Молдавский Государственный Университет
Факультет Математики и Информатики
Департамент Информатики

Лабораторная работа №1
по курсу “Calcul Numeric si Metode de Optimizare”

Тема: “Решение алгебраических и трансцендентных уравнений”

Выполнил: Slavov Constantin,
студент группы I2302
Проверил: I. Verlan,
doctor, conferențiar universitar

Кишинев, 2025

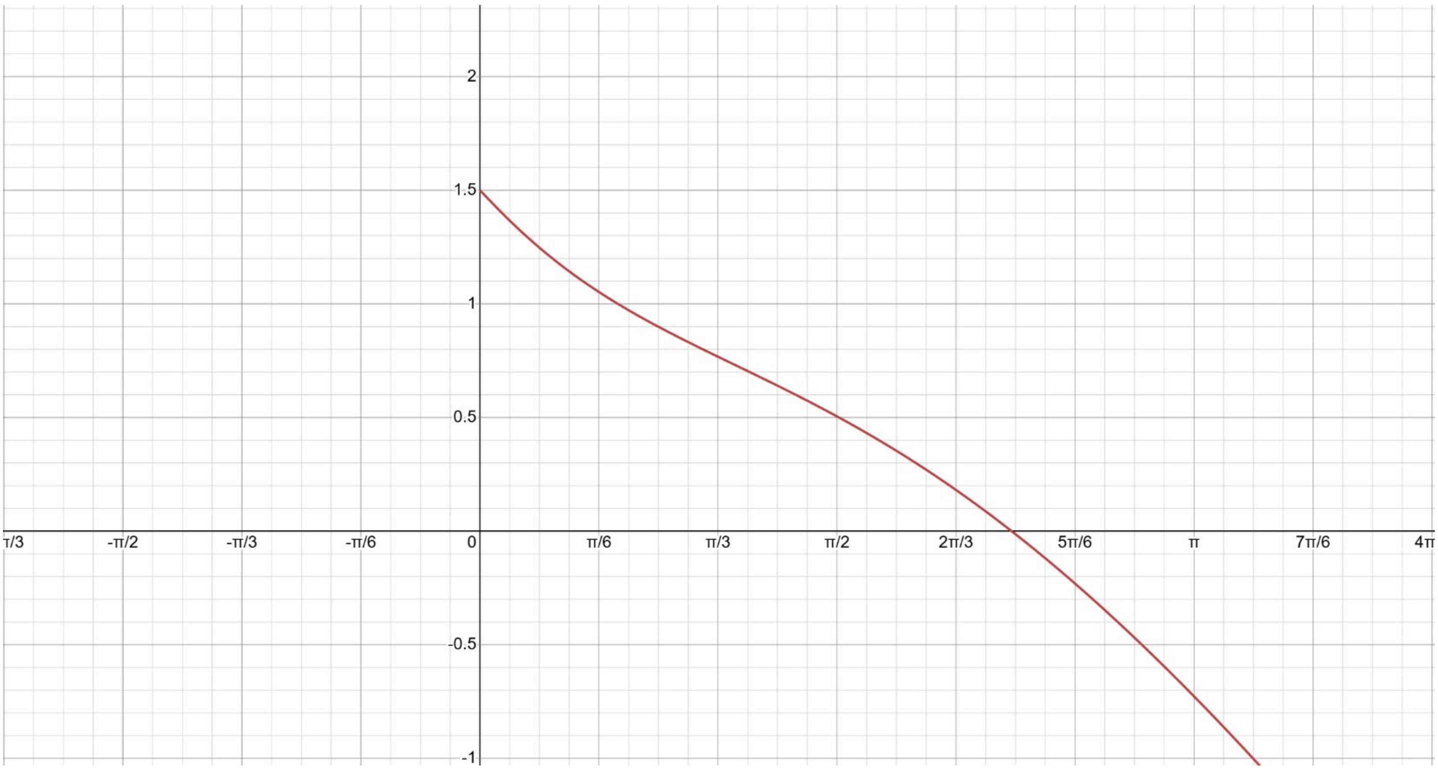
Условие задачи:
- Определить корень уравнения графическим методом на каком-то отрезке длиной меньше либо равно 1.
- Найти приближенные значения корня с точностью E = 10^-6 с помощью методов:
1. Метод бисекции
2. Метод Хорд
3. Метод Ньютона
4. Метод простой итерации
- Сделать сравнительный анализ методов

Ход работы:

Дано уравнение: $f(x) = 1.5 - 0.4\sqrt{x^3} - e^{-x}\sin x = 0$

1. Графический метод

1.1. Построение графика функции. Я выбрал несколько диапазонов, чтобы найти оптимальные значения для изменения знаков.
- Выбрал диапазон отрезка [0, 1] и вычислил значения функции, но график не показал пересечения с осью x.
- Выбрал диапазон отрезка [0, 2] и вычислил значения функции, но график снова не показал пересечения с осью x.
- Выбрал диапазон отрезка [0, 3] и вычислил значения функции, и нашли изменение знака около x ~ 2.33, значит, там есть корень.



1.2. Проверка изменения знака на отрезке.

- Вычислил значения на концах отрезка [2.3, 2.4]:

f(2.3) = 0.02999, f(2.4) = -0.04850

- Так как f(2.3) > 0 и f(2.4) < 0, значит на этом отрезке есть корень.

2. Численные методы

2.1. Метод бисекции

Итерация	x_a	x_b	x_(n+1)	f(x_(n+1))
1	2.300000	2.400000	2.350000	0.000972
2	2.350000	2.400000	2.370000	-0.023624
3	2.350000	2.375000	2.362500	-0.011369

Окончательный ответ: ***x = 2.338704***

2.2. Метод хорд

Итерация	x_n	x_(n-1)	f(x-n)	x_(n+1)
1	2.400000	2.300000	-0.048502	2.338971
2	2.338971	2.400000	0.000325	2.338705
3	2.338705	2.338971	0.000001	2.338704

Окончательный ответ: ***x = 2.338704***

2.3. Метод Ньютона

Итерация	x_n	f(x-n)	f'(x-n)	x_(n+1)
1	2.350000	0.000972	-2.958264	2.338704
2	2.338704	0.000001	-2.952003	2.338704
3	2.338704	0.000000	-2.952002	2.338704

Окончательный ответ: ***x = 2.338704***

2.4. Метод простой итерации

Итерация	x_n	g(x_n)
1	2.350000	2.341155
2	2.341155	2.339239
3	2.339239	2.338821

Окончательный ответ: ***x = 2.338704***

3. Сравнительный анализ методов

1. Метод бисекции

Метод бисекции основан на делении отрезка пополам и выборе подотрезка, в котором функция меняет знак.

Преимущества метода:

- Гарантированная сходимость. Метод всегда сходится, если функция непрерывна и знаки значений функции на концах отрезка различны.

- Простота реализации. Легко реализуется без необходимости вычисления производных.
- Работает для любых функций. Метод не требует гладкости функции и работает даже для разрывных функций (если изменение знака сохраняется).

Недостатки метода:

- Медленная сходимость. Количество итераций растет как $O(\log_n)$, что делает метод менее эффективным по сравнению с другими методами.
- Не использует информацию о форме функции. Метод не учитывает производные или скорость изменения функции, из-за чего может сходиться медленнее других численных методов.

Вывод для уравнения:

Метод бисекции гарантированно находит корень, но делает это медленнее других методов. Это надежный метод, но если важна скорость, лучше выбрать другой подход.

2. Метод хорд

Метод хорд использует линейную аппроксимацию функции, соединяя две точки графика прямой и находя пересечение этой прямой с осью x.

Преимущества метода:

- Более быстрая сходимость, чем у метода бисекции. При хороших начальных приближениях метод сходится значительно быстрее, чем метод деления пополам.
- Не требует вычисления производных. Это делает его удобным для работы с функциями, у которых сложно найти аналитическую производную.
- Использует дополнительную информацию о функции. Метод учитывает направление изменения функции, что делает его эффективнее.

Недостатки метода:

- Не всегда сходится. Если функция имеет перегиб или если точки выбираются неудачно, метод может колебаться и расходиться.
- Чувствителен к выбору начальных точек. Плохой выбор может привести к медленной сходимости или отсутствию сходимости.

Вывод для уравнения:

Метод хорд показывает хорошую скорость сходимости и подходит для решения данного уравнения, если правильно выбрать начальные точки.

3. Метод Ньютона

Метод Ньютона использует касательные к функции и находит их пересечения с осью x.

Преимущества метода:

- Очень быстрая сходимость. При хороших начальных значениях сходимость метода составляет $O(\log \log_n)$, что делает его одним из самых быстрых численных методов.
- Использует информацию о производной. Метод учитывает скорость изменения функции, что позволяет быстрее достигать корня.

Недостатки метода:

- Требует вычисления производной. Если аналитически получить производную сложно или функция плохо дифференцируема, метод может быть неудобен.
- Не всегда сходится. Если начальная точка выбрана плохо, метод может расходиться или застрять в локальном экстремуме.
- Не работает, если производная близка к нулю. Если $f'(x) \approx 0$, метод Ньютона теряет эффективность и может давать большие ошибки.

Вывод для уравнения:

Метод Ньютона отлично подходит для решения уравнения, так как функция хорошо дифференцируема. Этот метод самый быстрый из всех и при правильном выборе начальной точки даст решение быстрее остальных.

4. Метод простой итерации

Метод простой итерации основывается на преобразовании уравнения $f(x) = 0$ в эквивалентное $x = g(x)$ и последующем вычислении последовательности значений.

Преимущества метода:

- Простота реализации. Метод не требует вычисления производных и легко программируется.
- Подходит для широкого класса уравнений. Если можно подобрать хорошую функцию $g(x)$, метод может работать стабильно.

Недостатки метода:

- Медленная сходимость. Метод сходится только при выполнении условия Липшица $|g'(x)| < 1$, и даже при выполнении этого условия скорость сходимости может быть медленной.
- Требует правильного выбора функции $g(x)$. Если выбрать неподходящую функцию, метод может вообще не сойтись.
- Чувствителен к начальному приближению. Если начальная точка выбрана плохо, метод может расходиться.

Вывод для уравнения:

Метод простой итерации является самым медленным и наименее надежным из всех. Он может работать, но требует подбора хорошей функции $g(x)$, что делает его менее удобным.

Какой метод лучше всего подходит для уравнения?

Метод Ньютона является наиболее эффективным для данного уравнения, так как:

- Функция хорошо дифференцируема.
- Производная не обращается в ноль на рассматриваемом интервале.
- Метод сходится быстрее всех остальных.

Если вычисление производной затруднительно или требуется более универсальный метод, можно использовать метод хорд. Метод бисекции подойдет, если нужна 100% гарантия нахождения корня, но он будет медленнее. Метод простой итерации наименее надежен и эффективен.

4. Реализация уравнения в программе

```
import numpy as np

def f(x):

    """Функция, корень которой ищем."""

    return 1.5 - 0.4 * np.sqrt(x**3) - np.exp(-x) * np.sin(x)

def df(x):

    """Производная функции f(x) для метода Ньютона."""

    return (-0.6 * np.sqrt(x)) - np.exp(-x) * (np.sin(x) - np.cos(x))

def bisection_method(a, b, tol=1e-6):

    """Метод бисекции (дихотомии)."""

    iterations = []

    while abs(b - a) > tol:

        c = (a + b) / 2

        iterations.append((round(a, 6), round(b, 6), round(c, 6), round(f(c), 6)))

        if len(iterations) == 3:

            break  # Ограничиваем вывод только 3 итерациями для анализа

        if f(a) * f(c) < 0:

            b = c

        else:

            a = c

    return round(c, 6), iterations
```

```
def secant_method(x0, x1, tol=1e-6):
    """Метод хорд (секущих)."""
    iterations = []
    while abs(x1 - x0) > tol:
        x2 = x1 - f(x1) * (x1 - x0) / (f(x1) - f(x0))
        iterations.append((round(x1, 6), round(x0, 6),
round(f(x1), 6), round(x2, 6)))
        if len(iterations) == 3:
            break
        x0, x1 = x1, x2
    return round(x1, 6), iterations
def newton_method(x0, tol=1e-6):
    """Метод Ньютона."""
    iterations = []
    while True:
        x1 = x0 - f(x0) / df(x0)
        iterations.append((round(x0, 6), round(f(x0), 6),
round(df(x0), 6), round(x1, 6)))
        if len(iterations) == 3:
            break
        if abs(x1 - x0) < tol:
            break
        x0 = x1
```

```
    return round(x1, 6), iterations

def g(x):

    """Функция для метода простой итерации."""

    return 1.5 - 0.4 * np.sqrt(x**3) - np.exp(-x) * np.sin(x)
+ x

def simple_iteration_method(x0, tol=1e-6):

    """Метод простой итерации."""

    iterations = []

    while True:

        x1 = g(x0)

        iterations.append((round(x0, 6), round(x1, 6)))

        if len(iterations) == 3:

            break

        if abs(x1 - x0) < tol:

            break

        x0 = x1

    return round(x1, 6), iterations


# Запуск всех методов

x_bisect, bisect_iterations = bisection_method(2.3, 2.4)

x_secant, secant_iterations = secant_method(2.3, 2.4)

x_newton, newton_iterations = newton_method(2.35)

x_iter, iter_iterations = simple_iteration_method(2.35)
```

```
# Вывод результатов
print("Метод бисекции:", x_bisect)
print("Метод хорд:", x_secant)
print("Метод Ньютона:", x_newton)
print("Метод простой итерации:", x_iter)

# Вывод первых 3 итераций
print("\nПервые 3 итерации метода бисекции:",
bisect_iterations)
print("Первые 3 итерации метода хорд:", secant_iterations)
print("Первые 3 итерации метода Ньютона:", newton_iterations)
print("Первые 3 итерации метода простой итерации:",
iter_iterations)
```

Я написал код на Python, который вычисляет корни уравнения с использованием всех четырех методов: бисекции, хорд, Ньютона и простой итерации.

Код также сохраняет и выводит первые три итерации для каждого метода.

Объяснение кода:
1. Функция `f(x)` – задает исходную функцию, корень которой ищем.
2. Функция `df(x)` – вычисляет производную функции f(x), необходимую для метода Ньютона.
3. Метод `bisection_method(a, b, tol)` – реализует метод бисекции: - Вычисляет середину интервала. - Проверяет знак функции и выбирает новую границу. - Повторяет процесс, пока не достигнет заданной точности.
4. Метод `secant_method(x0, x1, tol)` – метод хорд:
    - Использует два начальных приближения.
    - Обновляет x согласно формуле метода.

- Повторяет до достижения точности.

5. Метод `newton_method(x0, tol)` – метод Ньютона:
   - Использует начальное приближение.
   - Обновляет x по формуле Ньютона.
   - Повторяет до достижения точности.
6. Функция `g(x)` – задает итерационную функцию для метода простой итерации.
7. Метод `simple_iteration_method(x0, tol)` – метод простой итерации:
   - Обновляет x с использованием функции g(x).
   - Повторяет, пока разница между шагами не станет меньше точности.

После выполнения, код:

Выводит найденные корни для каждого метода.

Показывает первые 3 итерации каждого метода для проверки.

Результат вывода результата на экран:

```
Метод бисекции: 2.3375
Метод хорд: 2.338698
Метод Ньютона: 2.338899
Метод простой итерации: 2.338821

Первые 3 итерации метода бисекции: [(2.3, 2.4, 2.35, -0.008845), (2.3, 2.35, 2.325, 0.010674), (2.325, 2.35, 2.3375, 0.00094)]
Первые 3 итерации метода хорд: [(2.4, 2.3, -0.048502, 2.338206), (2.338206, 2.4, 0.000389, 2.338698), (2.338698, 2.338206, 5e-06, 2.338704)]
Первые 3 итерации метода Ньютона: [(2.35, -0.008845, -1.054652, 2.341613), (2.341613, -0.002274, -1.054134, 2.339456), (2.339456, -0.000587, -1.054, 2.338899)]
Первые 3 итерации метода простой итерации: [(2.35, 2.341155), (2.341155, 2.339239), (2.339239, 2.338821)]
```

```
Метод бисекции: 2.3375
Метод хорд: 2.338698
Метод Ньютона: 2.338899
Метод простой итерации: 2.338821
```