**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ ИМЕНИ ПАТРИСА ЛУМУМБЫ**

**Факультет физико-математических и естественных наук**

**Кафедра «Математического моделирования и искусственного интеллекта»**

Компьютерный практикум

Лабораторная работа №1

**«Безусловная одномерная оптимизация методами дихотомии и золотого сечения и работа с комплексными числами»**

|  |  |
| --- | --- |
| Студент | Игнатова Анастасия Александровна |
| Группа | НБИбд-01-23 |
|  |  |

**Москва**

**20****24**

**1. Выполнение 1-го задания на C++:**  
Я написала этот код на C++, чтобы изучить работу с комплексными числами. В нём я использовала стандартную библиотеку <complex>, которая позволяет удобно работать с такими числами, состоящими из действительной и мнимой частей.

Сначала я определила несколько комплексных чисел a, b и d, чтобы показать различные операции, такие как сложение, вычитание, умножение и деление комплексных чисел. Для вывода результатов операций на экран я создала функцию operator<<, которая форматирует вывод комплексного числа в человекочитаемый вид типа a + bi.

Также я вычислила четвертую степень комплексного числа d и нашла его модуль и аргумент. Это помогло мне вычислить корни третьей степени из числа d.

Вся программа написана так, чтобы выводить результаты на экран на русском языке, поэтому я использовала setlocale(LC\_ALL, "RUS"), чтобы установить русскую локаль.

Этот код помог мне лучше понять работу с комплексными числами и их математические свойства в программировании на C++.

**2. Выполнение 1-го задания на Python:**Я написала этот код на Python с использованием библиотеки cmath, чтобы изучить работу с комплексными числами. В нём я определила несколько комплексных чисел a, b и d, используя функцию complex(), которая позволяет задать комплексное число с указанием его действительной и мнимой частей.

Затем я выполнила различные математические операции над этими числами: сложение (summa), вычитание (raznost), умножение (umnozhenie), деление (delenie) комплексных чисел a и b. Также я возвела комплексное число d в четвертую степень (stepen4).

Для вычисления модуля и аргумента комплексного числа d я использовала функции abs() и cmath.phase(). Это позволило мне вычислить корни третьей степени из числа d с помощью функции cmath.rect().

Результаты всех операций я вывела на экран с помощью функции print(), чтобы показать результаты вычислений на русском языке.

Этот код помог мне лучше понять и изучить работу с комплексными числами и их математические операции в Python с использованием библиотеки cmath.

**3. Выполнение 2-го задания на C++:**Этот код на C++ написан мной для решения уравнений различными численными методами. Я использовала стандартные библиотеки <iostream> для ввода-вывода и <cmath> для математических функций.

Я реализовала несколько методов:

1. Метод бисекции (bisection):

Я выбрала этот метод, потому что он прост в понимании и обеспечивает сходимость к корню на интервале [a, b]. Он делит интервал пополам и проверяет изменение знака функции f(x) между a и b, чтобы определить, в какой половине интервала находится корень. Я установила точность epsilon, чтобы остановить итерации, когда достигнута нужная точность.

2. Метод секущих (secant):

Этот метод также прост в понимании, поэтому я его включила. Он использует две начальные точки x0 и x1, чтобы приблизиться к корню, вычисляя точку пересечения касательной, проведенной через эти точки, с осью x. Я использовала условие fabs(x1 - x0) > epsilon для остановки итераций при достижении нужной точности.

3. Метод Ньютона (newton):

Этот метод я выбрала из-за его высокой скорости сходимости. Он использует текущее приближение x0 и производную функции df(x0), чтобы найти следующее приближение x1. Я написала цикл while, который продолжается до тех пор, пока разница между x1 и x0 не станет меньше epsilon.

4. Метод простой итерации (iteration):

Этот метод я использовала из-за его простоты и эффективности в определенных случаях. Он преобразует уравнение до формы, в которой x находится в одной сторонеот равенства. Я написала цикл while, который продолжается до достижения нужной точности.

Каждый из этих методов был написан мной с целью понять, как численные методы работают в реальных задачах, и проверить их эффективность при решении уравнений.

**4. Выполнение 2-го задания на Python:**

Я написала этот код на Python для решения уравнений различными численными методами. В коде используются методы: бисекции, секущих, Ньютона и простых итераций. Каждый метод написан для приближенного нахождения корней уравнения с заданной точностью epsilon.

Метод бисекции (bisection):

Начинает с интервала [a, b] и делит его пополам до тех пор, пока разница между a и b не станет меньше epsilon.

Метод секущих (secant):

Использует две начальные точки x0 и x1, чтобы приблизиться к корню уравнения, вычисляя точку пересечения касательной между ними.

Метод Ньютона (newton):

Использует текущее приближение x0 и производную функции df(x0), чтобы на каждой итерации приближаться к корню с учетом скорости сходимости.

Метод простых итераций (iteration):

Преобразует уравнение до формы, в которой x находится в одной стороне от равенства, и итеративно улучшает приближение к корню.

Я использовала стандартную библиотеку math для математических операций и функций, что позволило мне реализовать и проверить работу этих методов в практических задачах.

# Заключение.

В данной работе был показаны такие алгоритмы поиска минимума функции на отрезке, как метод дихотомии и метод золотого сечения. Как мы могли убедиться методу золотого сечения нужно на порядок больше итераций, чем методу дихотомии.

# Литература

1. <http://bpascal.ru/download/desc/319.php><http://www.mccme.ru/ium/ancient/combs93.html>
2. <http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B7%D0%BE%D0%BB%D0%BE%D1%82%D0%BE%D0%B3%D0%BE_%D1%81%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F>
3. <http://ssalab.ru/stud/numpractice/>