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 №2
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. Построение бифуркационной диаграммы

Рассматривается система авторепрессора с задержкой:

$$\dot{x} = \frac{\alpha}{1 + x^n(t-\tau)} - x$$

Для этой дифференциального уравнения необходимо построить бифуркационную диаграмму в плоскости (α, τ). Количество и положение состояний равновесия авторепрессора с задержкой не отличаются от системы без задержки. В свою очередь комбинация параметров влияет лишь на устойчивость состояния равновесия.

Бифуркационное условие определяется следующими уравнениями:

$$\begin{cases} x_0^{n+1} + x_0 - \alpha = 0 \\ \omega^2 = n^2(1 - \frac{x_0}{\alpha})^2 - 1 \\ cos(\omega\tau) = -\frac{1}{n(1-\frac{x_0}{\alpha})} \end{cases}$$

Задав значение α и решив первое уравнение, можно найти ω из второго, затем τ находим из третьего уравнения по формуле:

$$\tau = \frac{1}{\omega} arccos(-\frac{1}{n(1 - \frac{x_0}{\alpha})})$$

На Рис. 1 представленны кривые разделяющие области устойчивости и неустойчивости состояния равновесия для $N = 2, 4, 6$, для разных α, τ. Если выбрать такую комбинацию α, τ, что точка будет выше кривой, то система будет находится в режиме автоколебаний, а если она будет ниже, то авторепроцессор с задержкой будет иметь устойчивое состояние равновесия.

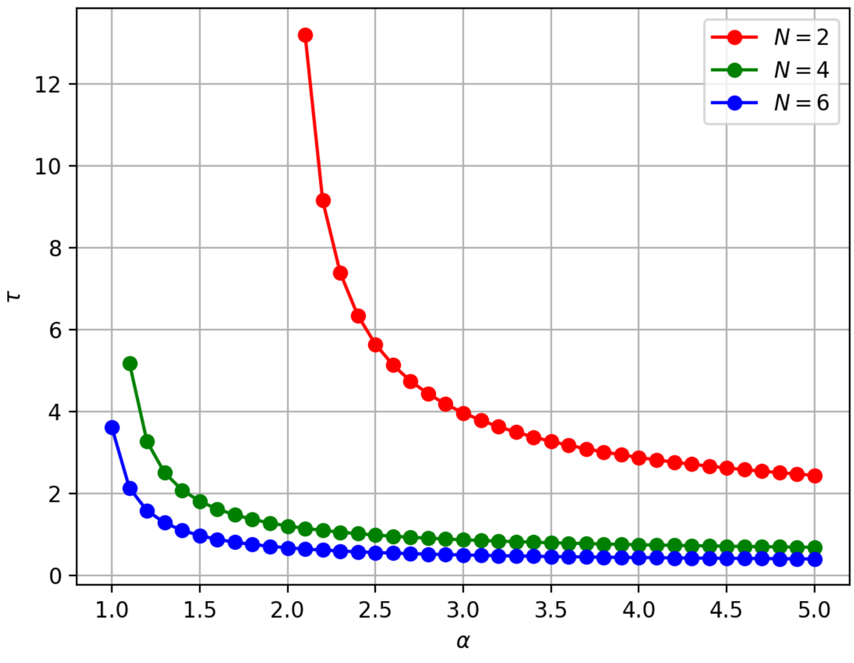

Рис. 1: Бифуркационная диаграмма системы при различных значениях N.

2. Приложение – Код программы

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

eps = 0.001

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

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

def main():
    alphaGrid = np.linspace(0.1, 5)
    colors = ['r', 'g', 'b']

    plt.xlabel('$\\alpha$')
    plt.ylabel('$\\tau$')

    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]
        w_values = []
        for j, x0 in enumerate(roots):
            rootArg = (i*2*(1 - x0 / alphaGrid[j]))**2 - 1.
            if rootArg > 0:
                w_values.append(np.sqrt(rootArg))
            else:
                w_values.append(np.nan)

        t_values = []
        for j, w in enumerate(w_values):
            acosArg = -1/((i*2)*(1 - roots[j] / alphaGrid[j]))
            if np.abs(acosArg) < 1:
                t_values.append(np.arccos(acosArg) \
                                / w if w is not np.nan else np.inf)
            else:
```

```
                t_values.append(np.nan)

        plt.plot(alphaGrid, t_values, colors[i - 1] + '-o', \
                 label = '$N = ' + str(i*2) + '$')

    plt.grid()
    plt.legend(loc = 'best', fontsize = 10)
    plt.savefig('lab2_biffur.png', dpi = 200)

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