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

Факультет Программной Инженерии и Компьютерной Техники **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Лабораторная работа №1 по дисциплине «Вычислительная математика»

Вариант 4



Выполнил:

Студент группы P3212

Данько Савелий Максимович

Преподаватель:

г. Санкт-Петербург

2025

**Цель лабораторной работы:**

Изучить численные методы решения нелинейных уравнений и их

систем, найти корни заданного нелинейного уравнения/системы нелинейных уравнений,

выполнить программную реализацию методов.

**Вычислительная реализация задачи**

**1. Решение нелинейного уравнения**

**Вид нелинейного уравнения для вычислительной реализации:**

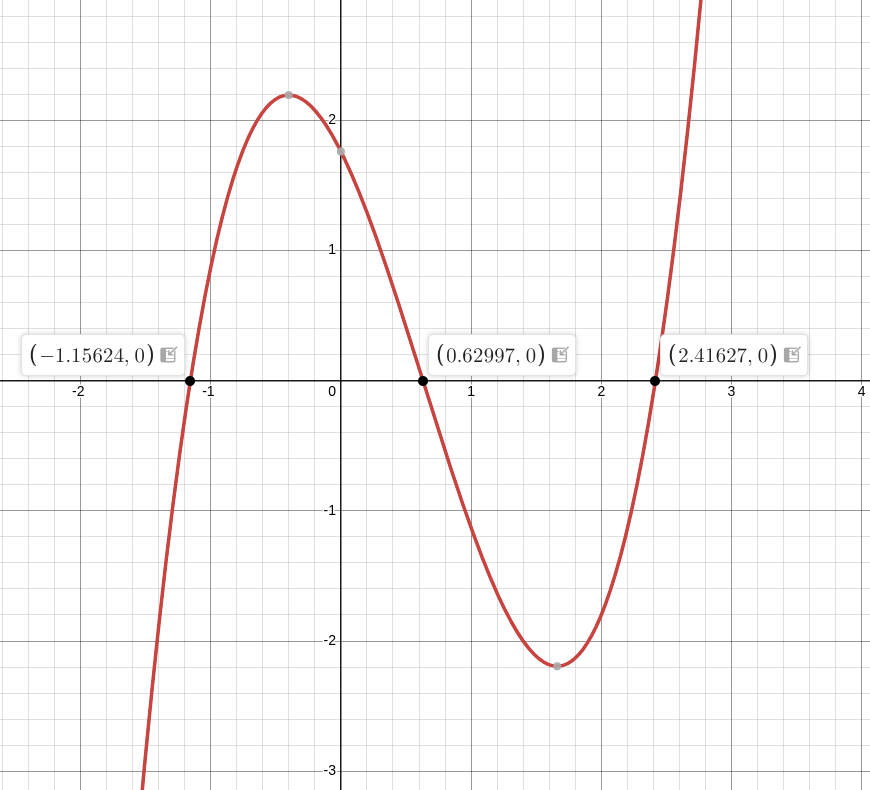
**Метода для вычислительной реализации:**

Крайний правый корень - Метод простых итераций

Крайний левый корень - Метод половинного деления

Центральный корень **-** Метод секущих

1. **Отделить корни нелинейного уравнения графически**

****

1. **Определить интервалы изоляции корней**

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.

Разобьем ось X на 4 интервала:

На каждом из этих интервалов нужно определить знак функции.

Для этого можем вычислить значения функции в произвольной точке каждого интервала.

для

для

для

для

|  |  |  |  |
| --- | --- | --- | --- |
| - | + | - | + |

Получаем следующие интервалы изоляции корней уравнения:

(-2; 0)

(0; 1)

(1; 3)

1. **Уточнить корни нелинейного уравнения**
2. **Уточнение корней многочлена**

Крайний правый корень - Метод простых итераций

Промежуток: (1;3)

итерационная последовательность сходится

****

| № |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2.000 | 2.140 | 2.200 | 0.140 |
| 2 | 2.140 | 2.235 | 2.905 | 0.095 |
| 3 | 2.235 | 2.299 | 3.483 | 0.064 |
| 4 | 2.299 | 2.340 | 3.922 | 0.041 |
| 5 | 2.340 | 2.367 | 4.224 | 0.027 |
| 6 | 2.367 | 2.384 | 4.432 | 0.017 |
| 7 | 2.384 | 2.396 | 4.458 | 0.012 |
| 8 | 2.396 | 2.403 | 4.664 | 0.007 |
| 9 | 2.403 | 2.407 | 4.722 | 0.004 |

Крайний левый корень - Метод половинного деления

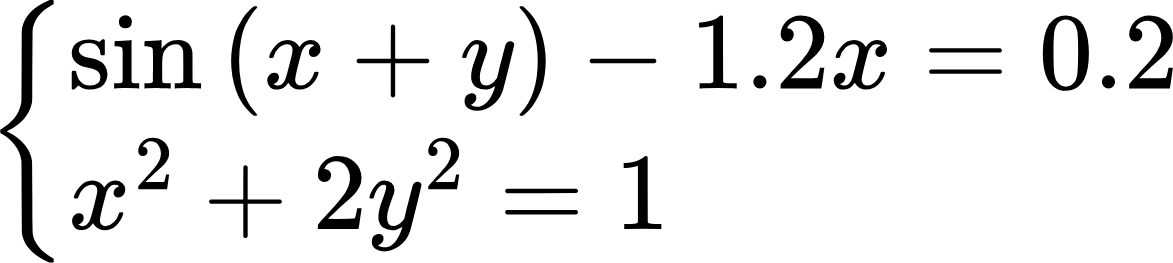
| № | a | b | x | f(a) | f(b) | f(x) | |a-b| |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | -2.000 | 0.000 | -1.000 | -9.800 | 1.760 | 0.870 | 2 |
| 2 | -2.000 | -1.000 | -1.500 | -9.800 | 0.870 | -2.867 | 1 |
| 3 | -1.500 | -1.000 | -1.250 | -2.867 | 0.870 | -0.646 | 0.5 |
| 4 | -1.250 | -1.000 | -1.125 | -0.646 | 0.870 | 0.194 | 0.25 |
| 5 | -1.250 | -1.125 | -1.187 | -0.646 | 0.194 | -0.201 | 0.125 |
| 6 | -1.187 | -1.125 | -1.156 | -0.201 | 0.194 | 0.001 | 0.062 |
| 7 | -1.187 | -1.156 | -1.171 | -0.201 | 0.001 | -0.095 | 0.031 |
| 8 | -1.171 | -1.156 | -1.163 | -0.095 | 0.001 | -0.043 | 0.015 |
| 9 | -1.163 | -1.156 | -1.159 | -0.-43 | 0.001 | -0.017 | 0.007 |
| 10 | -1.159 | -1.156 | -1.157 | -0.017 | 0.001 | 0.004 | 0.004 |

Центральный корень **-** Метод секущих

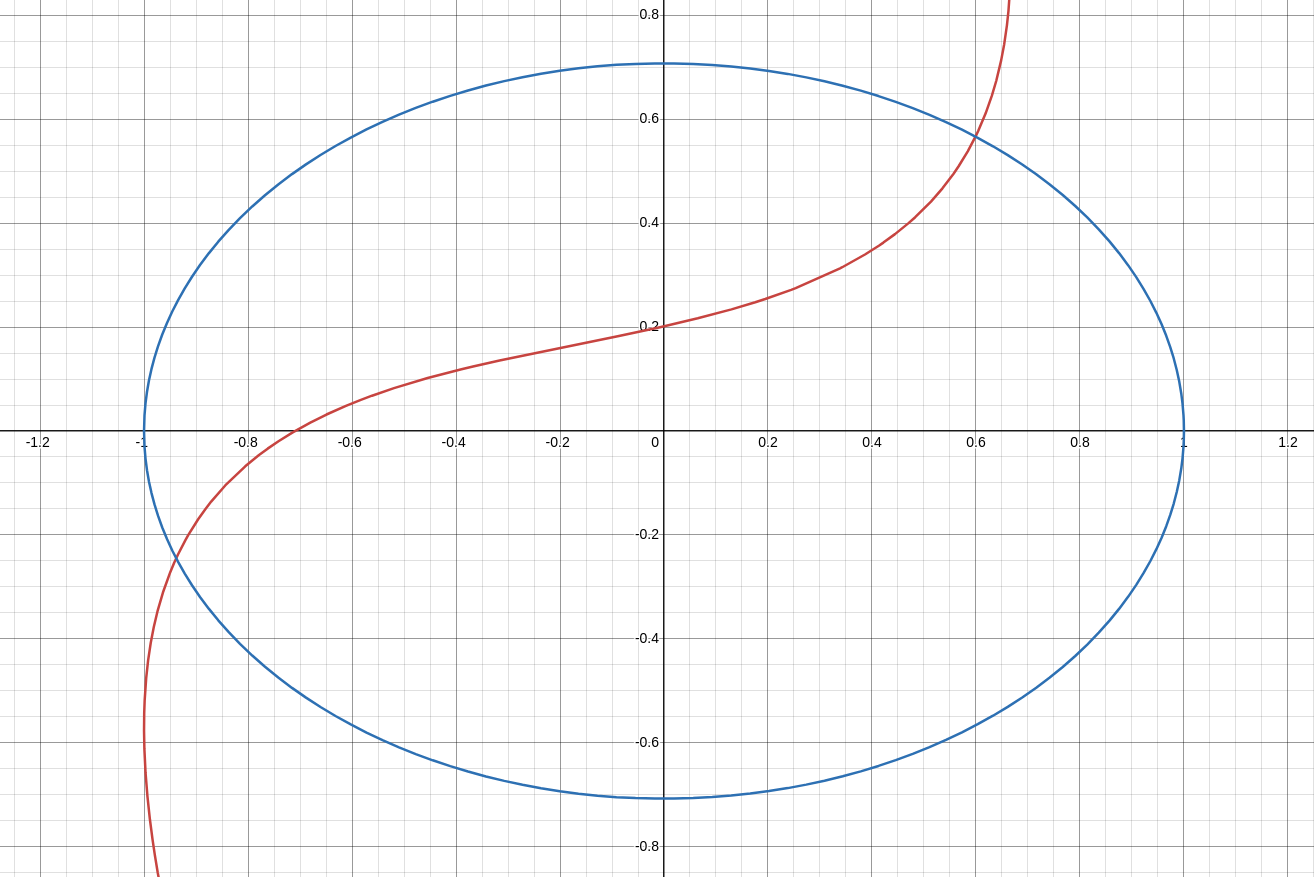
| № |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 0.000 | 1.000 | 0,608 | 0.070 | 0.392 |
| 2 | 1.000 | 0.608 | 0.631 | -0.003 | 0.023 |
| 3 | 0.608 | 0.631 | 0.629 | 0.003 | 0.002 |
| 4 | 0.631 | 0.629 | 0.629 | 0.003 | 0.000 |

**2. Решение системы нелинейных уравнений**

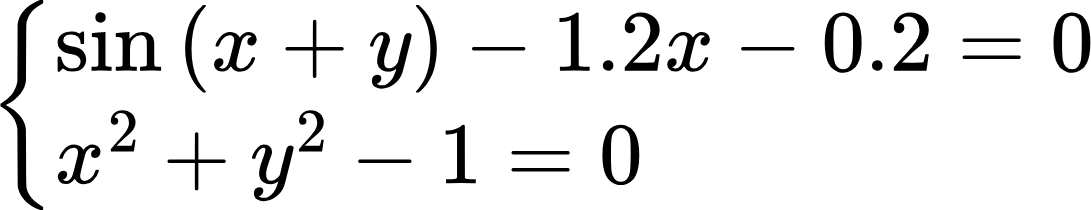
**Вид системы нелинейных уравнений:**

****

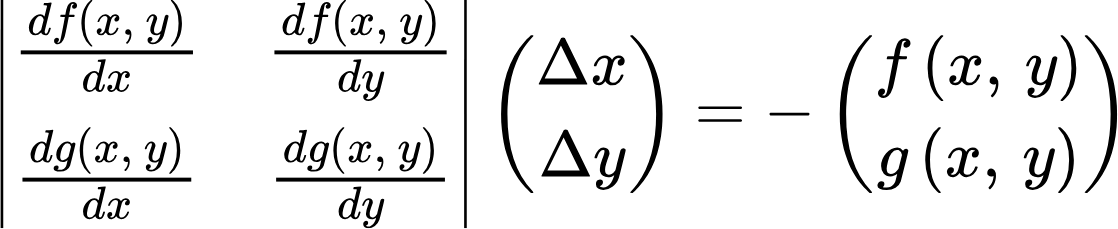
1. **Отделить корни системы нелинейных уравнений графически:**

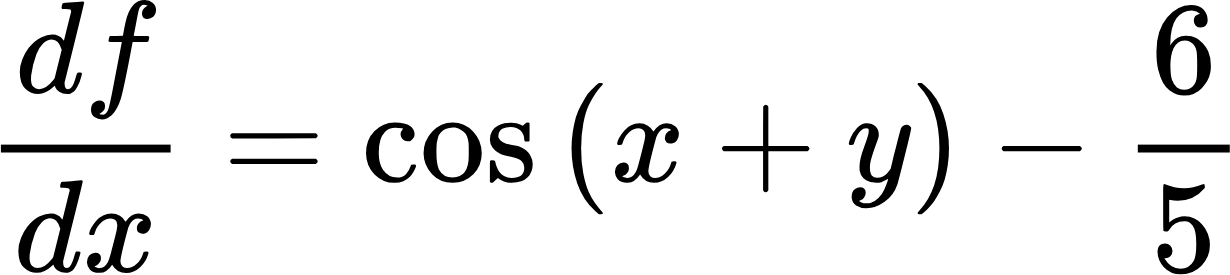
****

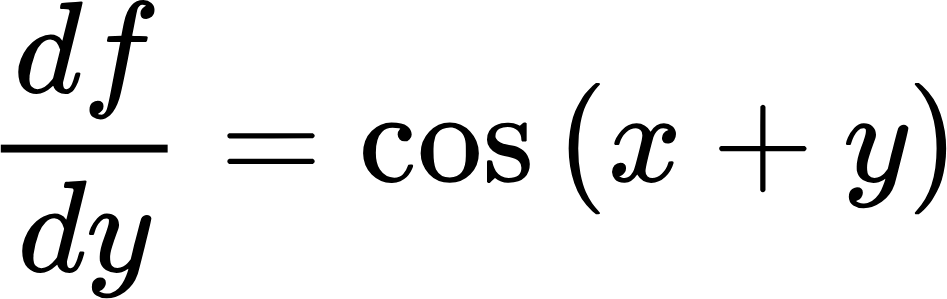
1. **Метод Ньютона:**

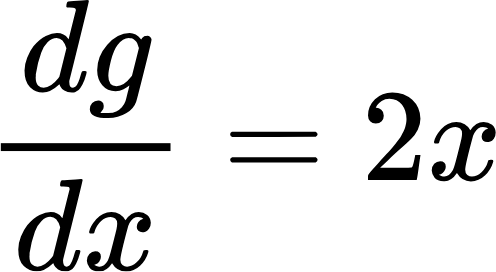
****

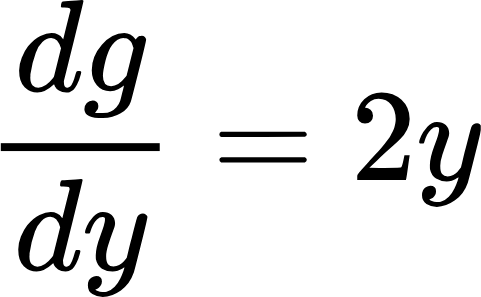
Матрица Якоби:

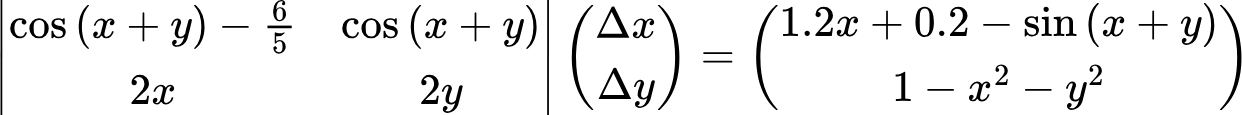


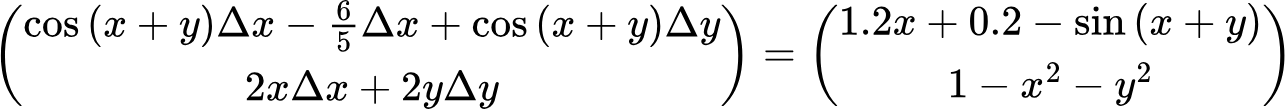












Возьмем начальное приближение

-0.369

-0.412

Аналогично находим другой корень

так как график симметричен относительно начала координат

**Программная реализация задачи**

**Код программы:**

[**git hub**](https://github.com/SaveliyDanko/NonlinearSystemSolver)

**Пример работы программы:**

**–**

Выберите действие:

1) Нелинейное уравнение

2) Система нелинейных уравнений

3) Выход

Введите номер пункта: 1

Доступные нелинейные уравнения:

1) x^2 - 5 = 0

2) sin(x) - x/2 = 0

3) e^x + x = 0

Введите номер уравнения (или 'q' для отмены): 1

Выберите метод решения нелинейного уравнения:

1) Метод хорд

2) Метод Ньютона

3) Метод простых итераций

Введите номер метода (или 'q' для отмены): 1

[Метод хорд] Решаем уравнение: x^2 - 5 = 0

Введите 'file' для чтения из файла или 'console' для ввода с консоли: console

Левая граница (a): -3

Правая граница (b): 0

Точность (eps): 0.01

Максимальное число итераций: 100

Найденный корень: -2.2357723577235773

Количество итераций: 5

Значение f(root): -0.0013219644391559981

**–**

Выберите действие:

1) Нелинейное уравнение

2) Система нелинейных уравнений

3) Выход

Введите номер пункта: 2

Доступные системы нелинейных уравнений:

1) x^2 + y^2 -1 = 0 и x^3 - y = 0

2) sin(x) + cos(y) = 0 и ln(x) + y^2 - 1 = 0

Введите номер системы (или 'q' для отмены): 1

Выберите метод решения системы нелинейных уравнений:

1) Метод простых итераций

Введите номер метода (или 'q' для отмены): 1

[Метод простых итераций] Решаем систему уравнений: ['x^2 + y^2 -1 = 0', 'x^3 - y = 0']

Введите 'file' для чтения параметров из файла или 'console' для ввода с консоли: console

Введите alpha (параметр релаксации): 0.01

Начальное приближение x0: 0.9

Начальное приближение y0: 0.6

Точность (eps): 0.01

Максимальное число итераций: 1000

Сходимость по изменению решения достигнута: ||Δ(x,y)|| = 0.002134033739189742 < 0.01

Результаты решения системы методом простых итераций с адаптивным шагом:

Найденное решение: x = 0.8983, y = 0.59871

Число итераций: 1

||F(x,y)|| = 0.20802423187619287

**Вывод:**

### **Вывод**

В ходе выполнения данной лабораторной работы я изучил различные методы решения нелинейных уравнений и систем нелинейных уравнений, а также реализовал их программно.

Метод половинного деления (метод бисекции)

Достоинства:

* Простая и надёжная идея, гарантирующая сходимость.
* Требует от функции f(x) только непрерывности, без необходимости вычисления производной.
* Обладает абсолютной сходимостью.

Рекомендация: Метод эффективен в ситуациях, когда важна высокая надёжность вычислений, а скорость работы не является критичной.

Недостатки:

* Если интервал содержит несколько корней, метод не позволяет определить, к какому из них стремится вычислительный процесс.
* Сходится медленно, поскольку обладает линейной сходимостью.

Метод Ньютона (касательных)

Достоинства:

* Квадратичная сходимость, что делает его значительно быстрее метода бисекции.

Недостатки:

* Требует, чтобы функция была дифференцируемой.
* На каждой итерации необходимо вычислять производную, что усложняет реализацию.
* Чувствителен к выбору начального приближения: неподходящий выбор может привести к расходимости.

Метод простой итерации

Достоинства:

* Простота реализации.

Недостатки:

* Сходимость возможна только в малой окрестности корня, поэтому требуется точный выбор начального приближения. В противном случае процесс может разойтись или сойтись к другому корню.
* Если ∣φ′(x)∣≈1, то скорость сходимости очень низкая.
* Требует преобразования уравнения к форме x=φ(x), что не всегда возможно.

Метод простой итерации для системы нелинейных уравнений

Достоинства:

* Легко реализуется программно.
* Гибкость метода позволяет адаптировать его к разным типам систем.
* Может быть распараллелен для ускорения вычислений.

Недостатки:

* Требует, чтобы начальное приближение находилось в малой окрестности корня, иначе процесс может расходиться.
* Медленная сходимость, если ∥Φ′(x)∥≈1.
* Не всегда применим, так как систему нужно преобразовать к форме x=Φ(x), что не всегда возможно.

Общий вывод:  
 Каждый из рассмотренных методов имеет свои сильные и слабые стороны. Метод бисекции надёжен, но медленен. Метод Ньютона быстрее, но требует вычисления производной и правильного начального приближения. Метод простой итерации прост в реализации, но его сходимость зависит от выбора начального приближения. Для систем уравнений метод простой итерации может быть полезен, но требует преобразования системы в удобный вид. Выбор метода зависит от конкретной задачи, требований к точности и вычислительным возможностям.