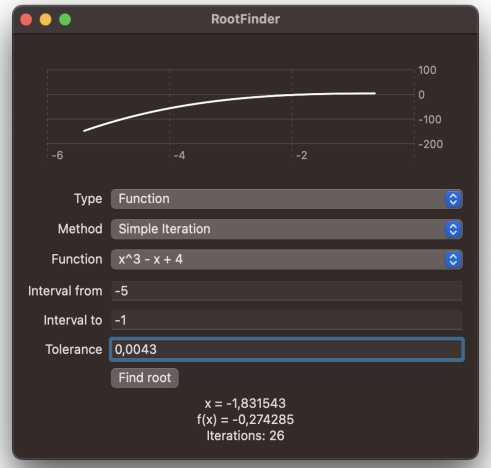
```
struct SystemSimpleIteration {
  let function1: Function2
  let function2: Function2

  let phiFunction1: Function2
```

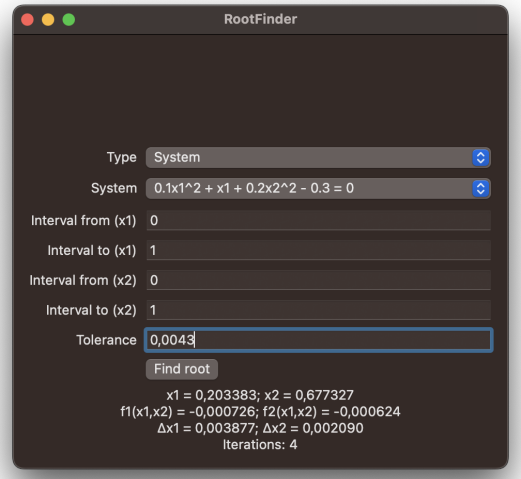
```
  let phiFunction2: Function2

  let phiDerivative11: Function2
  let phiDerivative12: Function2
  let phiDerivative21: Function2
  let phiDerivative22: Function2

  let interval1: Interval
  let interval2: Interval
  let tolerance: Double

  func run() throws -> SystemOutput {
    try self.run(
      x1: self.interval1.to,
      x2: self.interval2.to,
      iterationsCount: 0
    )
  }

  private func run(
    x1: Double,
    x2: Double,
    iterationsCount: Int
  ) throws -> SystemOutput {
    guard self.phiDerivative11(x1, x2) + self.phiDerivative12(x1, x2) < 1,
          self.phiDerivative21(x1, x2) + self.phiDerivative22(x1, x2) < 1 else {
      throw Error.common
    }
    let nextX1 = self.phiFunction1(x1, x2)
    let nextX2 = self.phiFunction2(x1, x2)
    let errorX1 = abs(nextX1 - x1)
    let errorX2 = abs(nextX2 - x2)
    guard errorX1 < self.tolerance, errorX2 < self.tolerance else {
      return try self.run(
        x1: nextX1,
        x2: nextX2,
        iterationsCount: iterationsCount + 1
      )
    }
    return .init(
      roots: (nextX1, nextX2),
      valuesInRoots: (
        self.function1(nextX1, nextX2),
        self.function2(nextX1, nextX2)
      ),
      errors: (errorX1, errorX2),
      iterationsCount: iterationsCount
    )
  }
}
```

Результаты выполнения программы









Вывод

Во время выполнения лабораторной работы познакомился с численными методами решения нелинейных уравнений и систем. Научился использовать и реализовывать программно некоторые популярные методы. Получил ценные знания, которые несомненно пригодятся в будущем.