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 №5
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 логистического отображения

Рассмотрим следующее отображение $x_{n+1} = rx_n(1 - x_n)$ оно называется логистическим. Из теории известно что при $r \in [0; 3]$ оно имеет единственную устойчивую неподвижную точку, при $r \geq 3$ бесконечно много бифуркаций удвоения периода.

Для построения бифуркационной диаграммы необходимо найти не только неподвижные точки, но и кратные значения. Для нахождения интересующих точек были сгенерированы 100 начальный точек в отрезке $[0; 1]$, для каждой из которых вычисляются 1600 итераций отображения. На Рис. 1 представлена получившаяся бифуркационная диаграмма, где пунктирной синей линией показана оценка r, после которой начинается хаос.

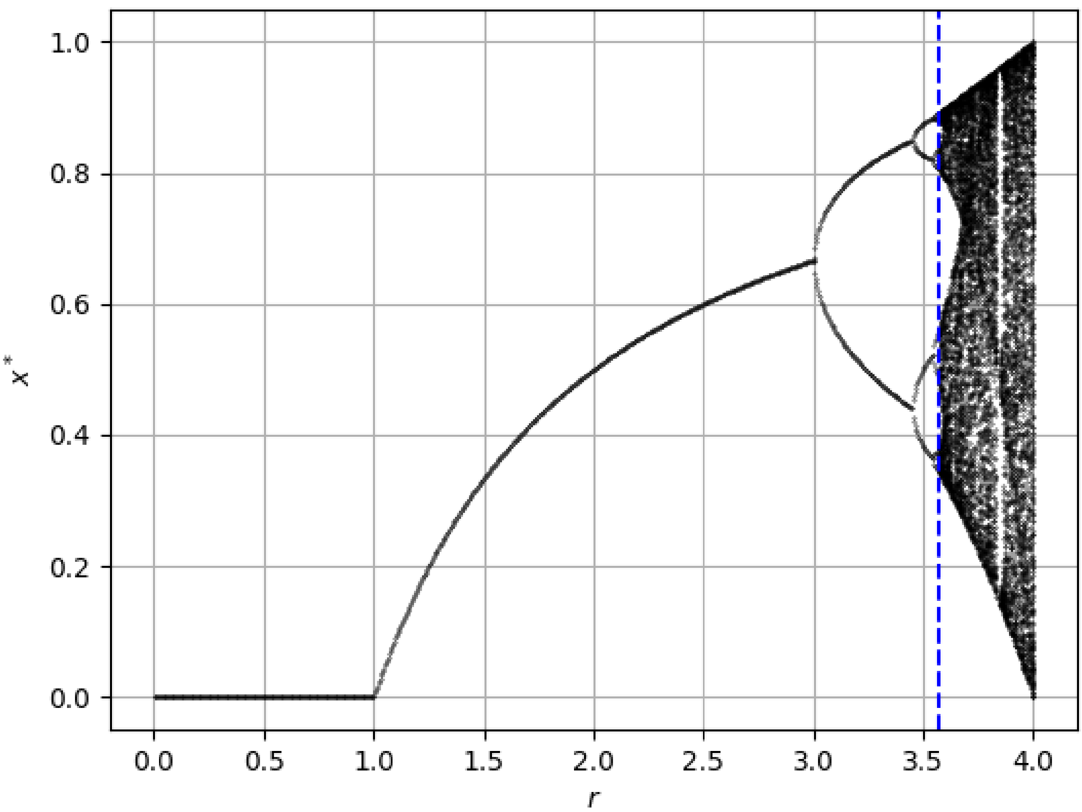

Рис. 1: Бифуркационная диаграмма логистического отображения.

2. Устойчивость логистического отображения с помощью Ляпунова

Устойчивость какой-либо траектории нашего отображения характеризуется показателем Ляпунова следующего вида:

$$\lambda = \frac{1}{N} ln|\prod_{k=1}^{N} f'(x_k)|$$

однако лучше использовать его другой вид:

$$\lambda = \frac{1}{N} \sum_{k=1}^{N} ln|f'(x_k)|$$

Известно, что при $\lambda < 0$ имеется устойчивость, при $\lambda > 0$ – неустойчивость, $\lambda = 0$ – переходное состояние. Видя Рис. 1 уже можно предсказать, что до перехода через $r \approx 3.57$ показатель будет отрицательный и в точках $r = 1, 3$ будет равен нулю. Посчитаем показатель и построим его значения в зависимости от r на Рис. 2

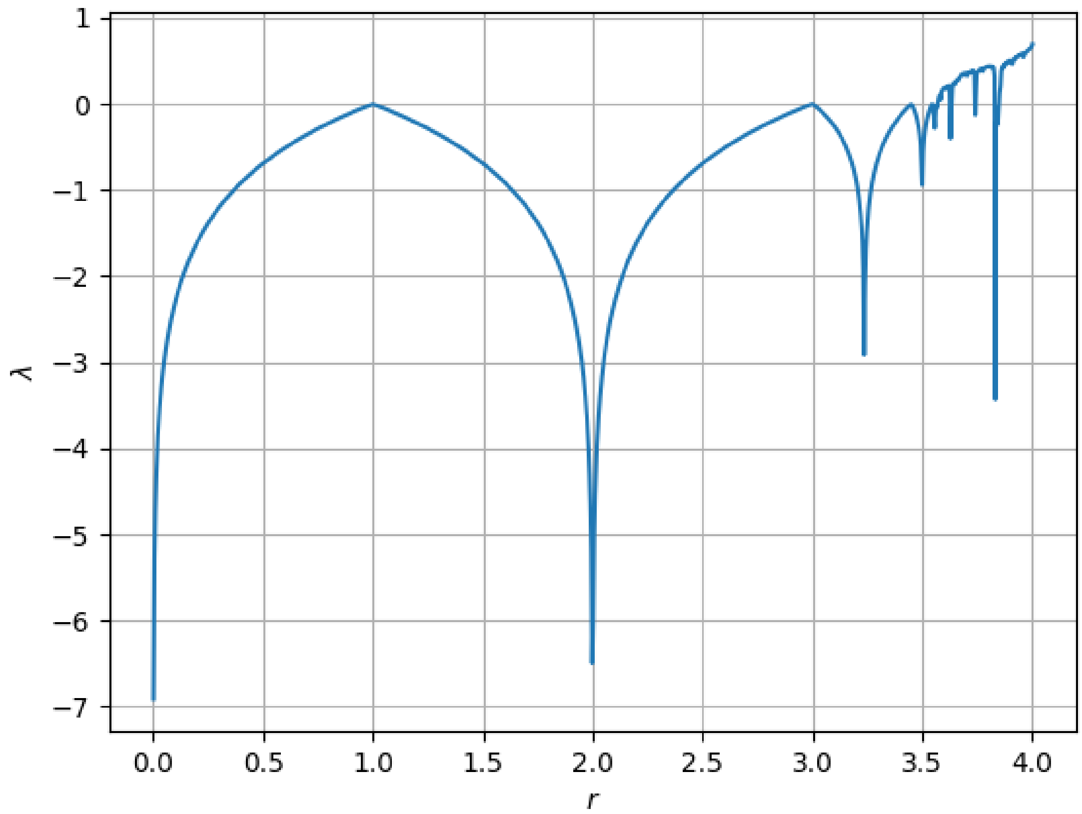

Рис. 2: Зависимость среднее показателя Ляпунова от параметра отображения r.

Видно, что сказанное выше согласуется с подсчитанными значениями. Как раз после $r \approx 3.57$ значение становится положительным, что говорит о пропаже устойчивости.

3. Приложение – Код программы

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

def delete_similar(array, value):
    sort_arr = []
    for x in array:
        if abs(x - value) > 0.001:
            sort_arr.append(x)
    return sort_arr

def comput_lapun(x_0, r, n):
    x = x_0
    lamd_lapun = 0
    for i in range(n):
        x = x*r*(1 - x)
        log = np.log(abs(r - 2*r*x))
        lamd_lapun += log
    return x, lamd_lapun / n

def sort_array(array):
    sort_arr = array
    for i in range(int(len(array) / 2 + 1)):
        sort_arr = delete_similar(sort_arr, array[i])
        sort_arr.append(array[i])
    return sort_arr

def main():
    #np.random.seed(100)
    n_iter = 1600
    impuls = 100
    r_list = np.linspace(1e-3, 4, 1000)
    x_start = np.random.rand(impuls)
    l_const = np.zeros((len(r_list), 1))
    x_finish = np.zeros((len(r_list), impuls))

    for j in range(impuls):
        for i, r in enumerate(r_list):
            x_end, lam = comput_lapun(x_start[j], r, n_iter)
            x_finish[i][j] = x_end
            l_const[i] += lam

    l_const /= impuls
    r_chaos = r_list[np.argmax(l_const >= 0.)]
    print('Граничное значение: r = {}'.format(r_chaos))

    plt.xlabel('$r$')
    plt.ylabel('$x^*$')
```

```
    for i, r in enumerate(r_list):
        tmp = np.array(sort_array(x_finish[i]))
        plt.plot([r], tmp.reshape((1, len(tmp))), 'o',
        markersize = 0.3, color='black')

    plt.axvline(x=r_chaos, color='b', linestyle='--')
    plt.grid()
    plt.savefig('bifur_diag.png')
    plt.clf()

    plt.xlabel('$r$')
    plt.ylabel(r'$\lambda$')
    plt.plot(r_list, l_const)
    plt.grid()
    plt.savefig('lapun_lamb.png')

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