Университет ИТМО

# Вычислительная математика
# Лабораторная работа №2
# «Численное решение нелинейных уравнений и систем»

Работу выполнил:
Бавыкин Роман
Группа: P3210
Вариант 2

Санкт-Петербург
2022 г.

**Цель работы:**
Изучить численные методы решения нелинейных уравнений и их систем, найти корни заданного нелинейного уравнения, выполнить программную реализацию методов.

**Порядок выполнения работы:**

1. № варианта определяется как номер в списке группы согласно ИСУ. .
2. Отделить корни заданного нелинейного уравнения графически (см. табл. 5)
3. Определить интервалы изоляции корней.
4. Вычислительная реализация задачи (в отчет):

Уточнить корни нелинейного уравнения (см. табл.5) с точностью ε=10-2. Вычисления оформить в виде таблиц (1-4), удержать 3 знака после запятой.

Представить в отчете заполненные таблицы (1-4). В таблице 6 указаны методы для каждого из 3-х корней многочлена.

4.1 Для метода половинного деления или метода хорд заполнить таблицу 1.
4.2 Для метода Ньютона или метода секущих заполнить таблицу 2.
4.3 Для метода секущих заполнить таблицу 3.
4.4 Для метода простой итерации заполнить таблицу 4.

5. Программная реализация задачи:

Для нелинейных уравнений:

5.2 Все численные методы (см. табл. 7) должны быть реализованы в виде отдельных подпрограмм или классов.

5.3 Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.

5.4 Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.

5.5 Выполнить верификацию исходных данных. Для метода половинного деления (метода хорд) анализировать наличие корня на введенном интервале. Для метода Ньютона (метода секущих) – выбор начального приближения (a или b). Для метода простой итерации – достаточное условие сходимости метода. Программа должна реагировать на некорректные введенные данные.

5.6 Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.

5.7 Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

Для систем нелинейных уравнений:

5.8 Рассмотреть систему двух уравнений.
5.9 Организовать вывод графика функций.
5.10 Для метода простой итерации проверить достаточное условие сходимости.
5.11 Вывод вектора неизвестных:
5.12 Вывод количества итераций, за которое было найдено решение.
5.13 Вывод вектора погрешностей:

**Рабочие формулы используемых методов:**

$$\textit{Метод Ньютона}: x_i = x_{i-1} - \frac{f(x_{i-1})}{(x_{i-1})}$$

$$\textit{Метод простой итерации}: x_{i+1} = \phi(x_i)$$

**Заполненные таблицы:**

$$-1{,}38\,x^3-5{,}42\,x^2+2{,}57\,x+10{,}95$$

| № варианта | *Крайний правый корень* | *Крайний левый корень* | *Центральный корень* |
|---|---|---|---|
| 2 | 5 | 2 | 1 |

Крайний правый корень: метод простой итерации

$$\lambda=-\frac{1}{max\,f\,'(x)}$$

$$f\,'(x)=-4{,}14\,x^2-10{,}84\,x+2{,}57$$

$$f\,'(1)=-12{,}41$$

$$f\,'(2)=-35{,}67$$

$$\lambda=-\frac{1}{-12{,}41}=0.08058$$

$$\phi(x)=-0{,}1112\,x^3-0{,}4367\,x^2+1{,}2071\,x+0.8824$$

$$\epsilon=0.01$$

| № итерации | $x_k$ | $f(x_k)$ | $x_{k+1}$ | $\varphi(x_k)$ | $\lvert x_k - x_{k+1}\rvert$ |
|---|---|---|---|---|---|
| 1 | 2 | -16.63 | 0.6600 | 0.6600 | 1.34 |
| 2 | 0.6600 | 9.8888 | 1.4568 | 1.4568 | 0.7968 |
| 3 | 1.4568 | -1.0751 | 1.3702 | 1.3702 | 0.0866 |
| 4 | 1.3702 | 0.7465 | 1.4303 | 1.4303 | 0.0601 |
| 5 | 1.4303 | -0.5003 | 1.3900 | 1.3900 | 0.0403 |
| 6 | 1.3900 | 0.3442 | 1.4177 | 1.4177 | 0.0277 |
| 7 | 1.4177 | -0.2329 | 1.3990 | 1.3990 | 0.0188 |
| 8 | 1.3990 | 0.1595 | 1.4118 | 1.4118 | 0.0129 |
| 9 | 1.4118 | -0.1084 | 1.4031 | 1.4031 | 0.0087 |

Крайний левый корень: метод хорд

$a=-4\,b=-3$

| № шага | a | b | x | f(a) | f(b) | f(x) | $\lvert x_k - x_{k+1}\rvert$ |
|---|---|---|---|---|---|---|---|
| 1 | -4 | -3 | -3.7848 | 2.2700 | -8.2800 | -1.5980 | 0.2152 |
| 2 | -4 | -3.7848 | -3.8737 | 2.2700 | -1.5980 | -0.1198 | 0.0889 |
| 3 | -4 | -3.8737 | -3.8801 | 2.2700 | -0.1198 | -0.0082 | 0.0063 |

Центральный корень — метод половинного деления

$a=-2\,b=-1$

| № шага | a | b | x | f(a) | f(b) | f(x) | \|a-b\| |
|---|---|---|---|---|---|---|---|
| 1 | -2 | -1 | -1.5 | -4.8300 | 4.34 | -0.4425 | 1 |
| 2 | -1.5 | -1 | -1.25 | -0.4425 | 4.34 | 1.9641 | 0.5 |
| 3 | -1.5 | -1.25 | -1.375 | -0.4425 | 1.9641 | 0.7565 | 0.25 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | -1.5 | -1.375 | -1.4375 | -0.4425 | 0.7565 | 0.1549 | 0.125 |
| 5 | -1.5 | -1.4375 | -1.46875 | -0.4425 | 0.1549 | -0.1444 | 0.0625 |
| 6 | -1.46875 | -1.4375 | -1.453125 | -0.1444 | 0.1549 | 0.0051 | 0.03125 |
| 7 | -1.46875 | -1.453125 | -1.4609375 | -0.1444 | 0.0051 | -0.0697 | 0.015625 |
| 8 | -1.4609375 | -1.453125 | -1.45703125 | -0.0697 | 0.0051 | -0.0323 | 0.0078125 |

**Листинг программы:**

Метод Ньютона:

```
public EquationResults solve() {
    if (function.getValue(inputData.getA()) * function.getValue(inputData.getB()) >= 0) {
        throw new SufficientConditionException(0,0);
    }
    EquationResults results = new EquationResults();
    results.setI(0);
    if (inputData.getA() * function.getSecondDerivative(inputData.getA()) > 0) {
        results.setX(inputData.getA());
    } else {
        results.setX(inputData.getB());
    }
    double oldX;
    do {
        oldX = results.getX();
        results.setX(oldX - function.getValue(oldX) / function.getFirstDerivative(oldX));
        results.setI(results.getI() + 1);
    } while (Math.abs(results.getX() - oldX) > inputData.getEps());
    results.setF(function.getValue(results.getX()));
    return results;
}
```

Метод простых итераций для уравнений:

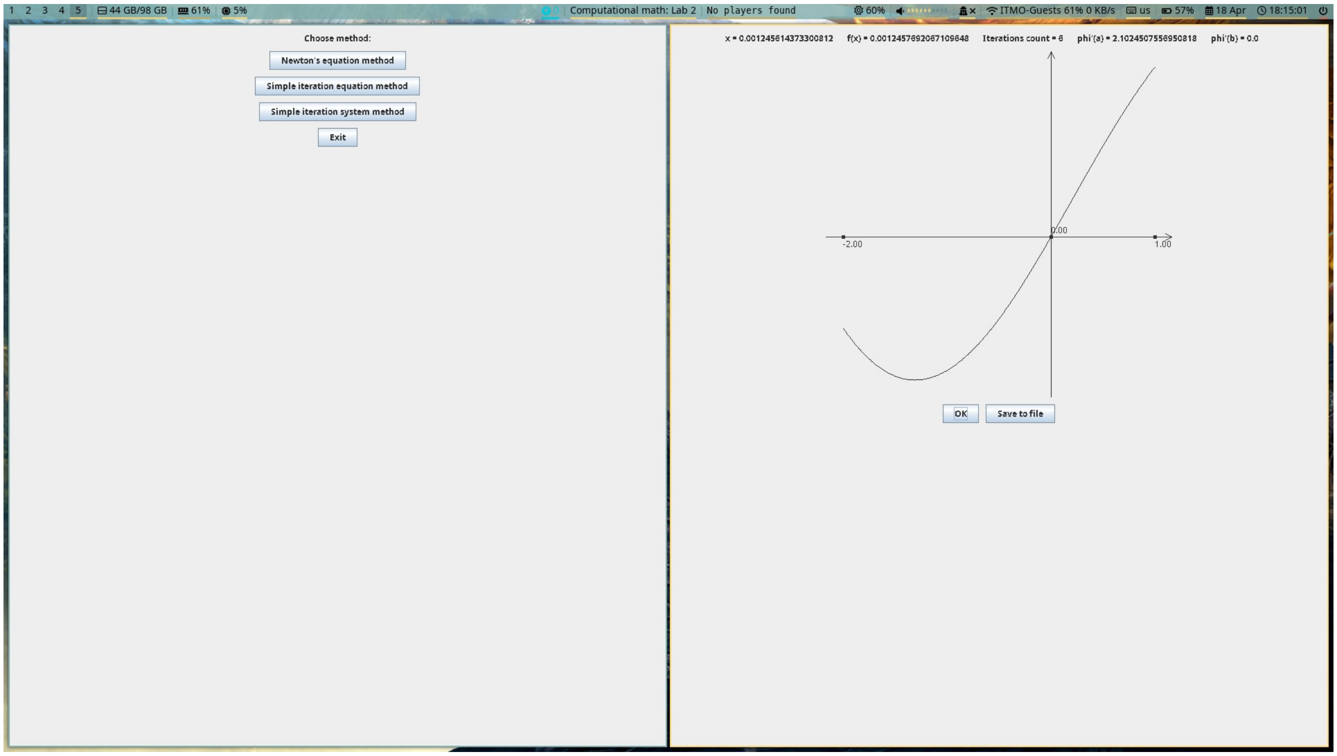
```
public EquationResults solve() {
    EquationResults results = new EquationResults();
    double lambda;
    if (function.getFirstDerivative(inputData.getA()) > function.getFirstDerivative(inputData.getB())) {
        results.setX(inputData.getA());
        lambda = -1 / function.getFirstDerivative(inputData.getA());
    } else {
        results.setX(inputData.getB());
        lambda = -1 / function.getFirstDerivative(inputData.getB());
    }
    results.setPhiA(Math.abs(1 + lambda * function.getFirstDerivative(inputData.getA())));
    results.setPhiB(Math.abs(1 + lambda * function.getFirstDerivative(inputData.getB())));
    if (Math.abs(1 + lambda * function.getFirstDerivative(inputData.getA())) >= 1 &&
            Math.abs(1 + lambda * function.getFirstDerivative(inputData.getB())) >= 1) {
        throw new SufficientConditionException(Math.abs(1 + lambda *
                function.getFirstDerivative(inputData.getA())),
                Math.abs(1 + lambda * function.getFirstDerivative(inputData.getB())));
    }
    results.setI(0);
    double oldX;
    do {
        oldX = results.getX();
        results.setX(oldX + lambda * function.getValue(oldX));
        results.setI(results.getI() + 1);
        if (results.getI() > 100) {
```

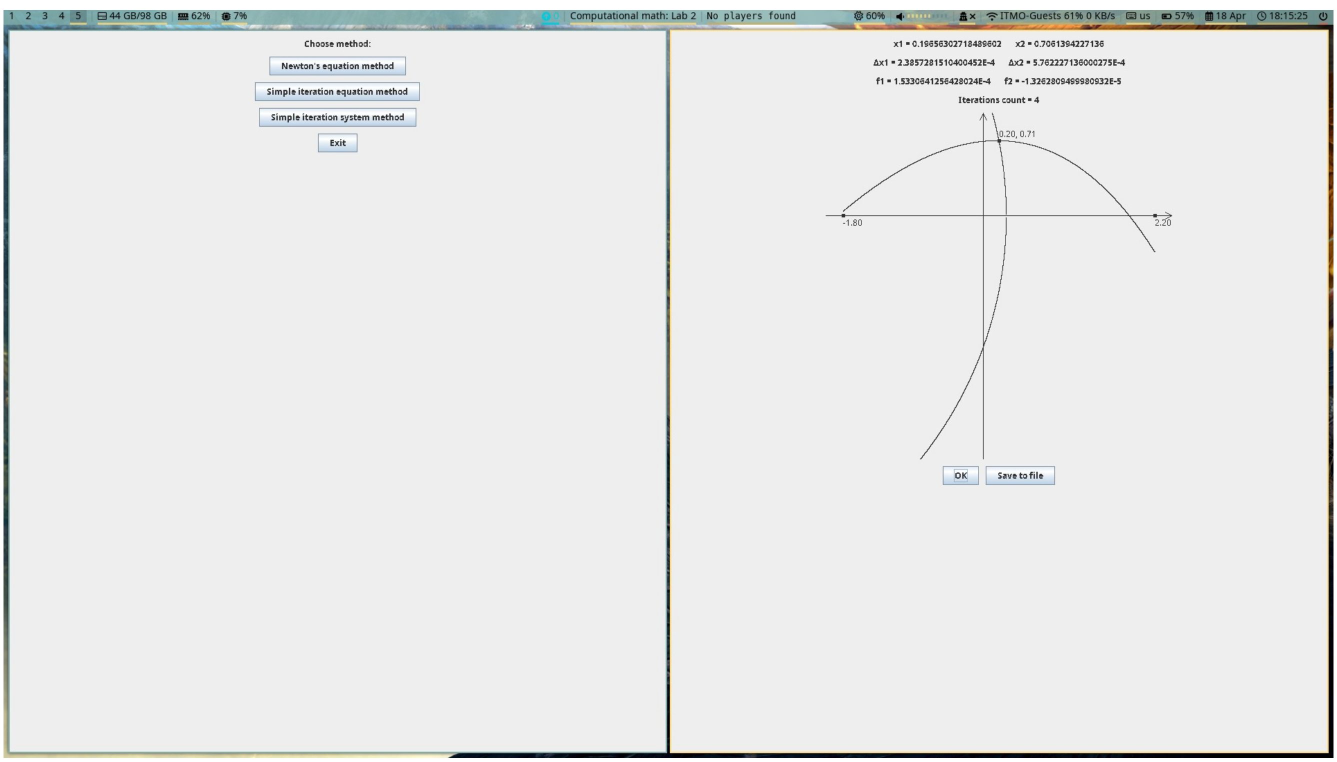
```
                throw new CountIteratioNException(results.getI());
            }
        } while (Math.abs(results.getX() - oldX) > inputData.getEps());
        results.setF(function.getValue(results.getX()));
        return results;
    }
```

Метод простых итераций для системы:

```
public SystemResults solve() {
    if (Math.abs(system.getPhiDerivative(1, 1, inputData.getX1(), inputData.getX2())) +
            Math.abs(system.getPhiDerivative(1, 2, inputData.getX1(), inputData.getX2())) >= 1 &&
            Math.abs(system.getPhiDerivative(2, 1, inputData.getX1(), inputData.getX2())) +
            Math.abs(system.getPhiDerivative(2, 2, inputData.getX1(), inputData.getX2())) >= 1) {
        throw new SufficientConditionException(0, 0);
    }
    SystemResults results = new SystemResults();
    results.setX1(inputData.getX1());
    results.setX2(inputData.getX2());
    results.setI(0);
    double oldX1;
    double oldX2;
    do {
        oldX1 = results.getX1();
        oldX2 = results.getX2();
        results.setX1(system.getPhi(1, oldX1, oldX2));
        results.setX2(system.getPhi(2, oldX1, oldX2));
        results.setI(results.getI() + 1);
    } while (Math.max(Math.abs(results.getX1() - oldX1),
            Math.abs(results.getX2() - oldX2)) > inputData.getEps());
    results.setErrorX1(Math.abs(results.getX1() - oldX1));
    results.setErrorX2(Math.abs(results.getX2() - oldX2));
    results.setF1(system.getValue(1, results.getX1(), results.getX2()));
    results.setF2(system.getValue(2, results.getX1(), results.getX2()));
    return results;
}
```

## Результаты выполнения программы:




**Выводы:** во время выполнения лабораторной работы ознакомился с методами решения нелинейных уравнений и систем нелинейных уравнений. С помощью методов простых итераций, хорд и половинного деления нашёл по одному из трёх корней данного уравнения. Реализовал программно метод Ньютона и метод простых итераций для уравнений и для систем.