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 12

Выполнила:

Студентка группы P3217

Русакова Е.Д.

Преподаватель:

Малышева Т.А.

Санкт-Петербург

2024

Оглавление

Цель работы: 3
Задание: 3
Задание для варианта 12: 4
Описание методов для решения нелинейных уравнений: 5
Метод половинного деления 5
Метод Ньютона 5
Метод секущих 5
Метод простой итерации 6
Описание методов для решения систем нелинейных уравнений: 6
Метод Ньютона 6
Метод простой итерации 7
Вычислительная часть: 8
Решение нелинейного уравнения: 8
Крайний правый корень - Метод секущих: 9
Крайний левый корень – Метод простой итерации 9
Центральный корень - Метод половинного деления 9
Решение системы нелинейных уравнений: 10
Программная реализация: 11
Описание разработанной программы: 11
Исходный код программы: 11
HalfDivisionMethod.java - Метод половинного деления для решения нелинейных уравнений 11
NewtonMethod.java – Метод Ньютона для решения нелинейных уравнений 12
SimpleIterationMethod.java – Метод простой итерации для решения нелинейных уравнений 13
SystemNewtonMethod.java – Метод Ньютона для решения системы нелинейных уравнений .14
Примеры работы программы: 16
Пример 1 16
Пример 2 17
Пример 3 20
Пример 4 20
Вывод: 21

Цель работы:

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

Задание:

1 Вычислительная реализация задачи:

Состоит из двух частей и отражается ТОЛЬКО в отчете.

1 часть. Решение нелинейного уравнения

Задание:

1. Отделить корни заданного нелинейного уравнения графически (вид уравнения представлен в табл. 6)
2. Определить интервалы изоляции корней.
3. Уточнить корни нелинейного уравнения (см. табл. 6) с точностью $\varepsilon=10^{-2}$.
4. Используемые методы для уточнения каждого из 3-х корней многочлена представлены в таблице 7.
5. Вычисления оформить в виде таблиц (1-5), в зависимости от заданного метода. Для всех значений в таблице удержать 3 знака после запятой.
   - 5.1 Для метода половинного деления заполнить таблицу 1.
   - 5.2 Для метода хорд заполнить таблицу 2.
   - 5.3 Для метода Ньютона заполнить таблицу 3.
   - 5.4 Для метода секущих заполнить таблицу 4.
   - 5.5 Для метода простой итерации заполнить таблицу 5. Проверить условие сходимости метода на выбранном интервале.
6. Заполненные таблицы отобразить в отчете.

2 часть. Решение системы нелинейных уравнений

Задание:

1. Отделить корни заданной системы нелинейных уравнений графически (вид системы представлен в табл. 8).
2. Используя указанный метод, решить систему нелинейных уравнений с точностью до 0,01.
3. Для метода простой итерации проверить условие сходимости метода.
4. Подробные вычисления привести в отчете.

2 Программная реализация задачи:

Для нелинейных уравнений:

1. Все численные методы (см. табл. 9) должны быть реализованы в виде отдельных подпрограмм/методов/классов.
2. Пользователь выбирает уравнение, корень/корни которого требуется вычислить (3-5 функций, в том числе и трансцендентные), из тех, которые предлагает программа.
3. Предусмотреть ввод исходных данных (границы интервала/начальное приближение к корню и погрешность вычисления) из файла или с клавиатуры по выбору конечного пользователя.
4. Выполнить верификацию исходных данных. Необходимо анализировать наличие корня на введенном интервале. Если на интервале несколько корней или они отсутствуют – выдавать соответствующее сообщение. Программа должна реагировать на некорректные введенные данные.
5. Для методов, требующих начальное приближение к корню (методы Ньютона, секущих, хорд с фиксированным концом, простой итерации), выбор начального приближения x_0(a или b) вычислять в программе.
6. Для метода простой итерации проверять достаточное условие сходимости метода на введенном интервале.
7. Предусмотреть вывод результатов (найденный корень уравнения, значение функции в корне, число итераций) в файл или на экран по выбору конечного пользователя.
8. Организовать вывод графика функции, график должен полностью отображать весь исследуемый интервал (с запасом).

Для систем нелинейных уравнений:

1. Пользователь выбирает предлагаемые программой системы двух нелинейных уравнений (2-3 системы).
2. Организовать вывод графика функций.
3. Начальные приближения ввести с клавиатуры.
4. Для метода простой итерации проверить достаточное условие сходимости.
5. Организовать вывод вектора неизвестных: $x_1,\ x_2$.
6. Организовать вывод количества итераций, за которое было найдено решение.
7. Организовать вывод вектора погрешностей: $|x_i^{(k)} - x_i^{(k-1)}|$
8. Проверить правильность решения системы нелинейных уравнений.

Задание для варианта 12:

Для вычислительной части:

Нелинейное уравнение: $f(x) = x^3 - 4{,}5x^2 - 9{,}21x - 0{,}383$, методы: секущих, простой итерации и половинного деления

Система нелинейных уравнений: $\begin{cases} x + \sin y = -0{,}4 \\ 2y - \cos(x+1) = 0 \end{cases}$, метод простой итерации

Для программной части:

Методы: половинного деления, Ньютона и простой итерации для нелинейных уравнений

Метод простой итерации для системы нелинейных уравнений

Описание методов для решения нелинейных уравнений:

Метод половинного деления

Идея метода: делим интервал пополам и выбираем ту половину, на концах которой функция имеет разные знаки, то есть в ней лежит корень.

Рабочие формулы:

$$x_i = \frac{a_i + b_i}{2}$$

$$\begin{bmatrix} \begin{cases} a_{i+1} = a_i \\ b_{i+1} = x_i \end{cases}, \text{если } f(a_i) * f(x_i) < 0 \\ \begin{cases} a_{i+1} = x_i \\ b_{i+1} = b_i \end{cases}, \text{если } f(x_i) * f(b_i) < 0 \end{bmatrix}$$

Приближенным значением корня будет $x^* = \frac{a_n + b_n}{2}$ или $x^* = a_n$ или $x^* = b_n$

Критерий окончания итерационного процесса:

$|x_{k+1} - x_k| \leq \varepsilon$ или $|f(x_k)| \leq \varepsilon$ или $|a_k - b_k| \leq \varepsilon$

Метод Ньютона

Идея метода: функция заменяется касательной и в качестве приближенного значения корня принимается точка пересечения касательной с осью абсцисс.

Рабочие формулы:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Выбор начального приближения: чтобы добиться быстрой сходимости

$$x_0 = \begin{cases} a_0, \text{если } f(a_0) * f''(a_0) > 0 \\ b_0, \text{если } f(b_0) * f''(b_0) > 0 \end{cases}$$

Приближенным значением корня будет $x^* = x_n$

Критерий окончания итерационного процесса:

$|x_{k+1} - x_k| \leq \varepsilon$ или $|f(x_k)| \leq \varepsilon$ или $\left|\frac{f(x_n)}{f'(x_n)}\right| \leq \varepsilon$

Метод секущих

Идея метода: упрощение метода Ньютона заменой производной на разностное приближение:

$$f'(x_{i)} \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

Рабочие формулы:

$$x_{i+1} = x_i - \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} * f(x_i)$$

Выбор начального приближения: x_0 выбираем также как в методе Ньютона, x_1 выбираемся рядом самостоятельно

Приближенным значением корня будет $x^* = x_n$

Критерий окончания итерационного процесса:

$|x_{k+1} - x_k| \leq \varepsilon$ или $|f(x_{k)}| \leq \varepsilon$

Метод простой итерации

Идея метода: Приводим уравнение $f(x) = 0$ к эквивалентному виду $x = \varphi(x)$, выразив x из исходного уравнения.

Рабочие формулы:

$$x_{i+1} = \varphi(x_i)$$

Приближенным значением корня будет $x^* = x_n$

Критерий окончания итерационного процесса: $|x_{k+1} - x_k| \leq \varepsilon$

Условие сходимости метода:

$$|\varphi'(x)| \leq 1$$

Описание методов для решения систем нелинейных уравнений:

Метод Ньютона

Идея метода: Раскладываем функции в ряд Тейлора в окрестности некоторой фиксированной точки, ограничившись только линейными членами. Задача состоит в поиске приращений к приближенным решениям для получения нового приближения.

Критерий окончания итерационного процесса: $|x_{k+1} - x_k| \leq \varepsilon$ и $|y_{k+1} - y_k| \leq \varepsilon$

Рабочие формулы для решения систем из двух нелинейных уравнений:

$$\begin{cases} f_1(x, y) = 0 \\ f_2(x, y) = 0 \end{cases}$$

Пусть x_i, y_i – текущее приближение

$$\begin{cases} \dfrac{\partial f_1(x_i, y_i)}{\partial x}\,\Delta x + \dfrac{\partial f_1(x_i, y_i)}{\partial y}\Delta y = -f_1(x_i, y_i) \\ \dfrac{\partial f_2(x_i, y_i)}{\partial x}\,\Delta x + \dfrac{\partial f_2(x_i, y_i)}{\partial y}\Delta y = -f_2(x_i, y_i) \end{cases}$$

Пусть $\frac{\partial f_1(x_i, y_i)}{\partial x} = a_i$, $\frac{\partial f_1(x_i, y_i)}{\partial y} = b_i$, $\frac{\partial f_2(x_i, y_i)}{\partial x} = c_i$, $\frac{\partial f_2(x_i, y_i)}{\partial y} = d_i$

Тогда система будет выглядеть:

$$\begin{cases} a_i\,\Delta x + b_i \Delta y = -f_1(x_i, y_i) \\ c_i\,\Delta x + d_i \Delta y = -f_2(x_i, y_i) \end{cases}$$

На этом шаге итерации известны численные значения a_i , $b_i, c_i, d_i, f_1(x_i, y_i), f_2(x_i, y_i)$, поэтому можем выразить через них Δx и Δy:

$$\begin{cases} \Delta x = \dfrac{(-b_i \Delta y - f_1)}{a_i} \\ c_i * \dfrac{(-b_i \Delta y - f_1)}{a_i} + d_i \Delta y = -f_2 \end{cases}$$

$$c_i(-b_i \Delta y - f_1) + a_i d_i \Delta y = -a_i f_2$$

$$\Delta y(-c_i b_i + a_i d_i) = -a_i f_2 + c_i f_1$$

$$\begin{cases} \Delta y = \dfrac{-a_i f_2 + c_i f_1}{a_i d_i - c_i b_i} \\ \Delta x = \dfrac{(-b_i \Delta y - f_1)}{a_i} \end{cases}$$

Тогда следующее приближение равно:

$$\begin{cases} x_{i+1} = x_i + \Delta x \\ y_{i+1} = y_i + \Delta y \end{cases}$$

Метод простой итерации

Идея метода: Приводим систему уравнений к эквивалентному виду $X = \varphi(X)$, выразив x_i из каждого уравнения исходной системы.

Рабочие формулы:

$$\begin{cases} x_1^{k+1} = \varphi_1\left(x_{1_k}, x_2^k, \dots, x_n^k\right) \\ x_2^{k+1} = \varphi_2\left(x_{1_k}, x_2^k, \dots, x_n^k\right) \\ \dots \\ x_n^{k+1} = \varphi_n\left(x_{1_k}, x_2^k, \dots, x_n^k\right) \end{cases}$$

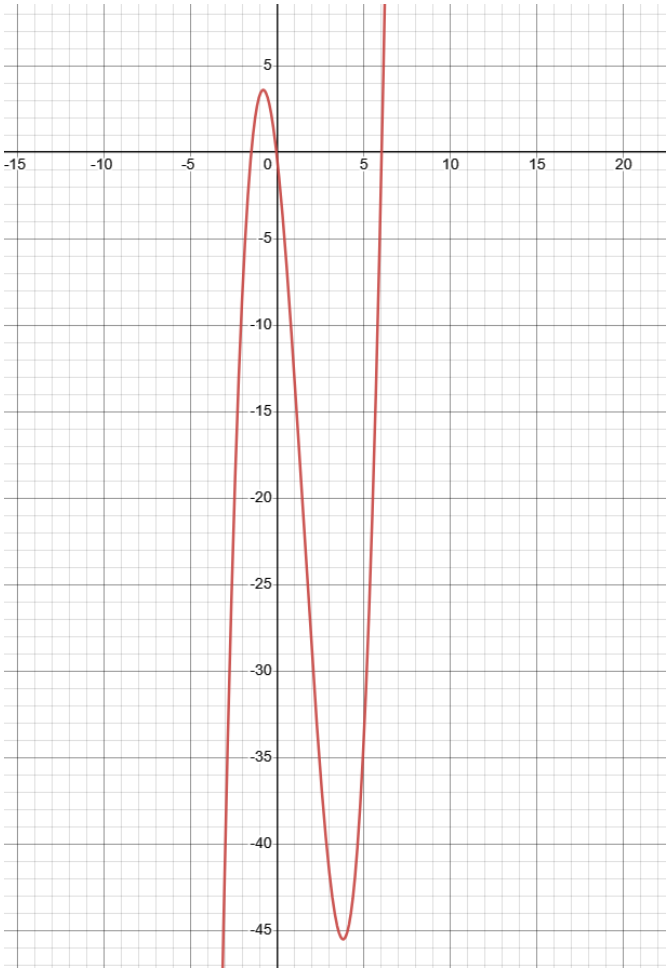
Критерий окончания итерационного процесса: $\max\limits_{1 \le i \le n} \left| x_i^{k+1} - x_i^k \right| \le \varepsilon$

Условие сходимости метода:

$$\max_{[x \in G]} \max_{[i]} \sum_{j=1}^{n} \left| \frac{\partial \varphi_i(X)}{\partial x_j} \right| \le q < 1$$

Вычислительная часть:

Решение нелинейного уравнения:



Нелинейное уравнение:

$$f(x) = x^3 - 4{,}5x^2 - 9{,}21x - 0{,}383$$

Интервалы изоляции корней:

[-2; -1], [-1; 1], [6; 7]

$$f'(x) = 3x^2 - 9x - 9{,}21$$

$$f''(x) = 6x - 9$$

Крайний правый корень - Метод секущих:

Границы интервала: [6; 7]

Выбираем начальное приближение:

$$f(6) * f''(6) = -1{,}643 * 27 < 0$$

$$f(7) * f''(7) = 57{,}647 * 33 > 0$$

Начальное приближение $x_0 = 7$, $x_1 = 6{,}5$

№ итерации (k)	x_{k-1}	x_k	x_{k+1}	$f(x_{k+1})$	$\lvert x_{k+1} - x_k \rvert$
0		7	6,5	24,252	0,5
1	7	6,5	6,137	4,749	0,363
2	6,5	6,137	6,049	0,584	0,088
3	6,137	6,049	6,037	0,033	0,012
4	6,049	6,037	6,036	-0,013	0,001

Корень $x = x_5 = 6{,}036$

Крайний левый корень – Метод простой итерации

Границы интервала: [-2; -1]

Составляем функцию:

$$f'(-2) = 20{,}79$$

$$f'(-1) = 2{,}79$$

На данном интервале $f'(x) > 0$

$$\lambda = -\frac{1}{\max|f'(x)|} = -\frac{1}{20{,}79} = -0{,}048$$

$$\varphi(x) = x + \lambda f(x) = -0{,}048x^3 + 0{,}216x^2 + 1{,}442x + 0{,}018$$

№ итерации (k)	x_k	x_{k+1}	$\varphi(x)$	$f(x_{k+1})$	$\lvert x_{k+1} - x_k \rvert$
0		-2	-1,61778	-7,963	
1	-2	-1,618	-1,54611	-1,498	0,382
2	-1,618	-1,546	-1,51744	-0,595	0,072
3	-1,546	-1,517	-1,5046	-0,258	0,029
4	-1,517	-1,505	-1,49908	-0,123	0,012
5	-1,505	-1,499	-1,49627	-0,057	0,006

Корень $x = x_5 = -1{,}499$

Центральный корень - Метод половинного деления

Границы интервала: [-1; 1]

№ итерации	a	b	x	f(a)	f(b)	f(x)	|a-b|
0	-1	1	0	3,327	-13,093	-0,383	2
1	-1	0	-0,5	3,327	-0,383	2,972	1
2	-0,5	0	-0,25	2,972	-0,383	1,623	0,5
3	-0,25	0	-0,125	1,623	-0,383	0,696	0,25

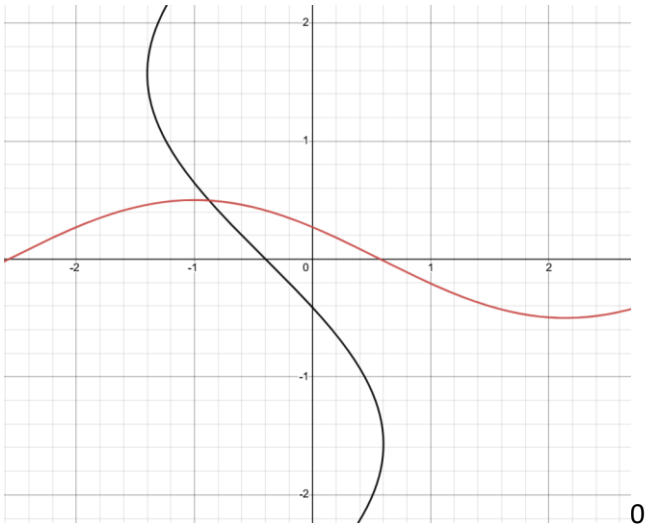
4	-0,125	0	-0,063	0,696	-0,383	0,179	0,125
5	-0,063	0	-0,032	0,179	-0,383	-0,093	0,063
6	-0,063	-0,032	-0,048	0,179	-0,093	0,049	0,031
7	-0,048	-0,032	-0,04	0,049	-0,093	-0,022	0,016
8	-0,048	-0,04	-0,044	0,049	-0,022	0,013	0,008

Корень $x = x_8 = -0{,}044$

Решение системы нелинейных уравнений:

$$\begin{cases} x + \sin y = -0{,}4 \\ 2y - \cos(x+1) = 0 \end{cases}$$

Метод простой итерации



0

$$\begin{cases} x = -0{,}4 - \sin y = \varphi_1 \\ y = \cos\dfrac{(x+1)}{2} = \varphi_2 \end{cases}$$

Проверяем условие сходимости на области G: $\begin{cases} -1 \le x \le 0 \\ 0 \le y \le 1 \end{cases}$

$$\frac{\partial\varphi_1}{\partial x} = 0\,;\ \frac{\partial\varphi_1}{\partial y} = -\cos y\,;\ \frac{\partial\varphi_2}{\partial x} = -\frac{\sin(x+1)}{2}\,;\ \frac{\partial\varphi_2}{\partial y} = 0$$

$$\left.\begin{array}{c} \left|\dfrac{\partial\varphi_1}{\partial x}\right| + \left|\dfrac{\partial\varphi_1}{\partial y}\right| = 0 + |-\cos y| < 1 \\ \left|\dfrac{\partial\varphi_2}{\partial x}\right| + \left|\dfrac{\partial\varphi_2}{\partial y}\right| = 0 + \left|-\dfrac{\sin(x+1)}{2}\right| < 1 \end{array}\right\} =>$$

Сходится на данной области.

Выберем начальное приближение: $x^0 = -1; y^0 = 0{,}5$

	x_k	y_k	$\lvert x_{k+1} - x_k\rvert$	$\lvert y_{k+1} - y_k\rvert$
0	-1	0,5		
1	-0,879	0,5	0,121	0
2	-0,879	0,496	0	0,004

Решение: $x^2 = -0{,}879; y^2 = 0{,}496$

Программная реализация:

Описание разработанной программы:

Исходный код программы:

Полный код программы выложен на Github и доступен по ссылке [lenapochemy/comp-math-lab2: вычмат лаба 2 (github.com)]

Далее приведен код классов, которые отвечают за решение

HalfDivisionMethod.java - Метод половинного деления для решения нелинейных уравнений

```
package methods;

import java.util.function.DoubleFunction;

public class HalfDivisionMethod extends AbstractMethod {

    public HalfDivisionMethod(DoubleFunction<Double> function, double eps,
double a, double b){
        super(function, eps, a, b);
    }
    private double x, f_a, f_b, f_x;
    @Override
    public void solve(){
        iterationNumber = 0;
        boolean flag =true;
        while (flag){
            x = (a + b) / 2;
            f_a = function.apply(a);
            f_b = function.apply(b);
            f_x = function.apply(x);
            flag = !checkEndConditional();
            if(f_a * f_x < 0) {
                b = x;
            } else if(f_x * f_b < 0){
                a = x;
            }
            writeIteration( "Итерация " + iterationNumber + "\nНовое
приближение: a = " + a + " b = " + b + "\n-------------------------\n");
            iterationNumber++;
        }

        if(Math.abs(f_a) < Math.abs(f_b)){
            x = a;
        } else x = b;

        writeResult( "Найденный корень: " + x + "\nЗначение функции в корне:
" + function.apply(x) +
```

```
                "\nЧисло итераций: " + iterationNumber);

    }


    //вернет true, если условие окончания выполняется и это последняя
итерация
    @Override
    public boolean checkEndConditional(){
        return Math.abs(a - b) < eps && Math.abs(function.apply(x)) < eps ;
    }

}
```

NewtonMethod.java – Метод Ньютона для решения нелинейных уравнений

```
package methods;

import java.util.function.DoubleFunction;

public class NewtonMethod extends AbstractMethod {

    public NewtonMethod(DoubleFunction<Double> function, double eps, double
a, double b){
        super(function, eps, a, b);
    }
    @Override
    public void solve(){
        //выбор начальногоо приближения
        x_i = chooseFirstApproximation();
//        x_i = b;
        writeIteration("Первое приближение: " + x_i + "\n----------------------
------\n");

        iterationNumber = 0;
        x_i_next = x_i - ( function.apply(x_i) / derive_function.apply(x_i)
);
        do {
            x_i = x_i_next;
            x_i_next = x_i - ( function.apply(x_i) /
derive_function.apply(x_i) );
            writeIteration( "Итерация " + iterationNumber + "\nНовое
приближение: " + x_i_next + "\n--------------------------\n");
            iterationNumber++;
        } while (!checkEndConditional());

        writeResult( "Найденный корень: " + x_i_next + "\nЗначение функции в
корне: " + function.apply(x_i_next) +
                "\nЧисло итераций: " + iterationNumber);
    }

    //вернет true, если условие окончания выполняется и это последняя
итерация
    @Override
    public boolean checkEndConditional(){
        return Math.abs(x_i_next - x_i) < eps;
    }
}
```

SimpleIterationMethod.java – Метод простой итерации для решения нелинейных уравнений

```
package methods;

import java.util.function.DoubleFunction;

public class SimpleIterationMethod extends AbstractMethod {
    public SimpleIterationMethod(DoubleFunction<Double> f, double eps, double
a, double b){
        super(f, eps, a, b);
    }
    private double lambda = 0;
    @Override
    public void solve(){
        double f1_a, f1_b;
        //ищем функцию
        f1_a = derive_function.apply(a);
        f1_b = derive_function.apply(b);
        if(f1_a > 0 && f1_b > 0) {
            lambda = -1 / Math.max(f1_a, f1_b);
        } else {
            lambda = 1 / Math.min(f1_a, f1_b);
        }
        DoubleFunction<Double> fi_function = x -> (x + lambda *
function.apply(x));
        DoubleFunction<Double> derive_fi = derive(fi_function);

        //проверка условия сходимости
        writeIteration("\n-------------------------\n" + "Значение
производной на границах: fi'(a) = " + derive_fi.apply(a) + " fi'(b) = " +
derive_fi.apply(b));
        if(Math.abs(derive_fi.apply(a)) > 1 || Math.abs(derive_fi.apply(b)) >
1){
            writeIteration("Достаточное условие сходимости метода простой
итерации на данном интервале не выполнено" + "\n-------------------------
\n");
            System.exit(0);
        } else writeIteration("Достаточное условие сходимости выполнено" +
"\n-------------------------\n");

        // первое приближение
        x_i = chooseFirstApproximation();
        writeIteration("Первое приближение: " + x_i + "\n-------------------
------\n");

        x_i_next = fi_function.apply(x_i);
        iterationNumber = 1;
        while (!checkEndConditional()) {
            x_i = x_i_next;
            x_i_next = fi_function.apply(x_i);
            writeIteration( "Итерация " + iterationNumber + "\nНовое
приближение: " + x_i_next + "\n-------------------------\n");
            iterationNumber++;
        }

        writeResult( "Найденный корень: " + x_i_next + "\nЗначение функции в
корне: " + function.apply(x_i_next) +
                "\nЧисло итераций: " + iterationNumber);
    }
```

```
    //вернет true, если условие окончания выполняется и это последняя
итерация
    @Override
    public boolean checkEndConditional(){
        return Math.abs(x_i_next - x_i) < eps &&
Math.abs(function.apply(x_i_next))<eps;
    }
}
```

SystemNewtonMethod.java – Метод Ньютона для решения системы нелинейных уравнений

```
package methods;

import java.io.FileWriter;
import java.io.IOException;
import java.util.function.BiFunction;


public class SystemNewtonMethod {

    public SystemNewtonMethod(BiFunction<Double, Double, Double> func1,
BiFunction<Double, Double, Double> func2,
                              double eps, double x_0, double y_0){
        this.func1 = func1;
        this.func2 = func2;
        this.eps = eps;
        this.x_0 = x_0;
        this.y_0 = y_0;
    }

    private final BiFunction<Double, Double, Double> func1, func2;
    private boolean solveMode, outputMode;

    private final double eps, x_0, y_0;
    private double dx, dy, x_k, y_k, x_k_next, y_k_next;
    private FileWriter file;
    public void setFile(FileWriter file){
        this.file = file;
    }

    public void setSolveMode(boolean solveMode){
        this.solveMode = solveMode;
    }
    public void setOutputMode(boolean outputMode){
        this.outputMode = outputMode;
    }

    public BiFunction<Double, Double, Double>
partialDeriveX(BiFunction<Double, Double, Double> f){
        double dx = 0.0001;
        return (x, y) -> ((f.apply(x + dx, y)) - (f.apply(x, y)) ) / dx;
    }

    public BiFunction<Double, Double, Double>
partialDeriveY(BiFunction<Double, Double, Double> f){
        double dy = 0.0001;
        return (x, y) -> ((f.apply(x, y + dy)) - (f.apply(x, y)) ) / dy;
    }
```

```
    public void solve(){
        BiFunction<Double, Double, Double> df1_dx = partialDeriveX(func1);
        BiFunction<Double, Double, Double> df2_dx = partialDeriveX(func2);
        BiFunction<Double, Double, Double> df1_dy = partialDeriveY(func1);
        BiFunction<Double, Double, Double> df2_dy = partialDeriveY(func2);

        x_k = x_0;
        y_k = y_0;
        writeIteration("Первое приближение: x = " + x_k + "  y = " + y_k +
"\n--------------------------\n");

        boolean flag = true;
        int i = 0;
        while (flag) {
            i++;
            double a = df1_dx.apply(x_k, y_k);
            double b = df1_dy.apply(x_k, y_k);
            double c = df2_dx.apply(x_k, y_k);
            double d = df2_dy.apply(x_k, y_k);
            double f1 = func1.apply(x_k, y_k);
            double f2 = func2.apply(x_k, y_k);

            dy = ( 0 - a * f2 + c * f1) / (d * a - c * b);
            dx = (0 - f1 - b * dy) / a;

            x_k_next = x_k + dx;
            y_k_next = y_k + dy;

            writeIteration( "Итерация " + i + "\nНовое приближение: x = " +
x_k_next + "  y = " + y_k_next + "\n--------------------------\n");

            if(checkEndCondition()) {
                flag = false;
                writeResult( "x = " + x_k_next + " y = " + y_k_next +
                        "\nЗначение функций в корне " + func1.apply(x_k_next,
y_k_next) + " " + func2.apply(x_k_next, y_k_next) +
                        "\nКоличество итераций: " + i +
                        "\nВектор погрешностей: " + Math.abs(x_k_next - x_k)
+" " + Math.abs(y_k_next - y_k));
            }

            x_k = x_k_next;
            y_k = y_k_next;
        }
    }


    private boolean checkEndCondition(){
        return (Math.abs(x_k_next - x_k) < eps && Math.abs(y_k_next - y_k) <
eps);
    }

    public void writeResult(String string){
        if(outputMode){
            System.out.println(string);
        }  else {
            try {
                file.write(string);
```

```
                file.close();
            } catch (IOException e){
                System.out.println("Проблемы с файлом");
            }
        }
    }

    public void writeIteration(String string){
        if(solveMode){
            if(outputMode){
                System.out.println(string);
            } else {
                try {
                    file.write(string);
                } catch (IOException e){
                    System.out.println("Проблемы с файлом");
                }
            }
        }
    }
}
```

Примеры работы программы:

Пример 1

```
Вы хотите решить нелинейное уравнение или систему нелинейных уравнений? (e/s)

Ответ должен быть "e" или "s"
Вы хотите решить нелинейное уравнение или систему нелинейных уравнений? (e/s)
e
Выберите функцию для решения:
    1. x³-x+4=0
    2. x³-4.5x²-9.21x-0.383=0
    3. x*sin(x)+2x-3=0
    4. x³+2x²-4x=0
    5. ln(x²)-x+10=0
2
Вы хотите вводить данные с клавиатуры или из файла? (k/f)
k
Введите значение точности [0.000001; 1]: 0.01
Введите левой границы интервала : -2
Введите правой границы интервала : -1
Выберите метод решения уравнения: Метод половинного деления(d), Метод Ньютона(n), Метод простой итерации(i) n
Нужно выводить результат каждой итерации решения? (y/n)y
Результаты вывести на экран или в записать в файл? (s/f)s
```

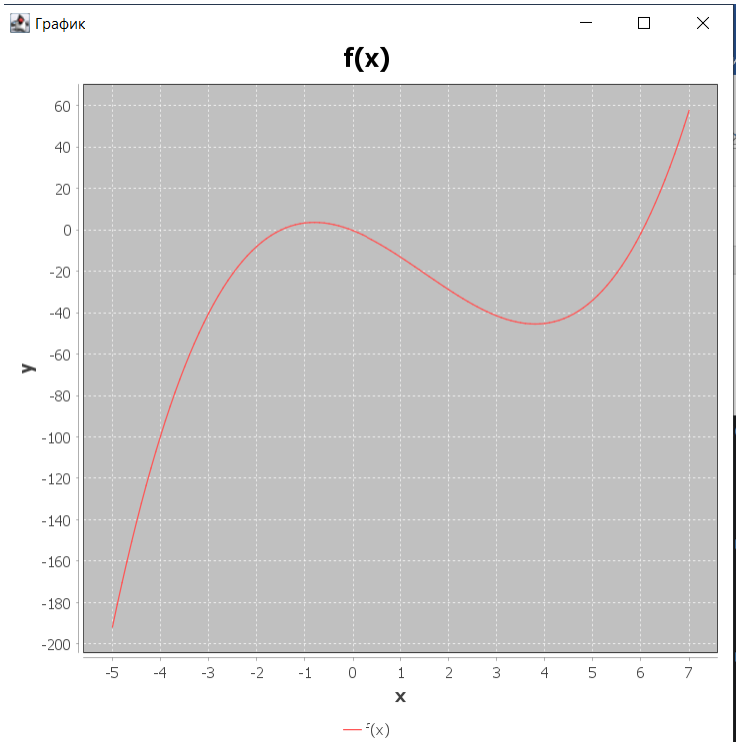
```
Первое приближение: -2.0
-------------------------

Итерация 0
Новое приближение: -1.5044145313168644
-------------------------

Итерация 1
Новое приближение: -1.4938995647044317
-------------------------

Итерация 2
Новое приближение: -1.4938093580512986
-------------------------

Найденный корень: -1.4938093580512986
Значение функции в корне: 7.934489509864306E-9
Число итераций: 3
```



Пример 2

```
Вы хотите решить нелинейное уравнение или систему нелинейных уравнений? (e/s)
e
Выберите функцию для решения:
    1. x³-x+4=0
    2. x³-4.5x²-9.21x-0.383=0
    3. x*sin(x)+2x-3=0
    4. x³+2x²-4x=0
    5. ln(x²)-x+10=0
5
Вы хотите вводить данные с клавиатуры или из файла? (k/f)
f
Введите путь к файлу:
C:\Users\Elena\IdeaProjects\compMath\lab2\src\files\file1
Введите значение точности [0.000001; 1]: 0.01
Введите левой границы интервала : -3
Введите правой границы интервала : 0
Выберите метод решения уравнения: Метод половинного деления(d), Метод Ньютона(n), Метод простой итерации(i) d
Нужно выводить результат каждой итерации решения? (y/n)y
Результаты вывести на экран или в записать в файл? (s/f)Введите путь к файлу:
C:\Users\Elena\IdeaProjects\compMath\lab2\src\files\result1
```

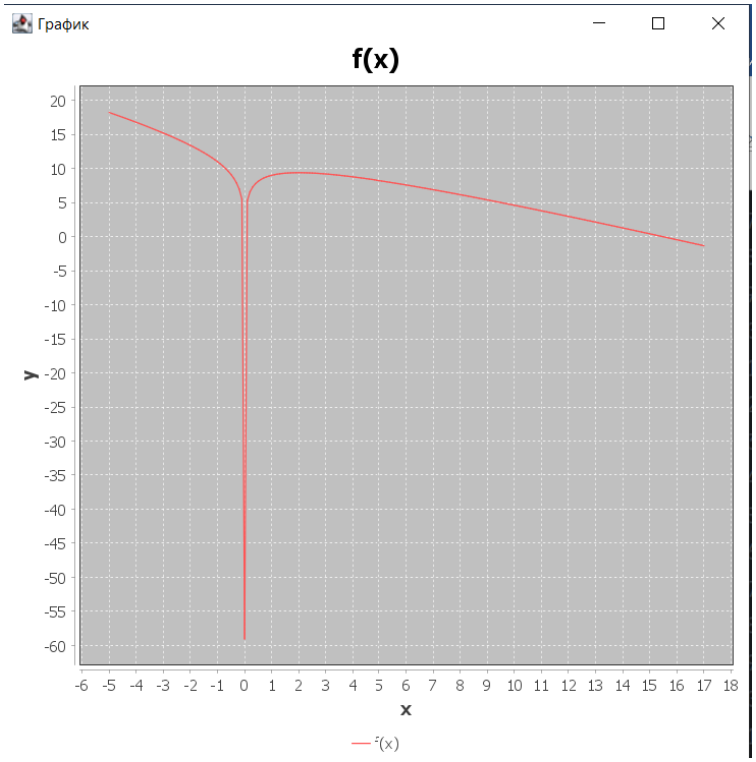
file1 × result1

```
1  0.01
2  -3
3  0
4  d
5  y
6  f
7  C:\Users\Elena\IdeaProjects\compMath\lab2\src\files\result1
```

file1 result1

```
Итерация 0
Новое приближение: a = -1.5 b = 0.0
--------------------------
Итерация 1
Новое приближение: a = -0.75 b = 0.0
--------------------------
Итерация 2
Новое приближение: a = -0.375 b = 0.0
--------------------------
Итерация 3
Новое приближение: a = -0.1875 b = 0.0
--------------------------
Итерация 4
Новое приближение: a = -0.09375 b = 0.0
--------------------------
Итерация 5
Новое приближение: a = -0.046875 b = 0.0
--------------------------
Итерация 6
Новое приближение: a = -0.0234375 b = 0.0
--------------------------
Итерация 7
Новое приближение: a = -0.01171875 b = 0.0
--------------------------
Итерация 8
Новое приближение: a = -0.01171875 b = -0.005859375
--------------------------
```

```
--------------------------
Итерация 9
Новое приближение: a = -0.0087890625 b = -0.005859375
--------------------------
Итерация 10
Новое приближение: a = -0.00732421875 b = -0.005859375
--------------------------
Итерация 11
Новое приближение: a = -0.00732421875 b = -0.006591796875
--------------------------
Итерация 12
Новое приближение: a = -0.0069580078125 b = -0.006591796875
--------------------------
Итерация 13
Новое приближение: a = -0.00677490234375 b = -0.006591796875
--------------------------
Итерация 14
Новое приближение: a = -0.00677490234375 b = -0.006683349609375
--------------------------
Найденный корень: -0.00677490234375
Значение функции в корне: 0.017714249289950246
Число итераций: 15
```



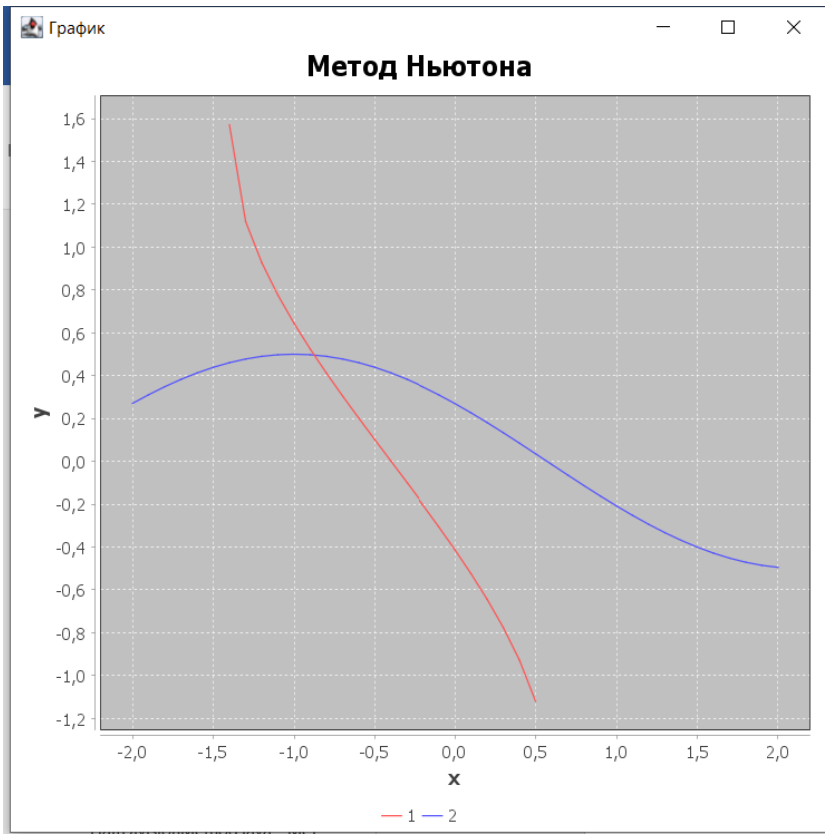
Пример 3

```
Вы хотите решить нелинейное уравнение или систему нелинейных уравнений? (e/s)
e
Выберите функцию для решения:
    1. x³-x+4=0
    2. x³-4.5x²-9.21x-0.383=0
    3. x*sin(x)+2x-3=0
    4. x³+2x²-4x=0
    5. ln(x²)-x+10=0
3
Вы хотите вводить данные с клавиатуры или из файла? (k/f)
k
Введите значение точности [0.000001; 1]: 0.001
Введите левой границы интервала : -4
Введите правой границы интервала : -1
На данном интервале несколько корней или они отсутствуют, выберите другой интервал
Введите левой границы интервала : 0
Введите правой границы интервала : 4
Выберите метод решения уравнения: Метод половинного деления(d), Метод Ньютона(n), Метод простой итерации(i) i
Нужно выводить результат каждой итерации решения? (y/n)y
Результаты вывести на экран или в записать в файл? (s/f)s

--------------------------
Значение производной на границах: fi'(a) = -0.45855258819393896 fi'(b) = 1.9999999999953388
Достаточное условие сходимости метода простой итерации на данном интервале не выполнено
--------------------------
```

Пример 4

```
Вы хотите решить нелинейное уравнение или систему нелинейных уравнений? (e/s)
s
Выберите функцию для решения:
    1. { x²+y²-4=0
         -3x²+y=0
    2. { x+sin(y)+0.4=0
          2y-cos(x+1)=0
    3. { x²+x+y=0
          y+x³-10=0
2
Вы хотите вводить данные с клавиатуры или из файла? (k/f)
k
Введите значение точности [0.000001; 1]: 0.0001
Введите начальное приближение для переменной x : 100
Введите начальное приближение для переменной y : 10,1
Значение начальное приближение для переменной y должно быть числом
Введите начальное приближение для переменной y : 100
Нужно выводить результат каждой итерации решения? (y/n)n
Результаты вывести на экран или в записать в файл? (s/f)s
x = -0.8760559488196266 y = 0.4961643820791022
Значение функций в корне -4.805211784031371E-12 -2.0220491947497976E-11
Количество итераций: 14
Вектор погрешностей: 4.092100243102692E-7 2.0145722956499412E-7
```



Вывод:

При выполнение лабораторной работы я познакомилась с различными методами решения нелинейных уравнений и систем нелинейных уравнений и выполнила программную реализацию некоторых из них.