МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
**«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)**

Институт информационных технологий, математики и механики

Направление подготовки: «Прикладная математика и информатика»
Профиль подготовки: «Вычислительные методы и суперкомпьютерные технологии»

Отчет
по лабораторной работе №1
по курсу «Прикладная нелинейная динамика»

Выполнил:
студент группы 3823М1ПМвм
Бекетов Е.В.

Проверил:
д.ф.-м.н., доц., зав.каф. ПМ
Иванченко М.В.

Нижний Новгород
2024

1. Сравнение скорости сходимости методов Ньютона и дихотомии

Для начала найдем корень следующего полинома $f(x) = x^{n+1} + x - \alpha$. Так как он записан в общем виде, необходимо взять какой-то его частный вид, например при $n = 2$ и $\alpha = 1$. Для нахождения корня $f(x) = x^3 + x - 1$ были использованы два метода: дихотомия и Ньютона. Как известно из теории первый метод обладает линейной сходимостью, а второй – квадратичной (т.к. в данном уравнении производная $f(x)$ нигде не обращается в 0). Начальным условием для метода Ньютона была выбрана точка $x_0 = 0$, а для дихотомии начальный отрезок $[-1; 1]$. Точность обоих методов: 10^{-6}.

Посмотрим результаты численных методов. Было получено приближённое значение корня $\widetilde{x} = 0.6823$ (значение округлено), а из Рис. 1 видно, что метод Ньютона сходится за 6 итерации, в то время как дихотомии необходимо 21 итерация для достижения той же точности. Что согласуется с теорией о медленной сходимости дихотомии в сравнении с Ньютоном.

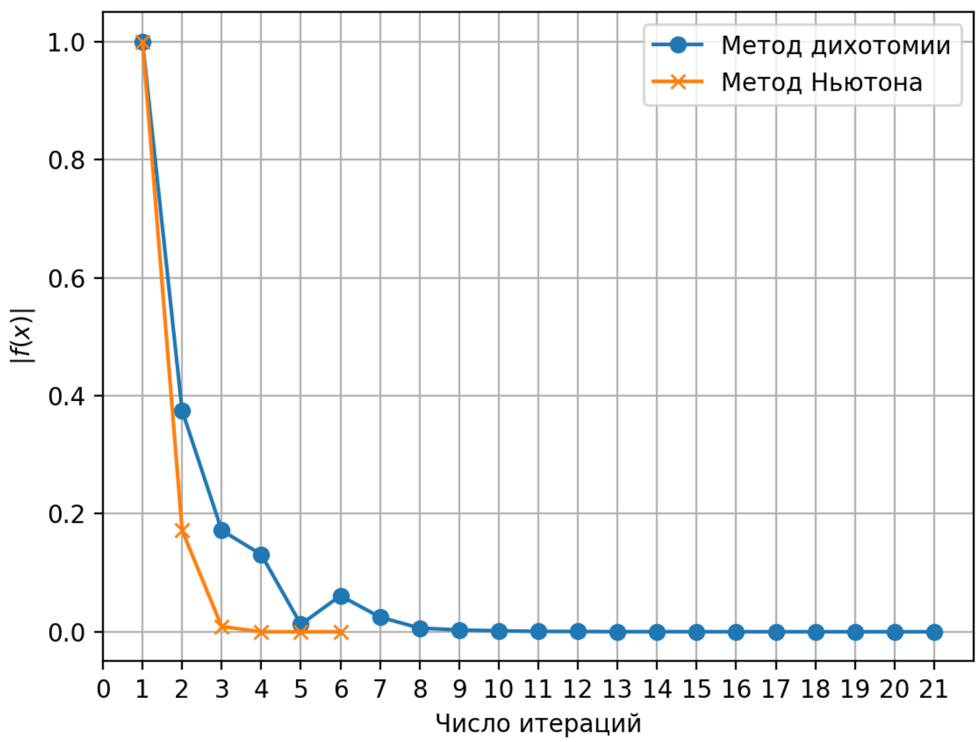

Рис. 1: Зависимость абсолютного значения полинома от номера итерации метода.

2. Построение зависимости корня полинома от параметра α

Выше уже был найден один корень для частного случая, однако при $\alpha > 0$ корней бесконечное множество. Найдем $x^*(\alpha)$ координату корня полинома $f(x) = x^{n+1} + x - \alpha$ от α при $n = 2, 4, 6$ на отрезке $[0; 10]$.

Из качественного анализа положения корня следует, что при достаточно малых α зависимость схожа с линейной, а при больших близка к функции $\alpha^{\frac{1}{n+1}}$.

Реализуем алгоритм поиска и посмотрим на результаты Рис. 2.

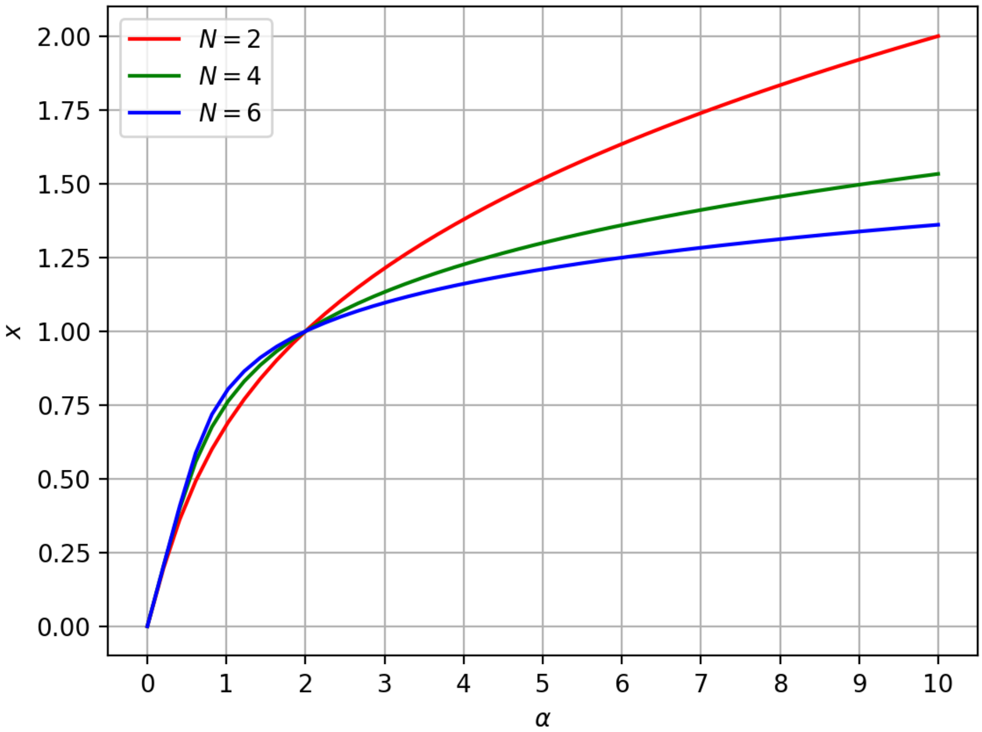

Рис. 2: Зависимость корня полинома $f(x)$ от параметра α при различных значениях N.

Собственно, что и предполагалось.

3. Вывод

В конечном итоге был написан скрипт на языке Python, для наглядной демонстрации скорости сходимости методов Ньютона и дихотомии, как и предполагалось, метод Ньютона быстрее достиг корня уравнения, чем дихотомия.

Вместе с тем, был продемонстрирован график всех возможных значений корней для разных $\alpha \in [0; 10]$ при $N = 2, 4, 6$. Практические наблюдения сходятся с теоретическими предположениями.

4. Приложение – Код программы

```
import numpy as np
import matplotlib.pyplot as plt

#! Глобальные переменные
eps = 0.000001

def f1(x):
    return x**3 + x - 1

def df1(x):
    return 3*x**2 + 1

def f2(x, n, alpha):
    return x**(n + 1) + x - alpha

def df2(x, n, alpha):
    return (n + 1)*x**n + 1

def dichotomy_method(left, right, f):
    x_path = []
    while right - left > eps:
        x_mid = (left + right) / 2
        if f(x_mid) == 0 or abs(f(x_mid)) < eps:
            return x_mid, x_path
        elif f(left)*f(x_mid) < 0:
            right = x_mid
        else:
            left = x_mid
        x_path.append(f(x_mid))
    return (left + left) / 2, x_path

def newton_method(x0, f, df):
    x_next = x0 - f(x0) / df(x0)
    x_path = [f(x_next)]
    while abs(x0 - x_next) > eps:
        x0, x_next = x_next, x0
        x_next = x0 - f(x0) / df(x0)
        x_path.append(f(x_next))
    return x_next, x_path

def main():
    x_opt, path_dichotomy = dichotomy_method(-1, 1, f1)
    print('Метод дихотомии = {}'.format(x_opt))
    x_opt, path_newton = newton_method(0, f1, df1)
    print('Метод Ньютона = {}'.format(x_opt))

    plt.plot(range(1, len(path_dichotomy) + 1), \
             np.abs(path_dichotomy), '-o', label='Метод дихотомии')
```

```
    plt.plot(range(1, len(path_newton) + 1), \
             np.abs(path_newton), '-x', label='Метод Ньютона')
    plt.xlabel('Число итераций')
    plt.ylabel('$|f(x)|$')
    plt.xticks(np.arange(0, len(path_dichotomy) + 1, 1))
    plt.grid()
    plt.legend(loc = 'best', fontsize = 10)
    plt.savefig('Лаб1_корень.png', dpi = 200)


    alphaGrid = np.linspace(0, 10)
    colors = ['r', 'g', 'b']
    plt.clf()
    plt.xlabel('$\\alpha$')
    plt.ylabel('$x$')

    for i in range(1, 4):
        roots = [newton_method(alpha, lambda x: f2(x, i*2, alpha), \
                               lambda x: df2(x, i*2, alpha))[0] \
                               for alpha in alphaGrid]
        plt.plot(alphaGrid, roots, colors[i - 1], \
                 label = '$ N = ' + str(i*2) + '$')
    plt.xticks(np.arange(0, 11, 1))
    plt.grid()
    plt.legend(loc = 'best', fontsize = 10)
    plt.savefig('Лаб1_корни.png', dpi = 200)


if __name__ == '__main__':
    main()
```