МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Вычислительная математика»

Тема: Метод бисекции

Студент гр. 3341 _____ Трофимов В.О.

Преподаватель _____ Пуеров Г.Ю.

Санкт-Петербург

2025

Цель работы

Изучить и реализовать метод бисекции для численного решения уравнения f(x)=0, а также исследовать зависимость числа итераций от заданной точности Epsilon и оценить чувствительность метода к ошибкам в исходных данных.

Задание

В лабораторной работе №3 предлагается, используя программы - функции BISECT и Round из файла methods.cpp (файл заголовков metods.h, дирек14 тория LIBR1), найти корень уравнения f (x) =0 методом бисекции с заданной точностью Eps, исследовать зависимость числа итераций от точности Eps при изменении Eps от 0.1 до 0.000001, исследовать обусловленность метода (чувствительность к ошибкам в исходных данных). Выполнение работы осуществляется по индивидуальным вариантам заданий (нелинейных уравнений), приведенным в подразделе 3.6. Номер варианта для каждого студента определяется преподавателем.

Вариант 20:

f(x) = exp(-x) – x^ 3

Порядок выполнения работы должен быть следующим:

1) Графически или аналитически отделить корень уравнения f (x) =0 (т.е. найти отрезки [Left, Right], на которых функция f (x) удовлетворяет условиям теоремы Коши).

2) Составить подпрограмму вычисления функции f (x).

3) Составить головную программу, содержащую обращение к подпрограмме f(x), BISECT, Round и индикацию результатов.

4) Провести вычисления по программе. Построить график зависимости числа итераций от Eps.

5) Исследовать чувствительность метода к ошибкам в исходных данных. Ошибки в исходных данных моделировать с использованием программы Round, округляющей значения функции с заданной точностью Delta.

Выполнение работы

Аналитически отделим корни уравнения (Найдём интервал [left, right]):

Рассмотрим уравнение f(x) = exp(-x) – x^ 3 = 0

Функция непрерывна (состоит из экспоненты и многочлена). По теореме Коши о промежуточных значения, если f(x) непрерывна на отрезке [a, b] и принимает значения разных знаков на концах отрезка, то внутри этого отрезка существует хотя бы одна точка c, в которой f(c) = 0.

Возьмём отрезок [0, 1] и вычислим:

f(0) = exp(0) – 0^3 = 1, что f(0) > 0

f(1) = exp(-1) – 1^3 = 1 / exp – 1, при exp ≈ 2,7 => f(1) < 0

Так как f(0) > 0 и f(1) < 0, выбираем отрезок [left, right] = [0, 1]

Реализация алгоритма на Python

Функции использованные для решения задачи:

def f(x) – функция принимает один аргумент x, являющийся точкой, в которой нужно вычислить значение f(x)

def Round(x, delta) – Функция принимает два аргумента x – число, которое нужно округлить, delta – величина округления, до которой будет округлено число x. Round(x, delta) выполняет округление числа x до ближайшего значения, кратного delta.

def Bisect(left, right, epsilon, delta) – функция, использующая метод бисекции для нахождения корня функции.

Метод бисекции ищет корень функции на интервале [left, right], где функция меняет знак (то есть, f(left) и f(right) имеют противоположные знаки).

Проверяем, если длина интервала right − left больше заданной точности epsilon, продолжаем работать.

На каждой итерации ищем середину отрезка middle = (left + right) / 2.

Округляем значения функции на этой средней точке и на левой границе интервала с учетом погрешности delta с помощью функции Round.

Если f(middle) = 0, то нашли корень и возвращаем его

Если произведение f(left) × f(middle) < 0, это означает, что корень находится на интервале [left, middle], и сужаем интервал, right = middle.

Если произведение f(middle) × f(right) < 0, то корень лежит на интервале [middle, right], и сужается интервал left = middle.

Дальше на каждом шаге увеличиваем счетчик итераций iter_count для того, чтобы в итоге можно было подсчитать, сколько итераций потребовалось для достижения желаемой точности.

def main() – головная процедура, внутри задаётся интервал [left, right], eps_values — список значений точности epsilon, с которыми будет производиться вычисление корня, delta_values — список значений округления delta, используемых для моделирования погрешностей, также вызывается функция plot_iterations, которая содержит в себе исследование зависимости изменения epsilon от числа итераций, исследование чувствительности метода к ошибкам в исходных данных(подробнее см. Тестирование).

Тестирование

Задачи:

1. Проверить правильность реализованного алгоритма с уже встроенным в модуле scipy.optimize bisect

2. Необходимо исследовать зависимость изменения epsilon от числа итераций

3. Необходимо исследовать чувствительность метода к ошибкам в исходных данных. Ошибки моделируются с использованием программы Round, округляющей значения функции с заданной точностью Delta.

Функция test_bisect проводит несколько тестов для проверки корректности работы метода бисекции, сравнивая результаты с библиотечным методом scipy.optimize.bisect.

Тест 1: Проверка корня:

Для каждого значения точности eps из списка eps_values функция Bisect и scipy_bisect вычисляют корни.

Проверка, что разница между корнями не превышает eps, что подтверждает корректность работы метода с заданной точностью.

Тест 2: Проверка числа итераций:

Для каждого значения eps проверяется, что количество итераций метода бисекции не уменьшается при уменьшении точности. Это гарантирует, что метод выполняет больше шагов при более высокой точности, как ожидается от численных методов.

Функция explore_iterations_by_eps_and_delta исследует, как количество итераций метода бисекции зависит от двух параметров: eps — точность (на которой должен быть найден корень), delta — погрешность округления значений функции, используемая в методе.

Параметры:

left, right — границы интервала для поиска корня,

eps_values — список значений точности, для которых будет исследована зависимость,

delta_values — список значений погрешности округления для анализа.

Функция plot_iterations строит график зависимости числа итераций от точности eps для разных значений погрешности delta.

Параметры:

left, right — границы интервала для поиска корня (те же, что в предыдущей функции),

eps_values — список значений точности,

delta_values — список значений погрешности округления.

Алгоритм:

Вызывает функцию explore_iterations_by_eps_and_delta, чтобы получить данные о числе итераций для различных значений eps и delta.

Строит график с помощью библиотеки matplotlib. Для каждого значения delta строится линия на графике, где по оси X откладываются значения eps (с логарифмической шкалой), а по оси Y — количество итераций.

Результаты тестирования представлены в таблице № 1 и результаты исследований представлены на рисунке № 1.

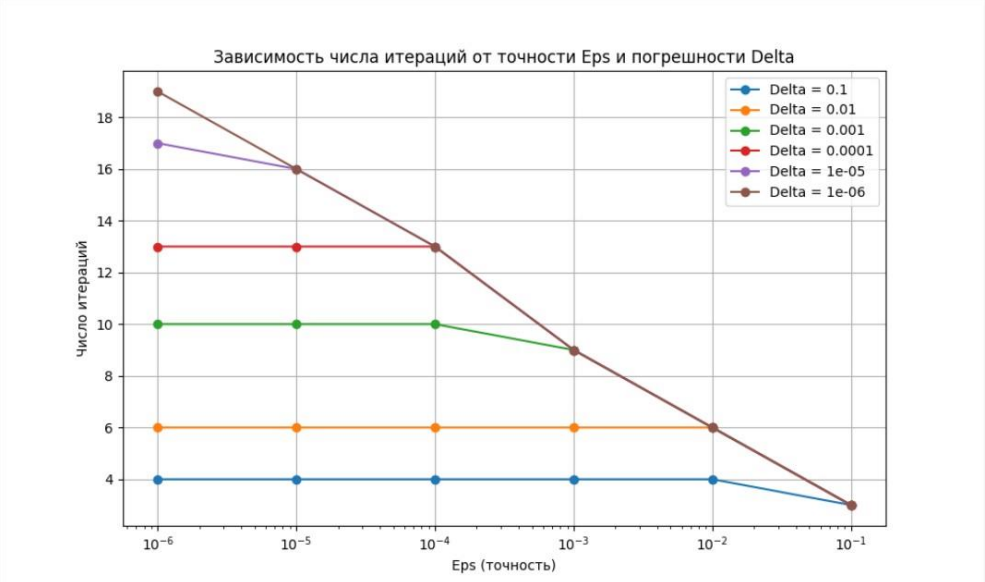

Рисунок № 1 – Зависимость числа итераций от точности Eps.

Таблица 1 – Результаты тестирования

epsilon	Полученный результат (Корень)	Ожидаемый результат	Комментарии
1e-1	0.8125	0.8125	Верно
1e-2	0.7734375	0.7734375	Верно
1e-3	0.7724609375	0.7724609375	Верно
1e-4	0.77288818359375	0.77288818359375	Верно
1e-5	0.7728805541992188	0.7728805541992188	Верно
1e-6	0.772883415222168	0.772883415222168	Верно

Выводы

В ходе выполнения задания, направленного на исследование метода бисекции для нахождения корней уравнения f(x) = exp(-x) – x^3 = 0, можно подвести следующие итоги:

1. Реализованный метод бисекции корректно находит корень функции на заданном интервале с необходимой точностью.

2. Метод является чувствительным к ошибкам округления, и погрешности могут влиять на конечный результат, особенно при больших значениях погрешности округления.

3. Для более точных расчетов важно учитывать влияние округлений и выбирать подходящее значение δ, чтобы минимизировать ошибки.

4. График зависимости числа итераций от точности подтвердил, что увеличение точности требует большего количества шагов, что соответствует теоретическим ожиданиям.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import math
import numpy as np
from scipy.optimize import bisect as scipy_bisect
import matplotlib.pyplot as plt


# Вариант 20: функция f(x) = exp(-x) - x^3
def f(x):
    return math.exp(-x) - x ** 3


def Round(x, delta):
    if x == 0.0 or delta == 0.0:
        return x  # без округления
    if x > 0.0:
        return delta * int((x / delta) + 0.5)
    else:
        return delta * int((x / delta) - 0.5)

def Bisect(left, right, epsilon, delta):
    iter_count = 0
    while (right - left) / 2 > epsilon:
        middle = (left + right) / 2
        f_middle = Round(f(middle), delta)
        f_left = Round(f(left), delta)

        if f_middle == 0:
            return middle, iter_count

        if f_left * f_middle < 0:
            right = middle
        else:
            left = middle

        iter_count += 1

    return (left + right) / 2, iter_count


def explore_iterations_by_eps_and_delta(left, right, eps_values,
delta_values):
    iterations_by_delta = {}

    for delta in delta_values:
        iterations = []
        for eps in eps_values:
            _, iter_count = Bisect(left, right, eps, delta)
            iterations.append(iter_count)
        iterations_by_delta[delta] = iterations
```

```
        return iterations_by_delta


    def plot_iterations(left, right, eps_values, delta_values):
        iterations_by_delta = explore_iterations_by_eps_and_delta(left,
right, eps_values, delta_values)
        plt.figure(figsize=(10, 6))
        for delta, iterations in iterations_by_delta.items():
            plt.plot(eps_values, iterations, marker='o', label=f'Delta
= {delta}')

        plt.xscale('log')  # Логарифмическая шкала по оси X
        plt.xlabel('Eps (точность)')
        plt.ylabel('Число итераций')
        plt.title('Зависимость числа итераций от точности Eps')
        plt.legend()
        plt.grid(True)
        plt.show()

    def test_bisect(left, right):
            eps_values = [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6]

            for eps in eps_values:
                root_custom, _ = Bisect(left, right, eps, delta=0)
                root_scipy = scipy_bisect(f, left, right, xtol=eps)
                assert abs(root_custom - root_scipy) < eps, f"Ошибка:
{root_custom} != {root_scipy} при eps={eps}"
                print(f"eps={eps}: {root_custom} == {root_scipy}")

            prev_iters = 0
            for eps in eps_values:
                _, iter_count = Bisect(left, right, eps, delta=0)
                assert iter_count >= prev_iters, f"Ошибка: итерации умен
ьшились при eps={eps}"
                prev_iters = iter_count

            print("Тест 1: Проверка корня (должен совпадать с
scipy.optimize.bisect)")
            print("Тест 2: Проверка числа итераций (должно увеличиваться при
уменьшении eps)")

    def main():
        left, right = map(float,input().split())  # Начальные границы от
резка
        test_bisect(left, right)
        eps_values = [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6]  # Множество з
начений Eps
```

```
        delta_values = [1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6]  # Множеств
о значений Delta для моделирования ошибок
        plot_iterations(left, right, eps_values, delta_values)

    main()
```