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

Межинский Павел Андреевич

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. 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], в котором функция меняет знак.

2. Вычисляется середина отрезка \( m = \frac{a + b}{2} \).

3. Если \( f(m) = 0 \) или длина отрезка меньше заданной точности, то корень найден.

4. Если \( f(a) \cdot f(m) < 0 \), то новый отрезок [a, m], иначе [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 \( x = g(x) \) и последовательном вычислении значений \( x\_{n+1} = g(x\_n) \) 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 <cmath>

#include <functional>

// Функция для нахождения корней

double f(double x) {

return log(x) + pow(x + 1, 3);

}

// Метод дихотомии

double bisectionMethod(double a, double b, double epsilon) {

double mid;

while ((b - a) >= epsilon) {

mid = (a + b) / 2;

if (f(mid) == 0.0)

break;

else if (f(mid) \* f(a) < 0)

b = mid;

else

a = mid;

}

return mid;

}

// Метод простых итераций

double iterationMethod(double x0, double epsilon) {

double x1;

do {

x1 = x0;

x0 = -pow(x1 + 1, 3); // Переписанное уравнение x = -pow(x+1,3)

} while (fabs(x1 - x0) >= epsilon);

return x0;

}

// Метод хорд

double secantMethod(double a, double b, double epsilon) {

double c;

while (fabs(b - a) >= epsilon) {

c = b - (f(b) \* (b - a)) / (f(b) - f(a));

a = b;

b = c;

}

return c;

}

// Метод Ньютона

double newtonMethod(double x0, double epsilon) {

double x1;

while (true) {

x1 = x0 - f(x0) / (1/x0 + 3\*pow(x0+1, 2));

if (fabs(x1 - x0) < epsilon)

break;

x0 = x1;

}

return x1;

}

int main() {

double a = 0.1, b = 1.0; // Интервал для методов дихотомии и хорд

double x0 = 0.5; // Начальное приближение для методов итераций и Ньютона

double epsilon = 1e-6;

std::cout << "Метод дихотомии: " << bisectionMethod(a, b, epsilon) << std::endl;

std::cout << "Метод простых итераций: " << iterationMethod(x0, epsilon) << std::endl;

std::cout << "Метод хорд: " << secantMethod(a, b, epsilon) << std::endl;

std::cout << "Метод Ньютона: " << newtonMethod(x0, epsilon) << std::endl;

return 0;

}

```

***Реализация на языке Python***

import math

# Функция для нахождения корней

def f(x):

return math.log(x) + (x + 1) \*\* 3

# Метод дихотомии

def bisection\_method(a, b, epsilon):

while (b - a) / 2.0 > epsilon:

midpoint = (a + b) / 2.0

if f(midpoint) == 0:

return midpoint

elif f(a) \* f(midpoint) < 0:

b = midpoint

else:

a = midpoint

return (a + b) / 2.0

# Метод простых итераций

def iteration\_method(x0, epsilon):

x1 = x0

while True:

x0 = x1

x1 = -((x0 + 1) \*\* 3)

if abs(x1 - x0) < epsilon:

break

return x1

# Метод хорд

def secant\_method(a, b, epsilon):

while abs(b - a) > epsilon:

c = b - (f(b) \* (b - a)) / (f(b) - f(a))

a = b

b = c

return c

# Метод Ньютона

def newton\_method(x0, epsilon):

while True:

x1 = x0 - f(x0) / (1/x0 + 3 \* (x0 + 1) \*\* 2)

if abs(x1 - x0) < epsilon:

return x1

x0 = x1

# Основная программа

a, b = 0.1, 1.0 # Интервал для методов дихотомии и хорд

x0 = 0.5 # Начальное приближение для методов итераций и Ньютона

epsilon = 1e-6

print("Метод дихотомии:", bisection\_method(a, b, epsilon))

print("Метод простых итераций:", iteration\_method(x0, epsilon))

print("Метод хорд:", secant\_method(a, b, epsilon))

print("Метод Ньютона:", newton\_method(x0, epsilon))

```

Заключение

В данной работе были рассмотрены и реализованы методы дихотомии, простых итераций, хорд и Ньютона для нахождения корней уравнения \( \ln(x) + (x + 1)^3 = 0 \) с точностью \(\epsilon = 10^{-6}\). 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.