Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа **№2**

# «Численное решение нелинейных уравнений и систем»

по дисциплине «Вычислительная математика»

Вариант: **2**

**Преподаватель:**
Машина Е. А.

**Выполнил:**
Вальц Мартин
**Группа:** P3210

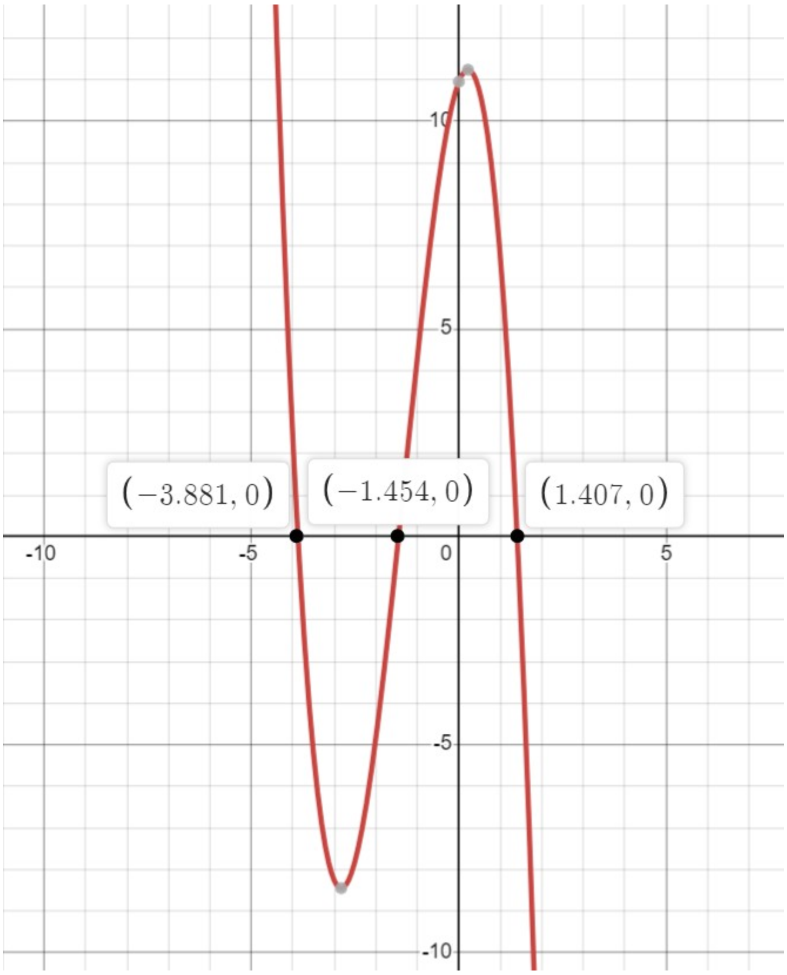
Санкт-Петербург, 2024 г.

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

# 1. Вычислительная реализация задачи

## 1. Решение нелинейного уравнения

**1.** $-1,38x^3-5,42x^2+2,57x+10,95$




**2.**

Для определения интервалов изоляции корней данного уравнения, можно воспользоваться методом интервалов знакопеременности. Для этого нужно найти значения функции на различных интервалах и определить знак функции на каждом из них.

Получим приближенные значения корней:
x ≈ -3.9, x ≈ -1.5, x ≈ 1.4

Теперь нужно разбить ось x на 4 интервала: (-∞, -3.9), (-3.9, -1.5), (-1.5, 1.4) и (1.4, +∞). На каждом из этих интервалов нужно определить знак функции.

Для этого можем вычислить значения функции в произвольной точке каждого интервала. Например, для интервала (-∞, -3.9) можно выбрать x = -4, для интервала (-3.9, -1.5) x = -2, для интервала (-1.5, 1.4) x = 0, и для интервала (1.4, +∞) x = 2.

Таким образом, получим следующие значения функции:

для x = -4: f(-4) = 2.27

для x = -2: f(-2) = -4.83

для x = 0: f(0) = 10.95

для x = 2: f(2) = -16.63

Знаки функции на каждом интервале будут соответственно:

| (-∞, -3.9) | (-3.9, -1.5) | (-1.5, 1.4) | (1.4, +∞) |
|---|---|---|---|
| + | - | + | - |

Таким образом, мы получаем два интервала изоляции корней уравнения:

(-4, -1.5), (-1.5, 1) и (1, 1.5).

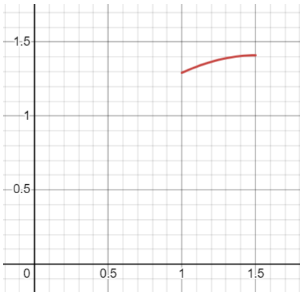
**3.**

$x_1 \approx -3{,}88$
$x_2 \approx -1{,}45$
$x_3 \approx 1{,}41$

**4.**

Крайний правый корень – **Метод простой итерации**

$\varphi(x)$

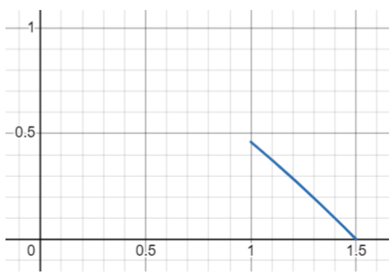



sПроверка **условия сходимости** метода на выбранном интервале:
$f(x) = -1{,}38x^3 - 5{,}42x^2 + 2{,}57x + 10{,}95 = 0$
$f'(x) = -4{,}14x^2 - 10{,}84x + 2{,}57$
$f'(a) = -12{,}41 < 0, f'(b) = -23{,}005 < 0$
*max*

$$\varphi(x) = x + \lambda f(x) = x + \frac{-1{,}38x^3 - 5{,}42x^2 + 2{,}57x + 10{,}95}{23{,}005}$$

$$\varphi'(x) = 1 + \lambda f'(x) = 1 + \frac{-4{,}14x^2 - 10{,}84x + 2{,}57}{23{,}005}$$

$\varphi'(x)$




На отрезке начального приближения [1, 1.5] функция $\varphi(x)$ определена, непрерывна и дифференцируема.

$|\varphi'(a)| = 0{,}461$
$|\varphi'(b)| = 0$
$|\varphi'(x)| \leq q, где\, q = 0{,}461$

$0 \le q < 1 \rightarrow$ **итерационная последовательность сходится,** скорость сходимости высокая,
$0 \le q < 0{,}5 \rightarrow$ критерий окончания итерационного процесса $|x_{k+1} - x_k| \le \varepsilon$, $x_0 = 1.5$

| № | $x_k$ | $x_{k+1}$ | $f(x_{k+1})$ | $\lvert x_{k+1} - x_k \rvert$ |
|---|---|---|---|---|
| 1 | 1.500 | 1.411 | -0.091 | 0.089 |
| 2 | 1.411 | 1.40704 | -0.00834798 | 0.00396 |
| 3 | 1.40704 | 1.40668 | -0.000833168 | 0.00036 |
| 4 | 1.40668 | **1.40664** | 0.00000163104 | 0.00004 |

Крайний левый корень – **Метод хорд**

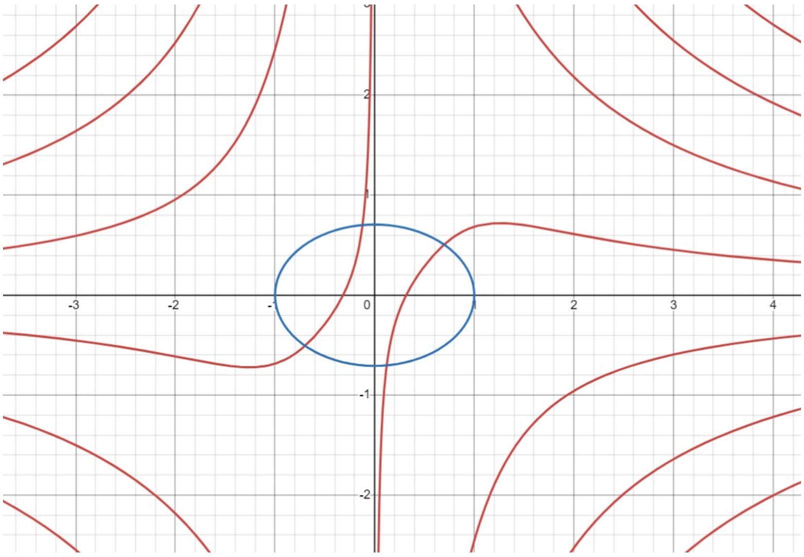
| № | a | b | x | f(a) | f(b) | f(x) | $\lvert x_{k+1} - x_k \rvert$ |
|---|---|---|---|---|---|---|---|
| 1 | -4.000 | -1.908 | -3.254 | 2.270 | -4.098 | -7.253 | 1.346 |
| 2 | -4.000 | -3.254 | -3.822 | 2.270 | -7.253 | -0.996 | 0.568 |
| 3 | -4.000 | -3.822 | -3.876 | 2.270 | -0.996 | -0.072 | 0.054 |
| 4 | -4.000 | -3.876 | **-3.880** | 2.270 | -0.072 | -0.005 | 0.004 |

Центральный корень – **Метод половинного деления**

| № | a | b | x | f(a) | f(b) | f(x) | $\lvert a – b \rvert$ |
|---|---|---|---|---|---|---|---|
| 1 | -1.500 | 1.000 | -0.250 | -0.443 | 6.720 | 9.990 | 2.500 |
| 2 | -1.500 | -0.250 | -0.875 | -0.443 | 9.990 | 5.476 | 1.250 |
| 3 | -1.500 | -0.875 | -1.188 | -0.443 | 5.476 | 2.566 | 0.625 |
| 4 | -1.500 | -1.188 | -1.344 | -0.443 | 2.566 | 1.058 | 0.312 |
| 5 | -1.500 | -1.344 | -1.422 | -0.443 | 1.058 | 0.305 | 0.156 |
| 6 | -1.500 | -1.422 | -1.461 | -0.443 | 0.305 | -0.070 | 0.078 |
| 7 | -1.461 | -1.422 | -1.441 | -0.070 | 0.305 | 0.117 | 0.039 |
| 8 | -1.461 | -1.441 | -1.451 | -0.070 | 0.117 | 0.024 | 0.020 |
| 9 | -1.461 | -1.451 | **-1.456** | -0.070 | 0.024 | -0.023 | 0.010 |

## 2. Решение системы нелинейных уравнений

**1.** $\begin{cases} tg(xy+0.1)=x^2 \\ x^2+2y^2=1 \end{cases}$, Метод Ньютона




**2.**

$$\begin{cases} tg(xy+0.1)=x^2 \\ x^2+2y^2=1 \end{cases} \rightarrow \begin{cases} f(x,y)=0 \\ g(x,y)=0 \end{cases} \rightarrow \begin{cases} tg(xy+0.1)-x^2=0 \\ x^2+2y^2-1=0 \end{cases}$$

Отметим, что решение системы уравнений являются точки пересечения эллипса и $tg(xy+0.1)-x^2=0$ , следовательно, система имеет не более четырех различных решений.

Построим матрицу Якоби:

$$\frac{\partial f}{\partial x}=y\,sec(xy+0.1)-2,\ \frac{\partial f}{\partial y}=x\,sec^2(xy+0.1),\ \frac{\partial g}{\partial x}=2x,\ \frac{\partial g}{\partial y}=4y$$

$$\begin{vmatrix} \frac{\partial f(x,y)}{\partial x} & \frac{\partial f(x,y)}{\partial y} \\ \frac{\partial g(x,y)}{\partial x} & \frac{\partial g(x,y)}{\partial y} \end{vmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = -\begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix}$$

$$\begin{vmatrix} y\,sec(xy+0.1)-2 & x\,sec^2(xy+0.1) \\ 2x & 4y \end{vmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} x^2-tg(xy+0.1) \\ 1-x^2-2y^2 \end{pmatrix}$$

$$\begin{cases} ysec(xy+0.1)\Delta x-2\Delta x+x\,sec^2(xy+0.1)\Delta y=x^2-tg(xy+0.1) \\ 2x\Delta x+4y\Delta y=1-x^2-2y^2 \end{cases}$$

**Корень 1:** Шаг 1: Выбираем $x_0=-0.12;\ y_0=0.7$

$$\begin{cases} ysec(xy+0.1)\Delta x-2\Delta x+x\,sec^2(xy+0.1)\Delta y=x^2-tg(xy+0.1) \\ 2x\Delta x+4y\Delta y=1-x^2-2y^2 \end{cases}$$

Шаг 2. Решаем полученную систему.

$\begin{cases} \Delta x+0.077\Delta y=0.0154 \\ -0.2\Delta x+2.8\Delta y=0.01 \end{cases} \rightarrow \Delta x=-0.0014; \Delta y=0.0019$

Шаг 3. Вычисляем очередные приближения:

$x_1=x_0+\Delta x=-0.12-0.0014=-0.1214$

$y_1=y_0+\Delta y=0.7+0.0019=0.7019$

$|x_1-x_0|\leq\varepsilon,\ |y_1-y_0|\leq\varepsilon$

$|-0.1214+0.12|\leq\varepsilon,\ |0.7019-0.7|\leq\varepsilon$ $\rightarrow$ ответ найден, **корень 1**: $(-0.1214, 0.7019)$

Аналогично находим **другой корень**: $(0.698, 0.506)$

Из графического решения, корни симметричны, следовательно, **другие 2 корня** $(-0.698, -0.506), (0.1214, -0.7019)$

## 2. Программная реализация задачи

```
package lab2.algebra;

public class NonLinearEquationSolver implements EquationSolver {
    private final int LIMIT = 1_000_000;

    private double ACCURACY = 1E-5;

    /**
     * Устанавливает точность
     */

    @Override
    public void setAccuracy(double accuracy) {
        if (0 > accuracy && accuracy > 1) {
            throw new IllegalArgumentException("Точность: можно указать тольков в интервале (0, 1)");
        }
        this.ACCURACY = accuracy;
    }

    /**
```

```java
 * Решить методом Хорд
 */
@Override
public Object[] solveByChord(Function function, double a, double b) {
    double xn;
    long iterations = 0;
    do {
        xn = a - (((b - a) * function.apply(a)) / (function.apply(b) - function.apply(a)));
        double f_xn = function.apply(xn);
        if (function.apply(a) * f_xn < 0) {
            b = xn;
        } else if (function.apply(b) * f_xn < 0) {
            a = xn;
        } else {
            throw new RuntimeException("На данном интервале либо несколько корней, либо они
отсутствуют");
        }
        iterations++;
        if (iterations == LIMIT) throw new RuntimeException("Превышен лимит итераций.");
    } while (Math.abs(function.apply(xn)) >= ACCURACY);
    double root = xn;
    double delta = Math.abs(function.apply(xn));
    return new Object[] { root, delta, iterations, function.apply(root) };
}


/**
 *  Решить нелинейное уравнение методом простых итераций
 */

@Override
public Object[] solveByIteration(Function function, double a, double b) {
    if (a > b) a = a + b - (b = a);
    double q = derivativeSeriesMax(function, a, b);
    if (Double.isNaN(q) || q >= 1) {
        throw new RuntimeException("Необходимое условие сходимости не соблюдается");
    }
    int iterations = 0;
    double delta, k = (1 - q) / q, prev, root = (a + b) / 2;
    do {
        prev = root;
        root = function.apply(root);
        delta = root > prev ? root - prev : prev - root;
        iterations++;
    } while (delta > k * ACCURACY && iterations < LIMIT);
    if (iterations == LIMIT) {
        throw new RuntimeException("Указанная точность не достигнута");
    }
    if (a == -5) root *= -1;
    return new Object[] { root, delta / k, iterations, function.apply(root) };
}

/**
```

```
     * решить уравнение методом Ньютона
     */

    @Override
    public Object[] solveByNewton(Function function, double a, double b) {
        double x0;
        if (function.apply(a) * function.derivative2(a, 1e-9) > 0) x0 = a;
        else x0 = b;
        double xi = x0;

        long iterations = 0;
        do {
            xi = xi - (function.apply(xi) / function.derivative(xi, 1e-9));
            iterations++;
            if (iterations == LIMIT) throw new RuntimeException("Превышено максимальное
количество итераций");
        } while (Math.abs(function.apply(xi)) > ACCURACY);
        double root = xi;
        double delta = Math.abs(function.apply(xi));

        return new Object[] { root, delta, iterations, function.apply(root) };
    }

    /**
     * поиск максимального значение производной функции на отрезке [a, b]
     */
    private double derivativeSeriesMax(Function function, double a, double b) {
        double max = 0, delta = (b - a) / 1000000;

        if (a == b) return Math.abs(function.derivative(0, 1e-9));

        for (double point = a; point <= b; point += delta) {
            max = Math.max(max, Math.abs(function.derivative(point, 1e-9)));
        }
        return max;
    }

    @Override
    public Object[][] solveByIterations(double[] G, Function... functions) {
        int iters = 0;
        for (double x = G[0] + 1e-5; x < G[1]; x += (G[1] - G[0]) / 1000d) {
            for (double y = G[2] + 1e-5; y < G[3]; y += (G[3] - G[2]) / 1000d) {
                double temp1 = functions[6].apply(x, y);
                double temp2 = functions[7].apply(x, y);
                double temp3 = functions[8].apply(x, y);
                double temp4 = functions[9].apply(x, y);
                if (Math.abs(temp1) + Math.abs(temp2) >= 1 || Math.abs(temp3) + Math.abs(temp4) >= 1) {
                    throw new RuntimeException("Метод расходится");
                }
            }
        }

        double xn = G[1], yn = G[3];
```

```
        double x_prev, y_prev;
        do {
            x_prev = xn;
            y_prev = yn;
            xn = functions[4].apply(xn, yn);
            yn = functions[5].apply(xn, yn);
            iters++;
        } while (Math.abs(xn - x_prev) > ACCURACY && Math.abs(yn - y_prev) > ACCURACY);
        Object[] X = new Object[] {xn, Math.abs(xn - x_prev)};
        Object[] Y= new Object[] {yn, Math.abs(yn - y_prev)};
        Object[] iterats = new Object[] {iters, null};
        return new Object[][] {X, Y, iterats};
    }
}
```

## Вывод

В ходе выполнения лабораторной работы были изучены численные методы решения нелинейных уравнений и систем нелинейных уравнений с использованием Java. В результате работы были найдены корни заданных уравнений и систем с использованием различных численных методов, а также были построены графики функций для полного представления исследуемых интервалов.