МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальная научно-образовательная корпорация ИТМО»

ФАКУЛЬТЕТ ПииКТ

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

«Вычислительная математика»

Вариант №3

Выполнил:

Студент группы P3219

Билобрам Денис Андреевич

Преподаватель:

Бострикова Дарья Константиновна

Санкт-Петербург, 2024

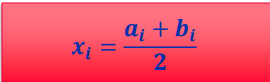
1. **Цель лабораторной работы:**

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

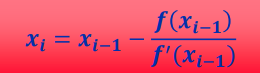
1. **Порядок выполнения:**
2. Вычислительная реализация задачи.
3. Решение нелинейного уравнения.
4. Решение системы нелинейных уравнений.
5. Программная реализация задачи.

**3. Рабочие формулы используемых методов:**

Метод половинного деления:



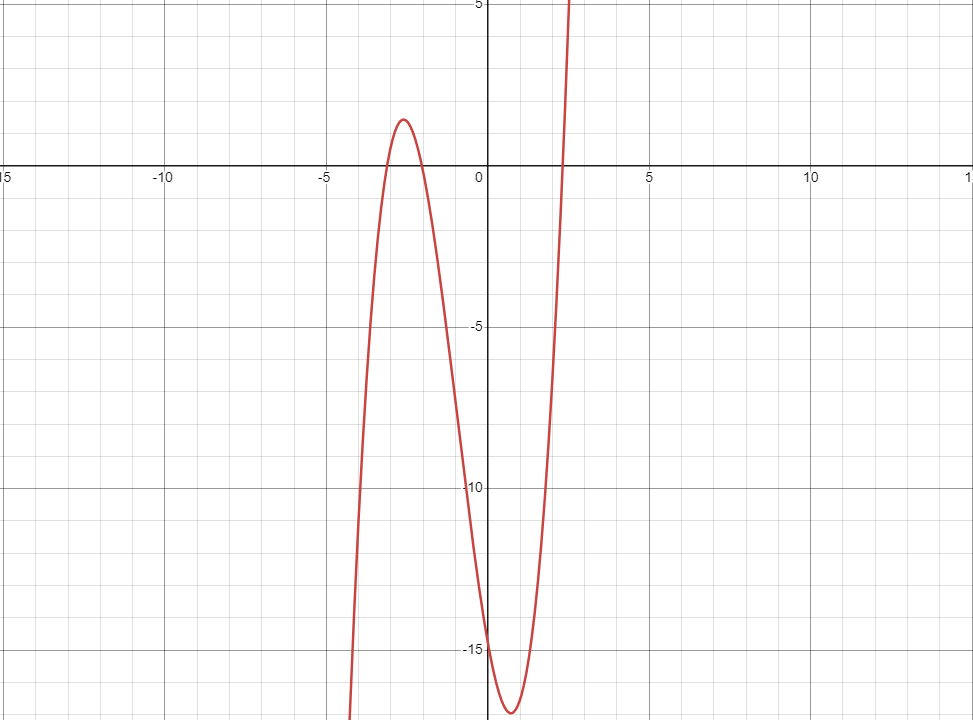
Метод Ньютона:

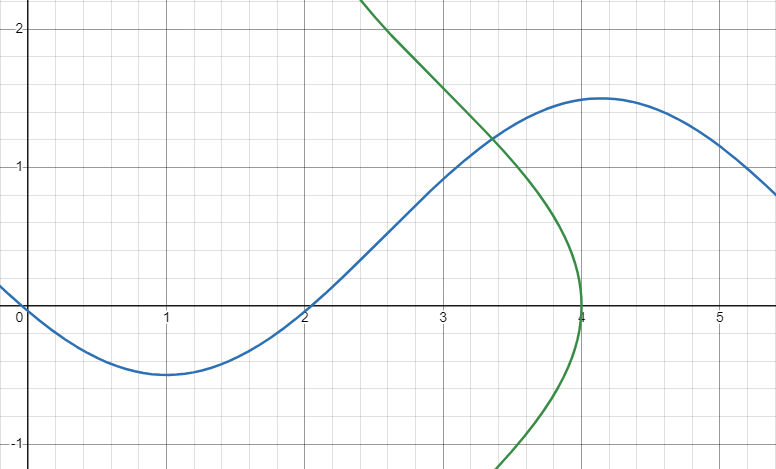


Метод простой итерации:



**4. Графики функций:**

****

****

**5. Таблицы вычислительной части 1:**

Крайне правый корень (метод половинного деления) [2; 3]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № шага | a | b | x | F(a) | F(b) | F(x) | |a-b| |
| 1 | 2 | 3 | 2.5 | -6,618 | 20,976 | 4,594 | 1 |
| 2 | 2 | 2,5 | 2.25 | -6,618 | 4,594 | -1,611 | 0,5 |
| 3 | 2,25 | 2,5 | 2,375 | -1,611 | 4,594 | 1,335 | 0,25 |
| 4 | 2,25 | 2,375 | 2,312 | -1,611 | 1,335 | -0,187 | 0,125 |
| 5 | 2,312 | 2,375 | 2,343 | -0,187 | 1,335 | 0,551 | 0,063 |
| 6 | 2,312 | 2,343 | 2,327 | -0,187 | 0,551 | 0,168 | 0,031 |
| 7 | 2,312 | 2,327 | 2,320 | -0,187 | 0,168 | 0,001 | 0,015 |
| 8 | 2,312 | 2,320 | 2,316 | -0,187 | 0,001 | -0,093 | 0,008 |

Крайне левый корень (метод простой итерации) [-3,5; -3]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № итерации |  |  |  |  |
| 1 | -3,5 | -3,243 | -0,824 | 0,257 |
| 2 | -3,243 | -3,193 | -0,464 | 0,050 |
| 3 | -3,193 | -3,174 | -0,337 | 0,019 |
| 4 | -3,174 | -3,166 | -0,285 | 0.008 |

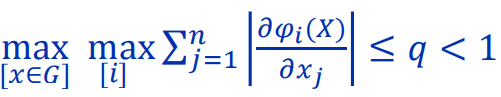
Центральный корень (метод Ньютона) [-2,5; -1,5]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № итерации |  |  |  |  |  |
| 1 | -2,5 | 1,374 | -1,056 | -1,99 | 0,51 |
| 2 | -1,99 | -0,244 | -5,028 | -2.038 | 0,048 |
| 3 | -2.038 | -0,099 | -4,721 | -2.04 | 0.002 |

**6. Решение системы нелинейных уравнений**

Метод простой итерации

Условие сходимости:



;; *;*

на [3; 4] < 0,91

на [1; 1,3] < 0,97

Следовательно, условие сходимости выполняется.

0.

1*.*

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

16.

17.

18.

19.

20.

**7. Листинг программы**

def bisection\_method(f, a, b, epsi):

    c = 0

    while (b - a) / 2.0 > epsi:

        c += 1

        midpoint = (a + b) / 2.0

        if (f(midpoint) == 0):

            break

        elif f(a) \* f(midpoint) < 0:

            b = midpoint

        else:

            a = midpoint

    print("Метод половинного деления: x = ", round((a + b) / 2.0, 6), "; Число итераций: ", c, ";")

def secant\_method(f, f\_p2, a, b, epsi):

    fa = f(a)

    fp2a = f\_p2(a)

    if fa \* fp2a > 0:

        x\_0 = a

    else:

        x\_0 = b

    x\_1 = a if x\_0 == b else b

    c = 0

    for \_ in range(1000):

        c += 1

        fx\_0 = f(x\_0)

        fx\_1 = f(x\_1)

        x\_new = x\_1 - fx\_1 \* (x\_1 - x\_0) / (fx\_1 - fx\_0)

        if abs(x\_new - x\_1) < epsi:

            break

        x\_0, x\_1 = x\_1, x\_new

    print("Метод секущих: x = ", round(x\_new, 6), "; Число итераций: ", c, ";")

def simple\_iteration\_method(f, fi, fi\_p, a, b, x0, epsi):

    if abs(fi\_p(a)) >= 1 or abs(fi\_p(b)) >= 1:

        print("Метод простой итерации расходится на интервале.")

        return

    else:

        x\_prev = x0

        x\_curr = fi(x\_prev)

        c = 0

        while abs(x\_curr - x\_prev) > epsi and abs(f(x\_curr)) > epsi:

            x\_prev = x\_curr

            x\_curr = fi(x\_prev)

            c += 1

            if abs(fi\_p(x\_curr)) >= 1:

                print("Метод может не сойтись на шаге", c)

                return

            if (x\_curr < a or x\_curr > b):

                print("Новое приближение метода простой итерации выходит за пределы интервала на шаге", c)

                return

    print("Метод простой итерации: x = ", round(x\_curr, 6), "; Число итераций: ", c, ";")

def newton\_method(F, J, x0, epsi=0.01, max\_iter=100):

    x = np.array(x0, dtype=float)

    errors = []

    for i in range(max\_iter):

        Fx = F(x)

        Jx = J(x)

        delta = solve\_system(Jx, -Fx)

        x\_new = x + delta

        error = norm(delta)

        errors.append(error)

        if error < epsi and delta[0] < epsi and delta[1] < epsi:

            print(f"Решение найдено: {x\_new} после {i+1} итераций.")

            print(f"Вектор погрешностей: {errors}")

            if np.linalg.norm(F(x\_new)) < epsi:

                print("Проверка решения: УСПЕШНО")

            else:

                print("Проверка решения: НЕ УСПЕШНО")

            return x\_new

        x = x\_new

    print("Решение не найдено.")

    exit(1)

**8. Результаты выполнения программы**

**Уравнения:**

**1) cos(x) - x = 0**

**2) x^3 - x = 0**

**3) e^x - 2x - 5 = 0**

**4) x^2 + 2x - 3 = 0**

**Введите номер уравнения: 2**

**Введите номер способа ввода начальных данных - 1) Клавиатура 2) Файл : 1**

**Введите начальную границу интервала a: -2**

**Введите конечную границу интервала b: -0.5**

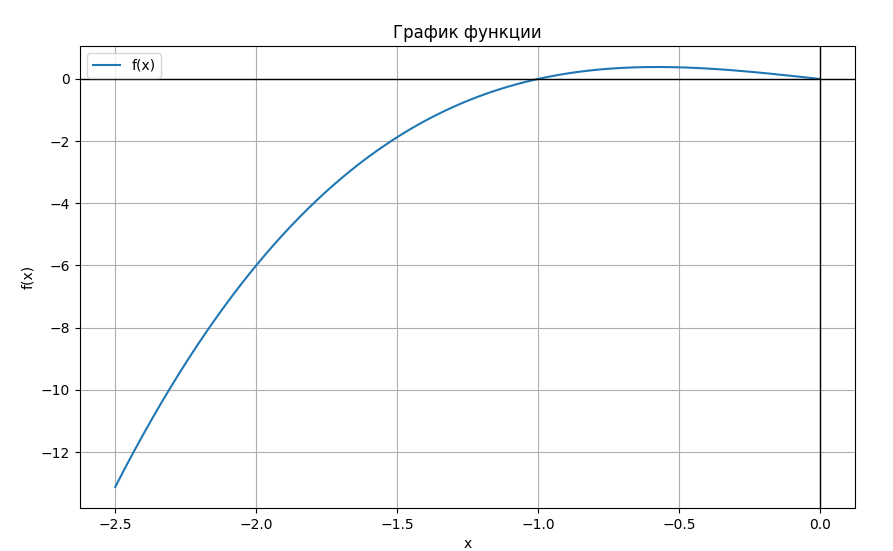
**Введите начальное приближение x0: -1.5**

**Введите погрешность epsi: 0.01**

**Метод половинного деления: x = -0.998047 ; Число итераций: 7 ;**

**Метод секущих: x = -0.999622 ; Число итераций: 10 ;**

**Метод простой итерации: x = -1.00167 ; Число итераций: 4 ;**

**  
  
  
Уравнения:**

**1) cos(x) - x = 0**

**2) x^3 - x = 0**

**3) e^x - 2x - 5 = 0**

**4) x^2 + 2x - 3 = 0**

**Введите номер уравнения: 1**

**Введите номер способа ввода начальных данных - 1) Клавиатура 2) Файл : 1**

**Введите начальную границу интервала a: 0**

**Введите конечную границу интервала b: 2**

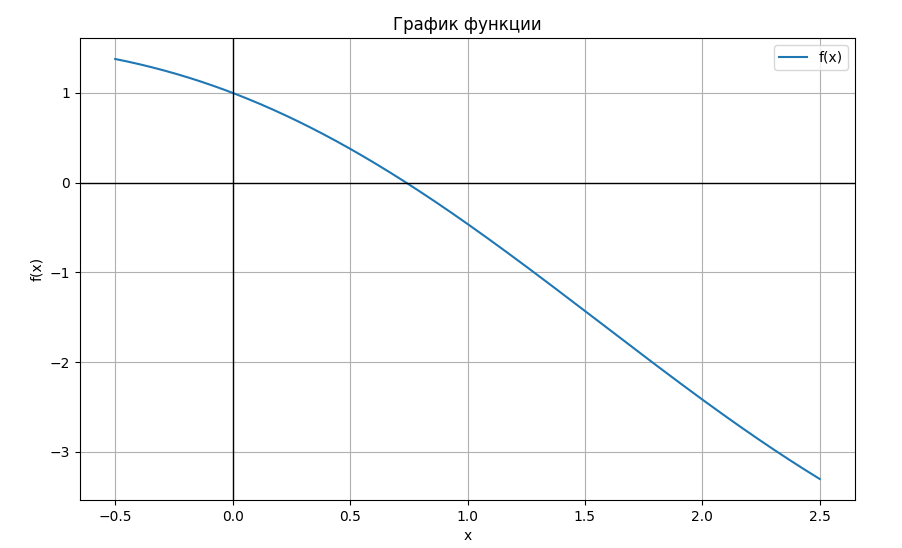
**Введите начальное приближение x0: 0**

**Введите погрешность epsi: 0.01**

**Метод половинного деления: x = 0.742188 ; Число итераций: 7 ;**

**Метод секущих: x = 0.739073 ; Число итераций: 4 ;**

**Метод простой итерации: x = 0.744237 ; Число итераций: 10 ;**

****

**Выберите систему уравнений для решения:**

**1: x^2 + y^2 - 4 = 0 и x^2 - y - 1 = 0**

**2: sin(x) + y^2 - 1.5 = 0 и x^2 - cos(y) - 0.5 = 0**

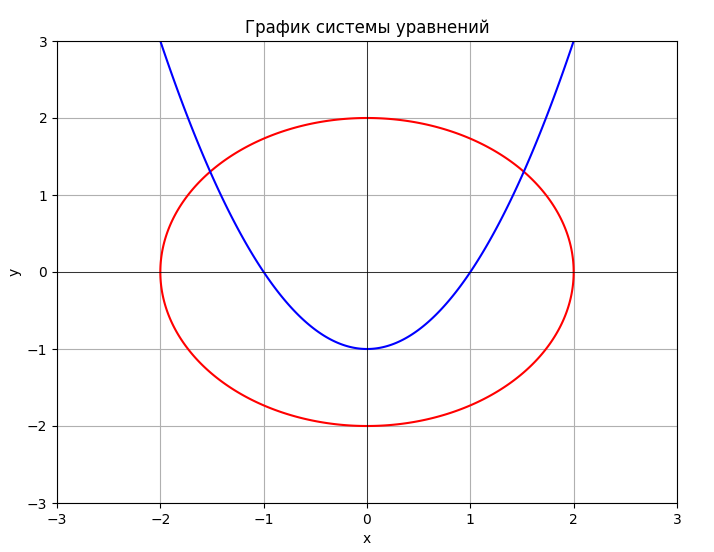
**Введите номер системы: 1**

**Введите начальные приближения x и y через пробел: 0.5 0.5**

**Решение найдено: [1.51748994 1.30277564] после 5 итераций.**

**Вектор погрешностей: [2.62797450520358, 1.0748451973479936, 0.29706793539670484, 0.028401150761749115, 0.00026572330198029563]**

**Проверка решения: УСПЕШНО**

****

**9. Выводы**

В ходе выполнения лабораторной работы были рассмотрены и реализованы методы численного решения нелинейных уравнений и систем нелинейных уравнений, в частности, метод половинного деления, метод секущих, метод простой итерации и метод Ньютона. Основное внимание уделялось пониманию принципов работы каждого из методов, а также особенностям их программной реализации на языке Python.