**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Математическое моделирование

**Отчет по лабораторной работе № 4**

**Тема:** «Исследование эволюции нелинейной диссипативной динамической системы»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМ-457 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Исаева А.А. |  |  |  |
| Принял | Лукащук С.Ю. |  |  |  |

**Уфа 2024**

**Цель работы:** получить навык численного исследования динамики нелинейной диссипативной динамической системы, обладающей странным аттрактором.

**Задание на лабораторную работу**

Работа выполнена согласно варианту № 15.

Рассматривается нелинейная двухпараметрическая автономная динамическая система

Для заданной системы выполнить следующие задания.

1. Определить области изменения параметров *a* и *b*, в которых данная динамическая система является диссипативной.
2. Определить стационарные точки диссипативной системы.
3. Исследовать стационарные точки на асимптотическую устойчивость по первому приближению.
4. Определить значения параметров *a* и *b*, при которых в системе появляется странный аттрактор.
5. Написать вычислительную программу на языке программирования Cи++, реализующую процедуру численного интегрирования исходной диссипативной системы по методу Рунге-Кутта 4-го порядка точности.
6. С использованием вычислительной программы провести серию вычислительных экспериментов, демонстрирующих различные виды динамики системы. Построить траектории системы в окрестности стационарных точек. Определить численно значения параметров *a* и *b,* при которых в системе существует странный аттрактор и при которых система переходит в режим автоколебаний.

**Практическая часть**

Исходная система, согласно варианту 10:

**Область изменения параметров.**

Динамическая система является диссипативной, если где .

Исходная системы будет диссипативной при:

**Стационарные точки.**

Получим два случая:

.

Таким образом, получили 2 стационарные точки:

**Исследование стационарных точек на асимптотическую устойчивость по первому приближению.**

Выполним линеаризацию в окрестностях стационарных точек

малые возмущения.

Система после разложения в ряд Тейлора в стационарной точке

Тогда общий вид матрицы системы будет иметь вид:

Тогда

1. Сделаем оценку устойчивости точки :

Матрица будет иметь вид

Тогда

Матрица Гурвица для .

*–* всегда выполняется;

*=>*

*=>* b > 0

На рисунке 1 показано как эти ограничения накладываются друг на друга.

Рисунок 1 – Область значений параметров и , при которых точка устойчива.

.

**Определение значений параметров a и b, при которых в системе появляется странный аттрактор.**

Странный аттрактор возникает, когда все стационарные точки неустойчивы.

На рисунке 3 тремя черными линиями ограничена область, в которой все три стационарные точки неустойчивы.

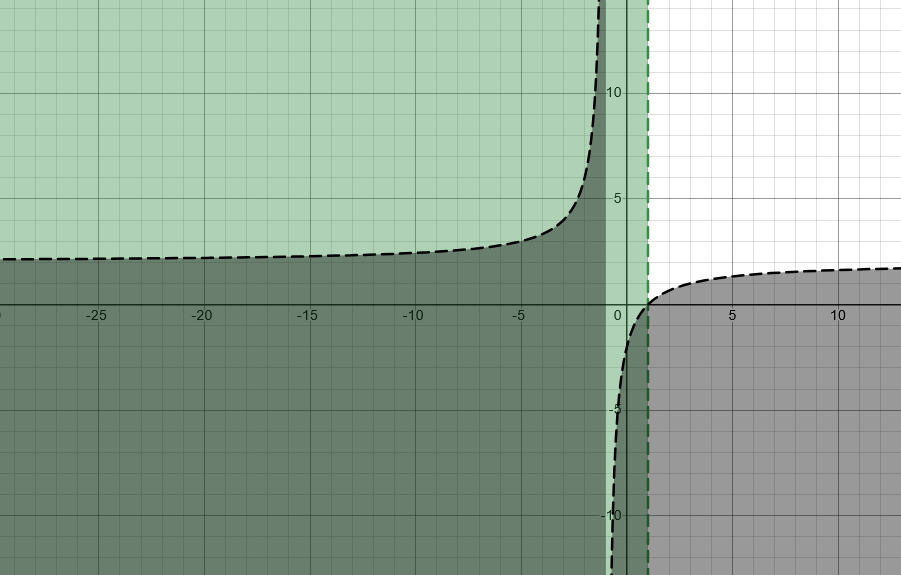
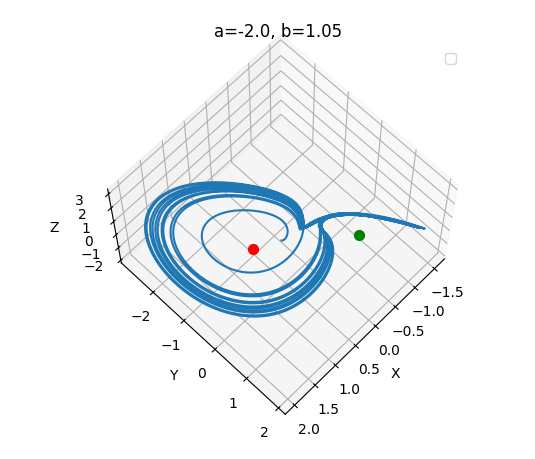
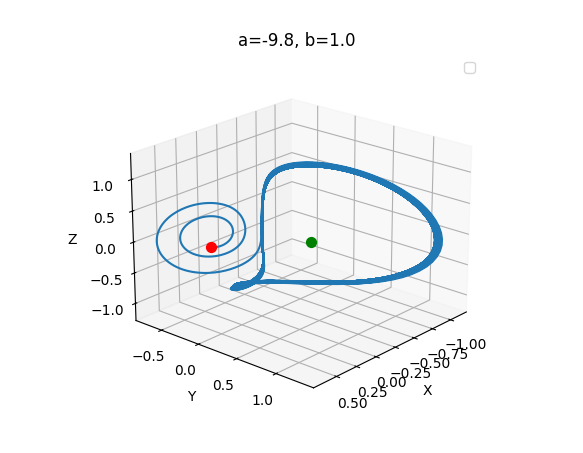
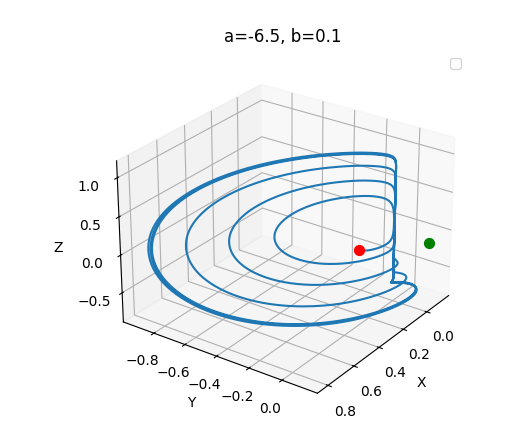
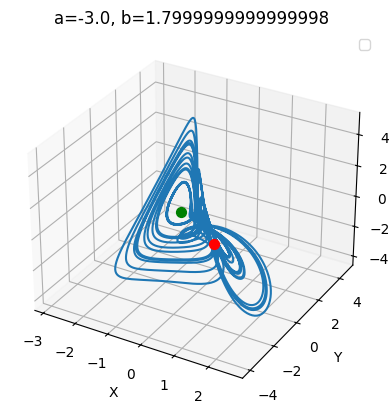
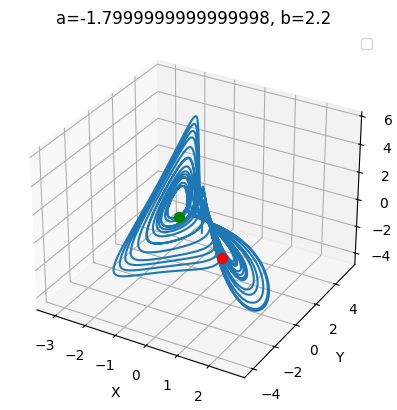


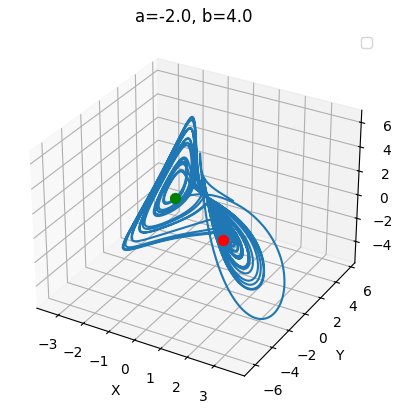
Рисунок 3 – Область значений параметров и , при которых все стационарные точки неустойчивы.

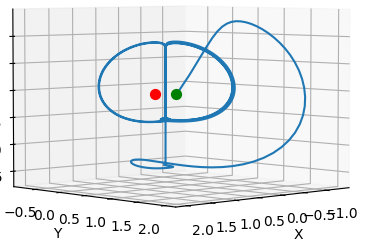
Для поиска странного аттрактора и автоколебаний будем численно перебирать параметры и , принадлежащие области из рисунка 3. Например, от -10. до 0, от 0 до 10.











**Вывод**

В данной лабораторной работе были получены навыки численного исследования динамики нелинейной диссипативной динамической системы, обладающей странным аттрактором. Были определены области изменения параметров a и b, в которых данная динамическая система является диссипативной, также были найдены параметры a и b при которых в системе существует странный аттрактор и при которых система переходит в режим автоколебаний.

**Приложение**

#include <iostream>

#include <vector>

#include <cmath>

#include <fstream>

const double a = -1.0;

const double b = -0.5;

std::vector<double> system(const std::vector<double>& state) {

std::vector<double> derivatives(3);

derivatives[0] = state[1];

derivatives[1] = a\*state[1] + (1- state[2])\* state[0];

derivatives[2] = b\*state[2] + state[0] \* state[0];

return derivatives;

}

std::vector<double> rungeKutta4(const std::vector<double>& state, double dt) {

std::vector<double> k1 = system(state);

std::vector<double> stateK2(3), stateK3(3), stateK4(3);

for (int i = 0; i < 3; ++i) stateK2[i] = state[i] + 0.5 \* dt \* k1[i];

std::vector<double> k2 = system(stateK2);

for (int i = 0; i < 3; ++i) stateK3[i] = state[i] + 0.5 \* dt \* k2[i];

std::vector<double> k3 = system(stateK3);

for (int i = 0; i < 3; ++i) stateK4[i] = state[i] + dt \* k3[i];

std::vector<double> k4 = system(stateK4);

std::vector<double> nextState(3);

for (int i = 0; i < 3; ++i) {

nextState[i] = state[i] + (dt / 6.0) \* (k1[i] + 2.0 \* k2[i] + 2.0 \* k3[i] + k4[i]);

}

return nextState;

}

int main() {

std::vector<double> state = { 1.0, 1.0, 1.0 };

double dt = 0.01;

int numSteps = 4000;

std::ofstream file("trajectory.txt");

for (int step = 0; step < numSteps; ++step) {

state = rungeKutta4(state, dt);

file << state[0] << " " << state[1] << " " << state[2] << "\n";

}

file.close();

std::cout << "Данные сохранены в файл trajectory.txt" << std::endl;

return 0;

}

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

def system(t, state, a, b):

    x, y, z = state

    dxdt = y

    dydt = a \* y + (1 - z) \* x

    dzdt = b \* z + x \* x

    return [dxdt, dydt, dzdt]

a\_vals = np.linspace(-1.15, 0, 10)

b\_vals = np.linspace(-0.9, 0, 10)

initial\_state = [0.1, 0.1, 0.1]

t\_span = (0, 100)

t\_eval = np.linspace(0, 100, 10000)

for a in a\_vals:

    for b in b\_vals:

            sol = solve\_ivp(system, t\_span, initial\_state, args=(a, b), t\_eval=t\_eval)

            x\_vals = sol.y[0]

            y\_vals = sol.y[1]

            z\_vals = sol.y[2]

            O1 = [0, 0, 0]

            O2 = [np.sqrt(-b), 0, 1]

            O3 = [-np.sqrt(-b), 0, 1]

            fig = plt.figure()

            ax = fig.add\_subplot(111, projection='3d')

            ax.plot(x\_vals, y\_vals, z\_vals, label=f'a={a}, b={b}')

            ax.set\_xlabel('X')

            ax.set\_ylabel('Y')

            ax.set\_zlabel('Z')

            ax.legend()

            ax.scatter(\*O1, color='red', s=50, label='O1 (0, 0, 0)')

            if not np.isnan(O2[1]):

                ax.scatter(\*O2, color='green', s=50, label=f'O2 (0, {np.sqrt(abs(b)):.2f}, {b})')

            if not np.isnan(O3[1]):

                ax.scatter(\*O3, color='blue', s=50, label=f'O3 (0, {-np.sqrt(abs(b)):.2f}, {b})')

            ax.set\_xlabel('X')

            ax.set\_ylabel('Y')

            ax.set\_zlabel('Z')

            plt.title(f'a={a}, b={b}')

            plt.show()