Министерство образования и науки РФ

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

факультет программной инженерии и компьютерной техники

**ЛАБОРАТОРНАЯ РАБОТА №2**

‘ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА’

Вариант №24

*Студент:*

Хоанг Ван Куан

Группа Р3266

*Преподаватель:*

Машина Екатерина Александровна

Санкт-Петербург, 2024

1. **Цель работы**

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

1. **Порядок выполнения работы**

* 1 часть: Решение нелинейного уравнения

Задание:

1. Отделить корни заданного нелинейного уравнения графически

2. Определить интервалы изоляции корней.

3. Уточнить корни нелинейного уравнения с точностью ε =

4. Используемые методы для уточнения каждого из 3-х корней многочлена

представлены в таблице 7.

Крайний правый корень: метод хорд

Крайний левый корень: метод Ньютона

Центральный корень: метод простой итерации

5. Вычисления оформить в виде таблиц (1-5), в зависимости от заданного метода. Для всех значений в таблице удержать 3 знака после запятой.

Для метода хорд заполнить таблицу 2.

Для метода Ньютона заполнить таблицу 3.

Для метода простой итерации заполнить таблицу 5. Проверить условие сходимости метода на выбранном интервале.

* 2 часть: Решение системы нелинейных уравнений

Задание:

1. Отделить корни заданной системы нелинейных уравнений графически

2. Используя метод Ньютона, решить систему нелинейных уравнений с точностью до 0,01.

* 3 часть: Программная реализация

Для нелинейных уравнений

- Метод хорд

- Метод секущих

- Метод простой итерации

Для систем нелнейных уравнений

- Метод простой интерации

1. **Рабочие формулы**

Рабочая формула метода хорд:

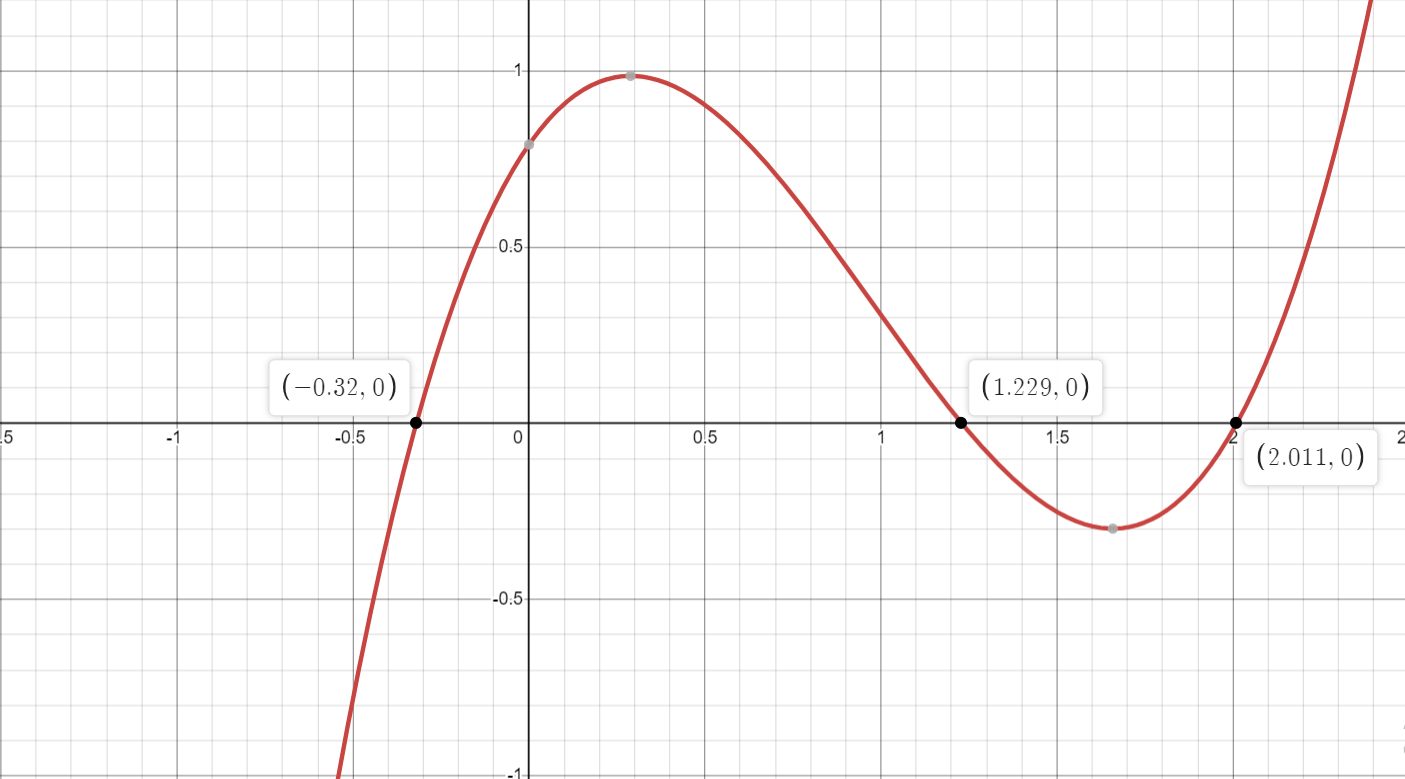
Рабочая формула метода Ньютона:

Рабочая формула метода простой интерации:

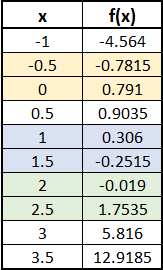
1. **Вычислительная часть 1**

1 - Отделить корни заданного нелинейного уравнения графически

Функция :



2 - Определить интервалы изоляции корней

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

Знаем, что если непрерывная функция f(x) на концах отрезка

[a; b] принимает значения разных знаков, т.е. f(a)·f(b)<0,

то на этом отрезке содержится хотя бы один корень уравнения

Тогда мы получаем 3 интервала изоляции корней уравнения:

(-0.5;0), (1;1.5) и (2;2.5)

3 - Уточнить корни нелинейного уравнения с точностью ε =

4 - Используемые методы для уточнения каждого из 3-х корней

* Метод хорд

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага |  |  |  |  |  |  |  |
| 0 | 2.00000 | 2.50000 | **2.00536** | -0.01900 | 1.75350 | -0.00951 | 0.00536 < ε |
|  |  |  |  |  |  |  |  |

* Метод Ньютона

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации |  |  |  |  |  |
| 0 | -0.50000 | -0.78150 | 5.10500 | -0.34691 | 0.15309 |
| 1 | -0.346910 | -0.10000 | 3.82203 | -0.32075 | 0.02616 |
| 2 | -0.32075 | -0.00269 | 3.61684 | **-0.32001** | 0.00074 < ε |

* Метод простой итерации

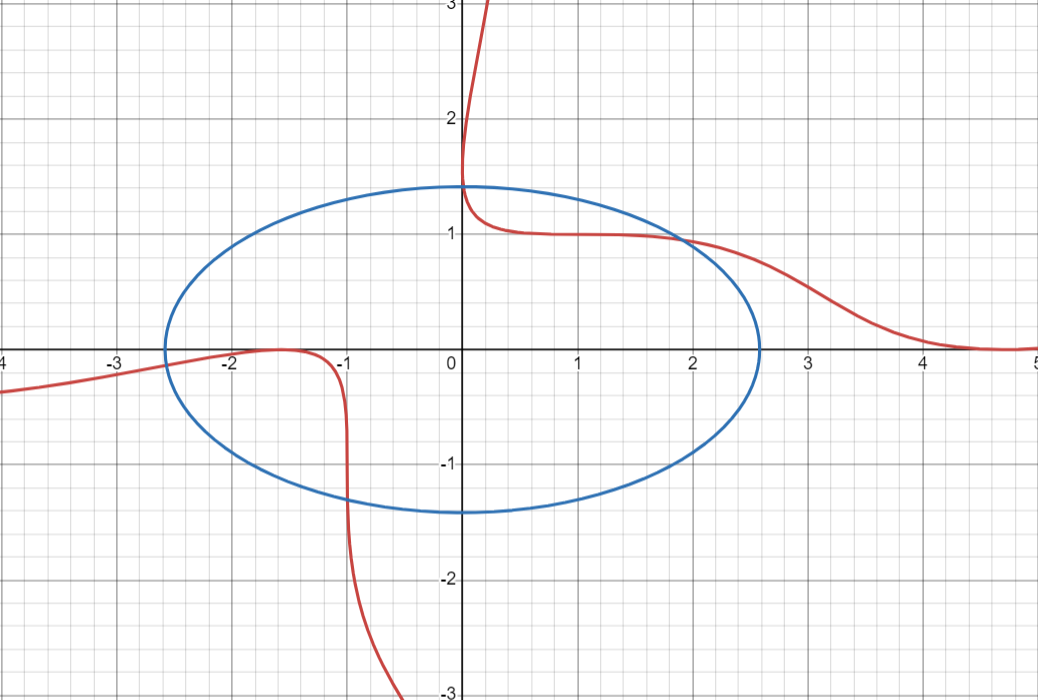
Преобразуем уравнение к виду

Условие сходимости выполняется

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации |  |  |  |  |  |
| 0 | 1.00000 | 1.21779 | 1.22785 | 0.01413 | 0.21779 |
| 1 | 1.21779 | 1.22785 | 1.22917 | 0.00185 | 0.01006 |
| 2 | 1.22785 | **1.22917** | 1.22935 | 0.00025 | 0.00132 < ε |

1. **Вычислительная часть 2**

1 - Отделить корни заданной системы нелинейных уравнений графически

**

2 – Решить систему нелинейных уравнений с точностью ε = по методу Ньютона

Построим матрицу Якоби

Выбираем

Аналогично, получается таблица

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № интерации |  |  |  |  |  |  |
| 0 | -3 | -1 | -0.0022 | 0.852 | -3.0022 | -0.1480 |
| 1 | -3.0022 | -0.1480 | 0.4008 | 0.0133 | -2.6014 | -0.1347 |
| 2 | -2.6014 | -0.1347 | 0.0312 | -0.0014 | -2.5702 | -0.1361 |
| 3 | **-2.5702** | **-0.1361** | 0.0002 < ε | -0.00005 < ε | - | - |

1. **Листинг программы**

Метод хорд

private void hordMethod() {  
 ans = new Answers();  
 double x = a - ((b-a) \* f.apply(a) )/(f.apply(b) - f.apply(a));  
 if(f.apply(a) \* f.apply(x) < 0) b = x;  
 if(f.apply(x) \* f.apply(b) < 0) a = x;  
  
 while(true) {  
 double x1 = a - ((b - a) \* f.apply(a)) / (f.apply(b) - f.apply(a));  
 if (Math.*abs*(x1 - x) <= epsilon || Math.*abs*(a - b) <= epsilon || Math.*abs*(f.apply(x1)) <= epsilon) {  
 ans.solution = x1;  
 ans.value = f.apply(x1);  
 ans.iteration = Math.*abs*(x1 - x);  
 break;  
 }  
 x = x1;  
 if (f.apply(a) \* f.apply(x) < 0) b = x;  
 if (f.apply(x) \* f.apply(b) < 0) a = x;  
 }  
}

Метод секущих

private void secanMethod() {  
 ans = new Answers();  
 double x = a;  
 if(f.apply(a) \* f.d(2, a) > 0) x = a;  
 if(f.apply(b) \* f.d(2, b) > 0) x = b;  
 double x1 = x - f.apply(x)/f.d(1,x);  
 while(true){  
 if(Math.*abs*(x1 - x) <= epsilon || Math.*abs*(f.apply(x1)) <= epsilon){  
 ans.solution = x1;  
 ans.value = f.apply(x1);  
 ans.iteration = Math.*abs*(x1 - x);  
 break;  
 }  
 double temp = x1;  
 x1 -= (x1 - x)\*f.apply(x1)/(f.apply(x1)-f.apply(x));  
 x = temp;  
 }  
}

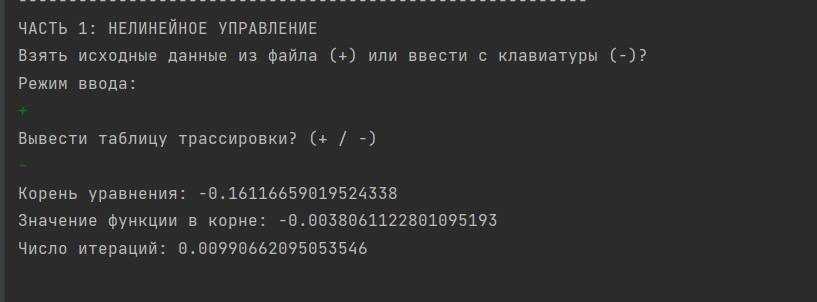
Метод простой интервал

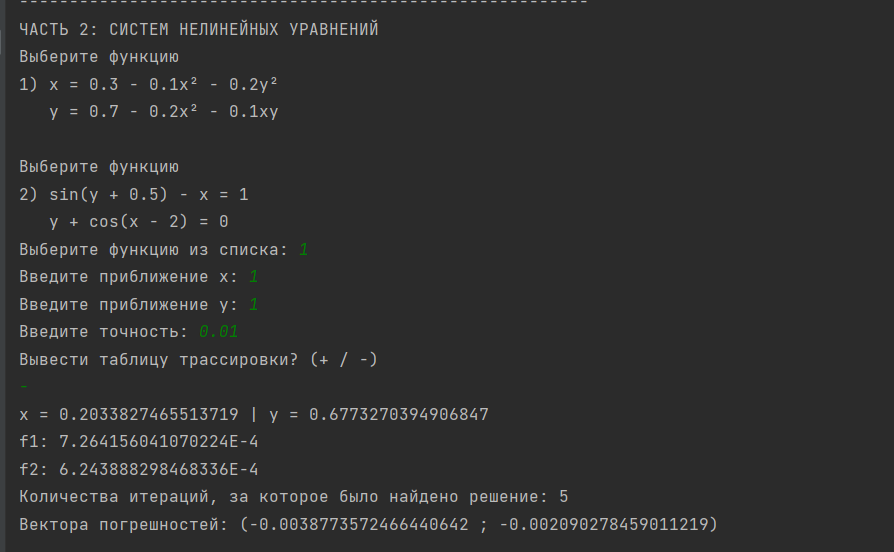
private void simpleIterationMethod() {  
 ans = new Answers();  
 Phi phi = new Phi(a,b,f);  
 if(Math.*max*(Math.*abs*(f.d(1,a)), Math.*abs*(f.d(1,b))) < 1) System.*out*.println("Условие сходимости ВЫПОЛНЯЕТСЯ");  
 else System.*out*.println("Условие сходимости НЕ ВЫПОЛНЯЕТСЯ");  
 while(true){  
 double x1 = phi.apply(x0);  
 if(Math.*abs*(x1 - x0) <= epsilon && Math.*abs*(f.apply(x1)) <= epsilon){  
 ans.solution = x1;  
 ans.value = f.apply(x1);  
 ans.iteration = Math.*abs*(x1 - x0);  
 break;  
 }  
 x0 = x1;  
 }  
}

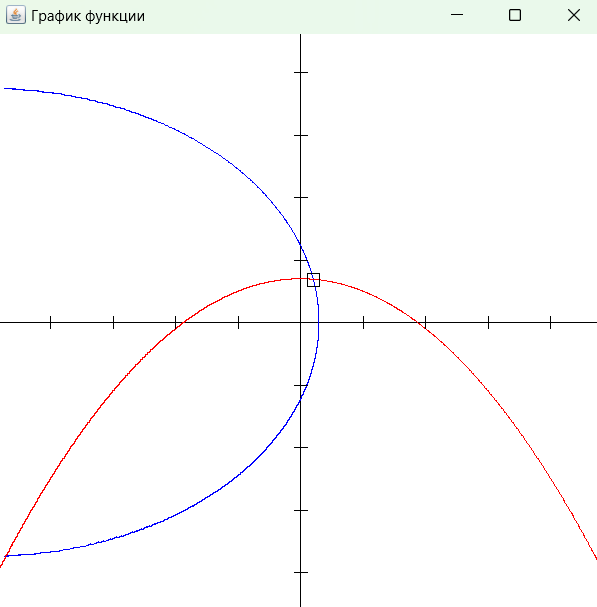
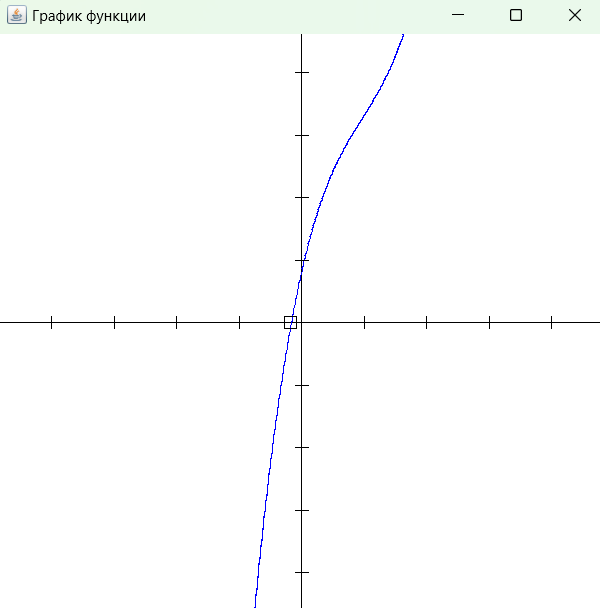
Метод простой интервал для решения систем нелинейных уравнений

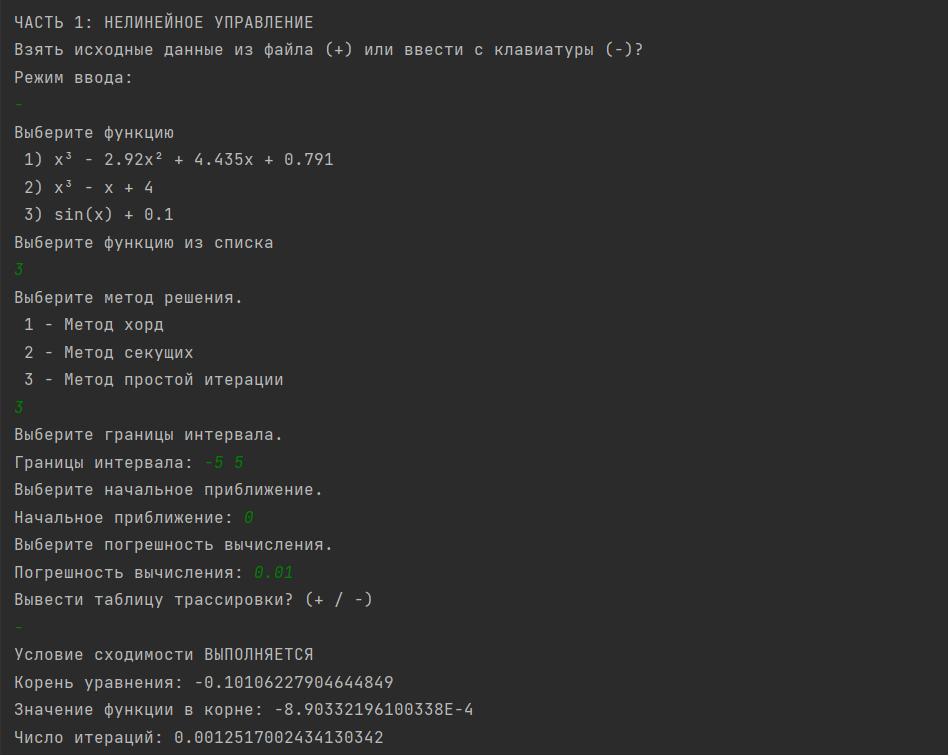
private Result simpleIterationMethod(double x, double y, double epsilon){  
 Result res = new Result();  
 double x0 = x, y0 = y;  
 if(Math.*max*((Math.*abs*(f.dx(x,y,1)) + Math.*abs*(f.dy(x,y,1))),(Math.*abs*(f.dx(x,y,2)) + Math.*abs*(f.dy(x,y,2)))) >= 1) {  
 System.*out*.println(f.dx(x,y,1));  
 System.*out*.println("условие сходимости итерационного процесса не выполнено");  
 }  
 while(true){  
 res.num++;  
 x = f.g\_x(x0,y0);  
 y = f.g\_y(x0,y0);  
 if(Math.*max*(Math.*abs*(x - x0), Math.*abs*(y-y0)) <= epsilon){  
 res.solutionX = x;  
 res.solutionY = y;  
 res.value1 = f.f1(x,y);  
 res.value2 = f.f2(x,y);  
 res.itr1 = x - x0;  
 res.itr2 = y - y0;  
 break;  
 }  
 x0 = x;  
 y0 = y;  
 }  
 return res;  
}

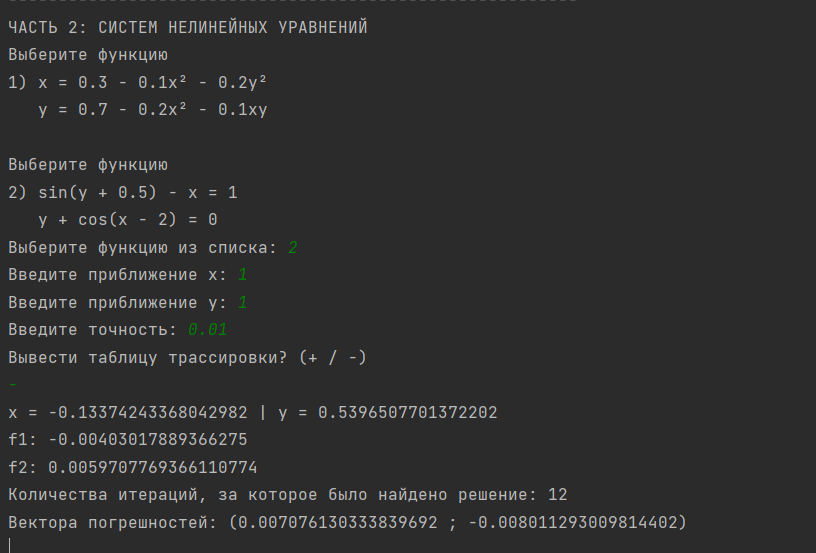
1. **Результаты выполнения программы**

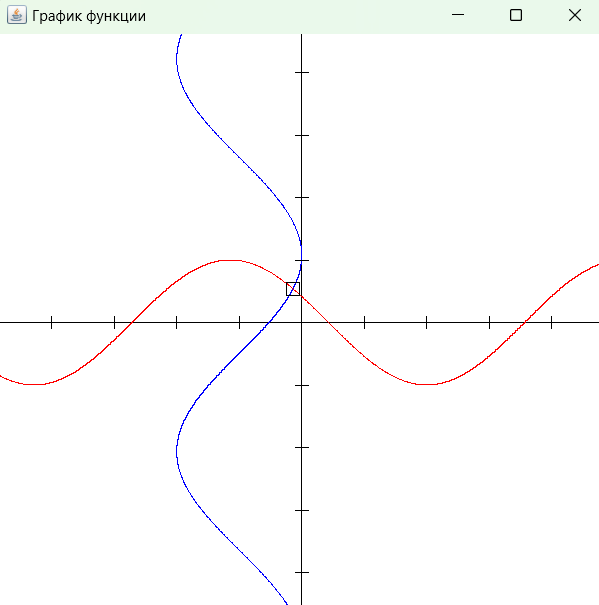
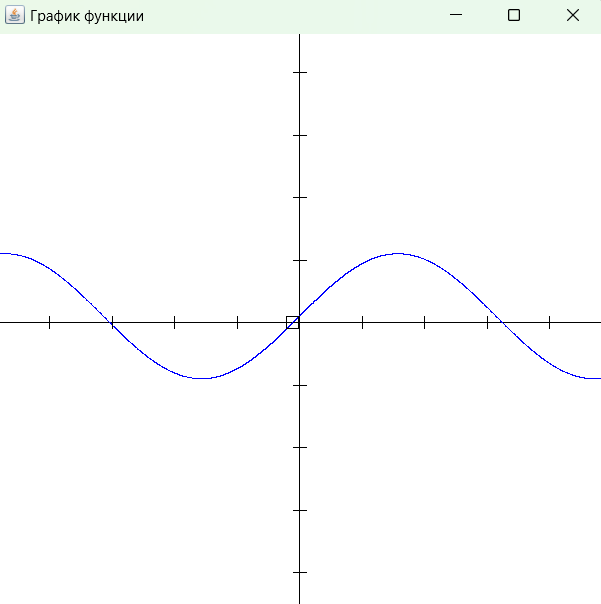
****

****

****

****

****

****

1. **Выводы**

В ходе выполнения лабораторной работы были изучены численные методы решения нелинейных уравнений и систем нелинейных уравнений с использованием Java. Была успешно реализована программа, предусматривающая выбор уравнений, методов решения, ввод исходных данных, проверку корректности данных и сходимости методов, а также вывод результатов на экран или в файл. 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.