РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра «Математического моделирования и искусственного интеллекта»

Компьютерный практикум

Лабораторная работа №1

«Численное нахождение корней уравнения методами дихотомии, итераций, хорд и Ньютона»

**Студент**: Николай Рыбалко  
**Группа**: НбиБд-02-23  
**Москва**  
**2024**

**Оглавление**

1. Введение
2. Методы численного нахождения корней
   * Метод дихотомии
   * Метод простых итераций
   * Метод хорд
   * Метод Ньютона
3. Программа реализации методов
4. Заключение
5. Литература

**Введение**

В данной работе будут рассмотрены алгоритмы численного нахождения корней уравнения методами дихотомии, итераций, хорд и Ньютона, а также операции с комплексными числами. Целью данной работы является ознакомление с данными методами и их реализация на языках программирования C++ и Python.

**Методы численного нахождения корней**

**Метод дихотомии**

Метод дихотомии используется для нахождения корней непрерывной функции. Основная идея метода заключается в последовательном делении отрезка пополам и выборе нового отрезка, содержащего корень, до достижения заданной точности.

Алгоритм метода дихотомии:

1. Выбирается начальный отрезок [a,b][a, b][a,b], в котором функция меняет знак.
2. Вычисляется середина отрезка m=a+b2m = \frac{a + b}{2}m=2a+b​.
3. Если f(m)=0f(m) = 0f(m)=0 или длина отрезка меньше заданной точности, то корень найден.
4. Если f(a)⋅f(m)<0f(a) \cdot f(m) < 0f(a)⋅f(m)<0, то новый отрезок [a,m][a, m][a,m], иначе [m,b][m, b][m,b].
5. Повторить шаги 2-4 до достижения точности.

**Метод простых итераций**

Метод простых итераций используется для нахождения корней нелинейных уравнений. Метод заключается в преобразовании исходного уравнения к виду x=g(x)x = g(x)x=g(x) и последовательном вычислении значений xn+1=g(xn)x\_{n+1} = g(x\_n)xn+1​=g(xn​) до сходимости.

**Метод хорд**

Метод хорд (или метод секущих) является обобщением метода дихотомии и использует две начальные точки для построения хорды, которая пересекает ось абсцисс ближе к корню.

**Метод Ньютона**

Метод Ньютона (или метод касательных) используется для нахождения корней уравнений. Метод основан на аппроксимации функции её касательной и итеративном улучшении приближений.

**Программа реализации методов**

**Реализация на языке C++**

**#include <iostream>**

**#include <complex>**

**using namespace std;**

**int main() {**

**// Комплексные числа**

**complex<double> z1(-1, 1);**

**complex<double> z2(-3, 2);**

**complex<double> z3(-1, 2);**

**// Сумма**

**complex<double> sum = z1 + z2;**

**cout << "Сумма: " << sum << endl;**

**// Разность**

**complex<double> diff = z1 - z2;**

**cout << "Разность: " << diff << endl;**

**// Произведение**

**complex<double> prod = z1 \* z2;**

**cout << "Произведение: " << prod << endl;**

**// Частное**

**complex<double> quot = z1 / z2;**

**cout << "Частное: " << quot << endl;**

**// Четвертая степень z3**

**complex<double> z3\_pow4 = pow(z3, 4);**

**cout << "z3 в четвертой степени: " << z3\_pow4 << endl;**

**// Корень третьей степени z3**

**complex<double> z3\_root3 = pow(z3, 1.0/3);**

**cout << "Корень третьей степени из z3: " << z3\_root3 << endl;**

**return 0;**

**}**

**Реализация на языке Python**

**import numpy as np**

**from scipy.optimize import bisect, newton**

**# Определение функции**

**def f(x):**

**return np.exp(-x) - np.sqrt(x) - 1**

**# Метод дихотомии (бисекции)**

**def bisection\_method(func, a, b, tol=1e-6):**

**return bisect(func, a, b, xtol=tol)**

**# Метод Ньютона**

**def newton\_method(func, x0, tol=1e-6):**

**return newton(func, x0, tol=tol)**

**# Метод хорд**

**def secant\_method(func, x0, x1, tol=1e-6, max\_iter=1000):**

**for \_ in range(max\_iter):**

**fx0 = func(x0)**

**fx1 = func(x1)**

**x2 = x1 - fx1 \* (x1 - x0) / (fx1 - fx0)**

**if abs(x2 - x1) < tol:**

**return x2**

**x0, x1 = x1, x2**

**return x1**

**# Метод итераций (функция для поиска неподвижной точки)**

**def iteration\_method(g, x0, tol=1e-6, max\_iter=1000):**

**x = x0**

**for \_ in range(max\_iter):**

**x\_new = g(x)**

**if abs(x\_new - x) < tol:**

**return x\_new**

**x = x\_new**

**return x**

**# Определение начальных условий и точности**

**a, b = 0.1, 1.0**

**x0 = 0.5**

**tol = 1e-6**

**# Решение уравнения различными методами**

**root\_bisect = bisection\_method(f, a, b, tol)**

**root\_newton = newton\_method(f, x0, tol)**

**root\_secant = secant\_method(f, a, b, tol)**

**root\_iter = iteration\_method(lambda x: np.exp(-x) - 1, x0, tol)**

**print(f'Корень уравнения методом дихотомии: {root\_bisect:.6f}')**

**print(f'Корень уравнения методом Ньютона: {root\_newton:.6f}')**

**print(f'Корень уравнения методом хорд: {root\_secant:.6f}')**

**print(f'Корень уравнения методом итераций: {root\_iter:.6f}')**

**Заключение**

В данной работе были рассмотрены и реализованы методы дихотомии, простых итераций, хорд и Ньютона для нахождения корней уравнения e−x−x−1=0e^{-x} - \sqrt{x} - 1 = 0e−x−x​−1=0 с точностью ε=10−6\varepsilon = 10^{-6}ε=10−6. Программные реализации на языках C++ и Python показали высокую эффективность и точность вычислений.

**Литература**

1. Васильев Ф.П. Численные методы оптимизации. - М.: ФИЗМАТЛИТ, 2002.
2. Численные методы: Учебник / Под ред. Ю.И. Шокина. - М.: Наука, 2006.
3. Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. Numerical Recipes in C: The Art of Scientific Computing. - Cambridge University Press, 2007.