ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Дожность		Колесникова С.И
старший преподаватель	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5

Моделирование линейных/нелинейных объектов. Линейные системы. Передаточные функции. Модели детерминированного хаоса. Режимы устойчивости/неустойчивости. Автоколебательные модели.

по дисциплине: Компьютерное моделирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4236		Л. Мвале
	подпись, дата	инициалы, фамилия

Санкт-Петербург
2025

Цель работы

Цель настоящей работы: знакомство с элементами синергетического управления применительно к моделям детерминированного хаоса, с принципами организация обратных связей в сложных объектах для достижения режима устойчивости функционирования нелинейного объекта.

Ход работы .

Часть 1

1. Ознакомиться со справочными сведениями.

2. Построить графики и фазовые портреты нелинейной модели для устойчивого и неустойчивого режимов.

3. Разработать программу, реализующую алгоритм управления хаотической моделью с целью стабилизации объекта в окрестности устойчивого состояния.

4. Получить сравнительные графики управляемой и неуправляемой моделей.

5. Составить и представить преподавателю отчет о работе.

Часть 2.

1. Ознакомиться со справочными сведениями относительно применения дискретных/непрерывных блоков Simulink.

2. Построить модель системы автоматического регулирования в Simulink.

3. В отчет включить схему и скриншоты окон настроек каждого блока.

4. Описать принцип работы блока Линейные системы.

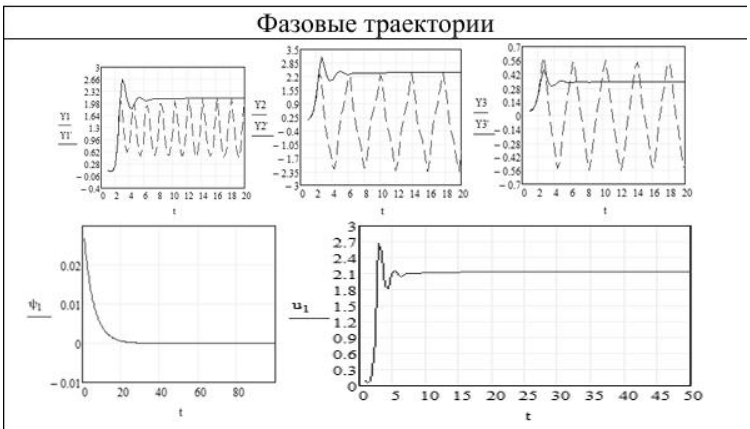
5. Представить необходимые графики.

Моделирование объектов детерминированного хаоса

Внимание! Подробный вывод управляющих воздействий должен быть приложен к отчету ЛР-5. Указать вид функционала качества управляющего воздействия и вид уравнения Эйлера-Лагранжа для него. Часть 1. Получить АКС определенного вида на основе принудительного возмущения u системы уравнений (1). Показать графики устойчивого и неустойчивого поведения (1) при разных параметрах целенаправленного возмущения (управления динамикой (1)).

Блок заданий 4. Получить АКС определенного вида на основе принудительного возмущения u_1 системы уравнений (4). Показать графики устойчивого и неустойчивого поведения (4) при разных параметрах целенаправленного возмущения (управления динамикой (4)).

Описание объекта	Макропеременная. Управление.
$\frac{dY_1}{dt} = \alpha Y_2 Y_3 - \gamma Y_1;$ $\frac{dY_2}{dt} = \mu(Y_2 + Y_3) - \beta Y_1 Y_3;$ (4) $\frac{dY_3}{dt} = \delta Y_2 - \lambda Y_3 + u_1,$	$\psi_1(t) = Y_3 - \rho Y_2.$ $u_1 = \lambda Y_3 - \delta Y_2 + \rho(\mu(Y_2 + Y_3) - \beta Y_1 Y_3) - \frac{\psi_1(t)}{T_1}.$ Получить систему управления. Выбором параметра T_1 можно разные формы переходных процессов.
Построить траектории с управлением и без (графики $Y_i(t), i = 1,2,3$ от времени), график управления, макропеременной и фазовый портрет. Получить два варианта картинок ФП: устойчивого поведения и неустойчивого.	



Указать, где находят применение данный тип модели (Лоренца).

Вариант

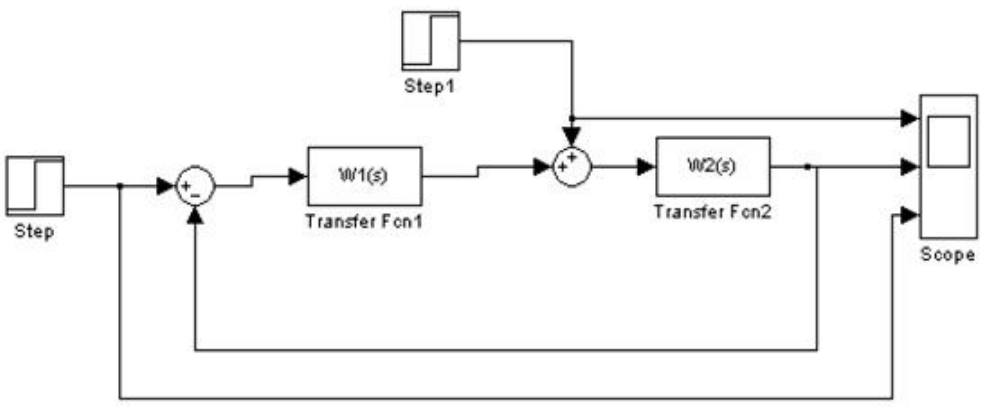
α=3, β=8, γ=1.8, μ=2.4, λ=3.10, δ=0.7

Часть 2

1. Собрать схему.

W1(s)~ 40; W2(s) ~ $0.4/(2s^2+s+1)$

2. Установить возмущение равным нулю и снять переходную характеристику по задающему воздействию. По полученному графику оценить показатели качества системы. Изменить значение статического коэффициента W1(s) в 10 раз в большую и меньшую стороны и для каждого измененного значения получить переходные характеристики и оценки показателей качества. Сравнить.



Решение

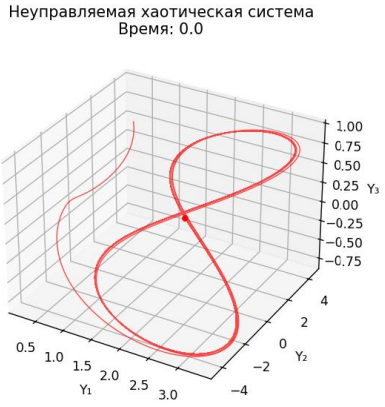
Часть 1

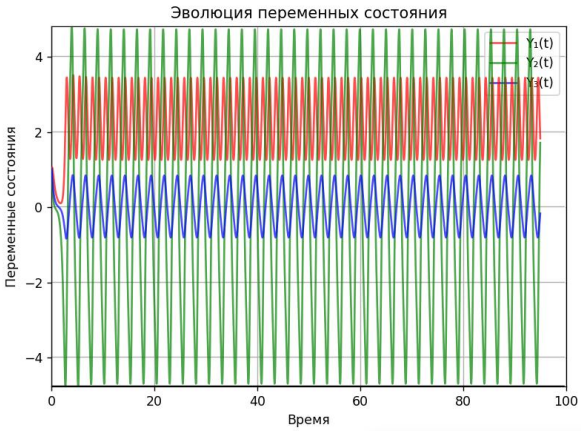
Решение

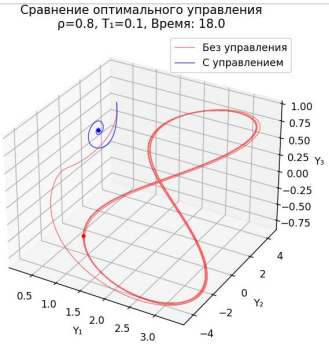
```
КОМПЛЕКСНЫЙ АНАЛИЗ С АНИМАЦИЯМИ - ВАРИАНТ 15
======================================================================
Параметры системы: α=3, β=8, γ=1.8
                   μ=2.4, λ=3.1, δ=0.7
======================================================================

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ
Уравнения системы:
dY₁/dt = αY₂Y₃ - γY₁
dY₂/dt = μ(Y₂ + Y₃) - βY₁Y₃
dY₃/dt = δY₂ - λY₃ + u₁

Закон управления (синергетическое управление):
Макропеременная: ψ = Y₃ - ρY₂
Управление: u₁ = λY₃ - δY₂ + ρ[μ(Y₂+Y₃) - βY₁Y₃] - ψ/T₁

Критерий устойчивости: ψ → 0 при t → ∞

2. АНАЛИЗ НЕУПРАВЛЯЕМОЙ СИСТЕМЫ
Ожидается: Хаотическое поведение из-за положительных показателей Ляпунова

Создание Анимации 1: Неуправляемая хаотическая система...

3. УПРАВЛЯЕМАЯ СИСТЕМА - ОПТИМИЗАЦИЯ ПАРАМЕТРОВ
Тестирование различных комбинаций (ρ, T₁) для поиска оптимальной устойчивости...

Тестирование: ρ=0.5, T₁=0.05 (Очень сильное управление)
  Конечный ψ: 0.000000
  Среднее |ψ| (последние 10%): 0.000000
  Макс |u₁|: 9.200000
  Время установления: 0.20
  Оценка устойчивости: 0.920000
```

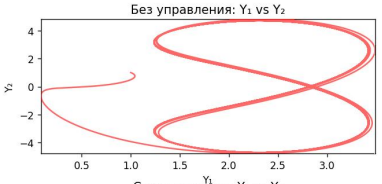
```
Тестирование: ρ=0.5, T₁=0.1 (Сильное управление)
  Конечный ψ: 0.000000
  Среднее |ψ| (последние 10%): 0.000000
  Макс |u₁|: 4.200000
  Время установления: 0.40
  Оценка устойчивости: 0.420000

Тестирование: ρ=0.8, T₁=0.1 (Умеренное ρ, сильное управление)
  Конечный ψ: 0.000000
  Среднее |ψ| (последние 10%): 0.000000
  Макс |u₁|: 2.160000
  Время установления: 0.30
  Оценка устойчивости: 0.216000

Тестирование: ρ=1.0, T₁=0.1 (Большое ρ, сильное управление)
  Конечный ψ: 0.000000
  Среднее |ψ| (последние 10%): 0.000000
  Макс |u₁|: 2.361118
  Время установления: Не установилось
  Оценка устойчивости: 0.236112

Тестирование: ρ=0.5, T₁=1.0 (Слабое управление)
  Конечный ψ: 0.000000
  Среднее |ψ| (последние 10%): 0.000000
  Макс |u₁|: 2.444015
  Время установления: 3.92
  Оценка устойчивости: 0.244402

Тестирование: ρ=0.5, T₁=5.0 (Очень слабое управление)
  Конечный ψ: 0.000000
  Среднее |ψ| (последние 10%): 0.000000
  Макс |u₁|: 2.511852
  Время установления: 19.56
  Оценка устойчивости: 0.251185
```

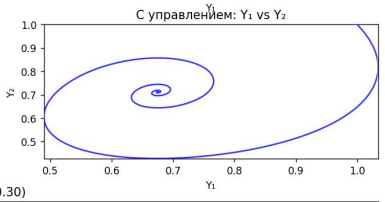
```
============================================================
4. РЕЗУЛЬТАТЫ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ
============================================
Найдены оптимальные параметры: ρ = 0.8, T₁ = 0.1
Лучшая оценка устойчивости: 0.216000

Создание Анимации 2: Сравнение управляемой и неуправляемой систем...

5. ДЕМОНСТРАЦИЯ УСТОЙЧИВОСТИ/НЕУСТОЙЧИВОСТИ

Создание Анимации 3: Улучшенное сравнение перебора параметров...

==========================================================
6. ФИНАЛЬНОЕ СВОДНОЕ ОПИСАНИЕ
==========================================================

Таблица сравнения параметров:
-------------------------------------------------------------------------------------
    ρ |       T₁ | Конечный |ψ| |   Время уст-ия |   Макс |u| |    Оценка уст-ти
-------------------------------------------------------------------------------------
 0.50 |    0.050 |     0.000000 |          0.20 |     9.2000 |         0.920000
 0.50 |    0.100 |     0.000000 |          0.40 |     4.2000 |         0.420000
 0.80 |    0.100 |     0.000000 |          0.30 |     2.1600 |         0.216000
 1.00 |    0.100 |     0.000000 |           Н/Д |     2.3611 |         0.236112
 0.50 |    1.000 |     0.000000 |          3.92 |     2.4440 |         0.244402
 0.50 |    5.000 |     0.000000 |         19.56 |     2.5119 |         0.251185
-------------------------------------------------------------------------------------

Оптимальные параметры: ρ = 0.8, T₁ = 0.1
Лучшая оценка устойчивости: 0.216000

Создание статической визуализации всех переменных состояния...

Создание визуализации сигнала управления...
```

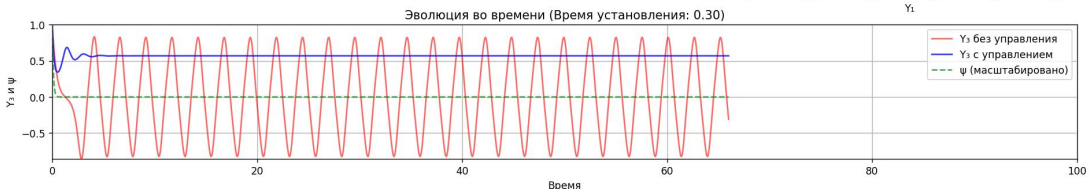

Неуправляемая хаотическая система
Время: 0.0
Y₁
Y₂
Y₃

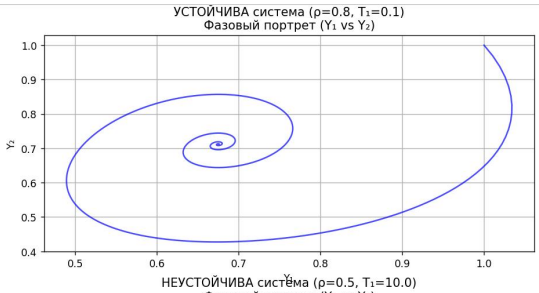

Эволюция переменных состояния
Y₁(t)
Y₂(t)
Y₃(t)
Переменные состояния
Время

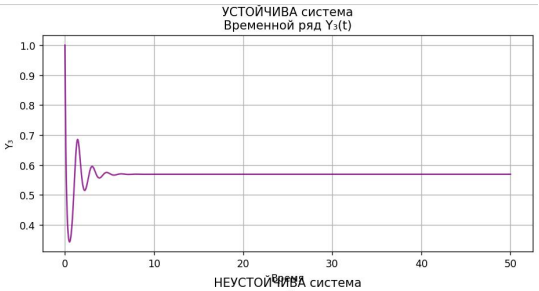

Сравнение оптимального управления
ρ=0.8, T₁=0.1, Время: 18.0
Без управления
С управлением
Y₁
Y₂
Y₃

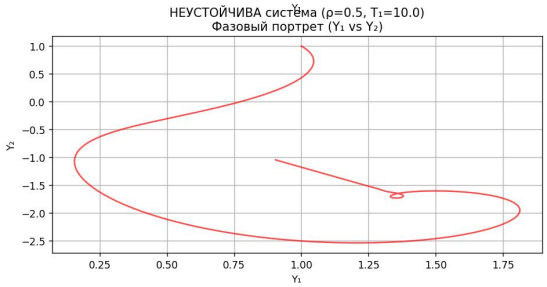

Без управления: Y₁ vs Y₂
Y₁
Y₂

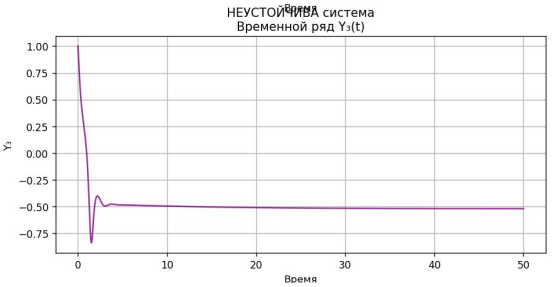

С управлением: Y₁ vs Y₂
Y₁
Y₂

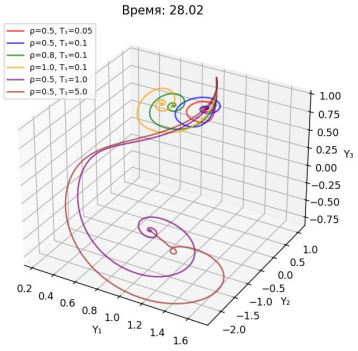

Эволюция во времени (Время установления: 0.30)
Y₃ без управления
Y₃ с управлением
ψ (масштабировано)
Y₃ и ψ
Время

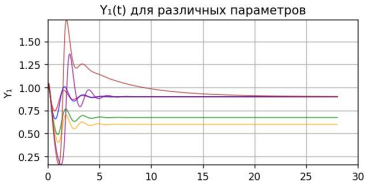

УСТОЙЧИВА система (ρ=0.8, T₁=0.1)
Фазовый портрет (Y₁ vs Y₂)
Y₁
Y₂

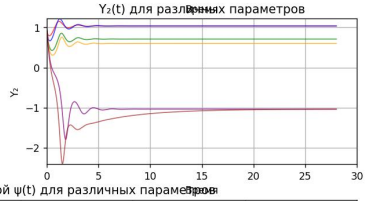

УСТОЙЧИВА система
Временной ряд Y₃(t)
Y₃
Время

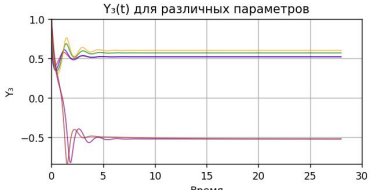

НЕУСТОЙЧИВА система (ρ=0.5, T₁=10.0)
Фазовый портрет (Y₁ vs Y₂)
Y₁
Y₂

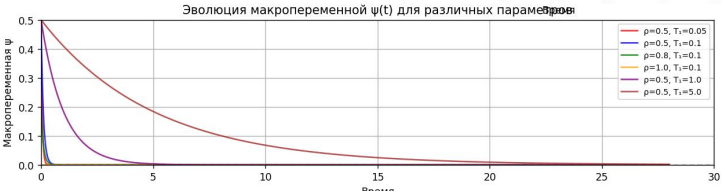

НЕУСТОЙЧИВА система
Временной ряд Y₃(t)
Y₃
Время

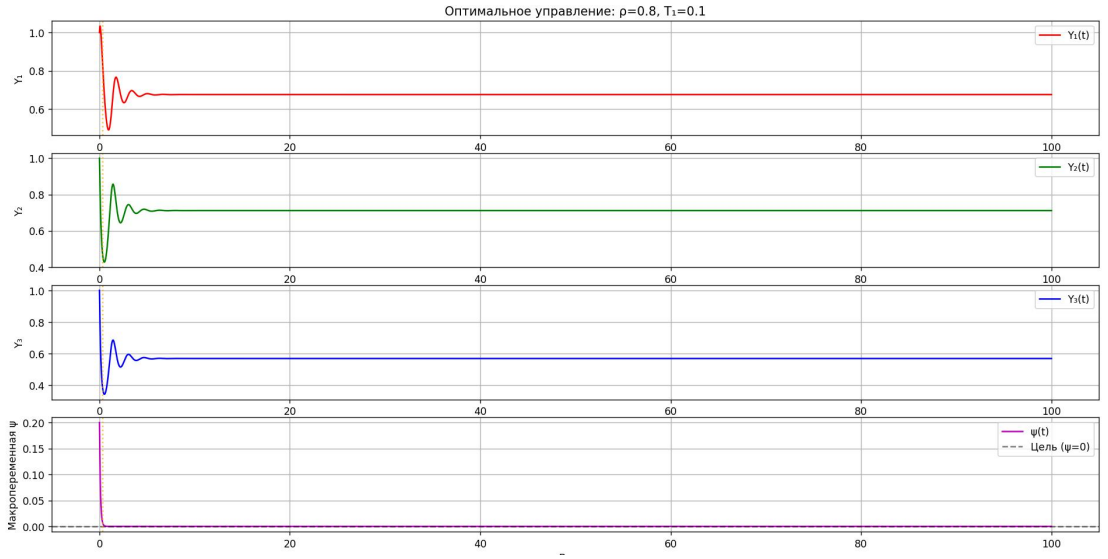

Время: 28.02
ρ=0.5, T₁=0.05
ρ=0.5, T₁=0.1
ρ=0.8, T₁=0.1
ρ=1.0, T₁=0.1
ρ=0.5, T₁=1.0
ρ=0.5, T₁=5.0
Y₁
Y₂
Y₃

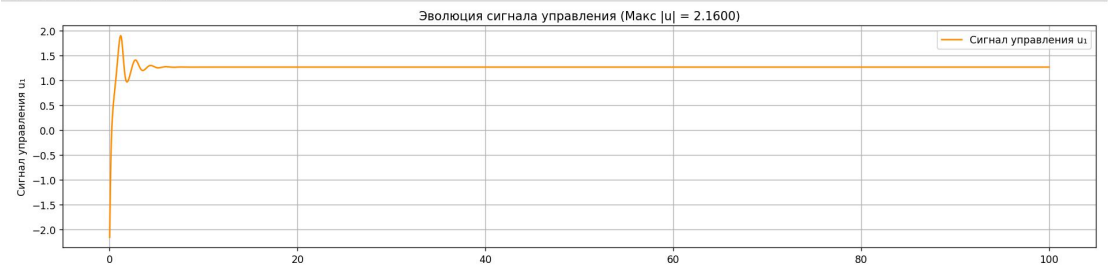

Y₁(t) для различных параметров
Y₁

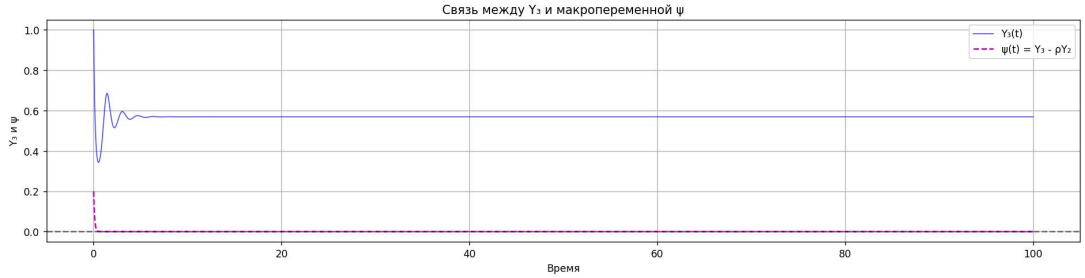

Y₂(t) для различных параметров
Y₂


Y₃(t) для различных параметров
Y₃


Эволюция макропеременной ψ(t) для различных параметров
Макропеременная ψ
ρ=0.5, T₁=0.05
ρ=0.5, T₁=0.1
ρ=0.8, T₁=0.1
ρ=1.0, T₁=0.1
ρ=0.5, T₁=1.0
ρ=0.5, T₁=5.0

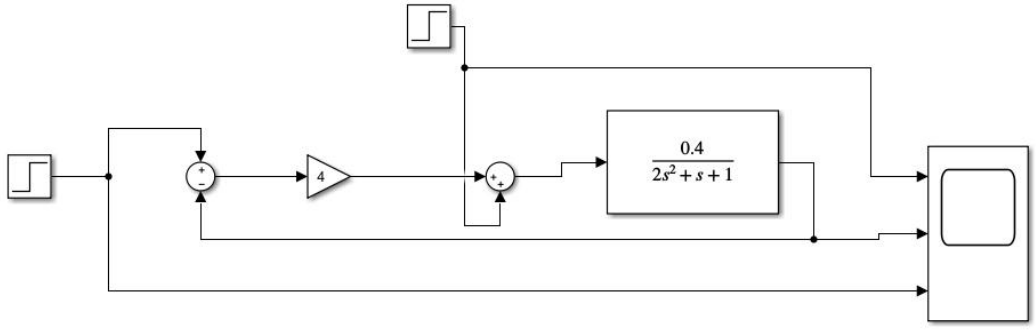

Оптимальное управление: ρ=0.8, T₁=0.1
Y₁(t)
Y₂(t)
Y₃(t)
ψ(t)
Цель (ψ=0)
Макропеременная ψ

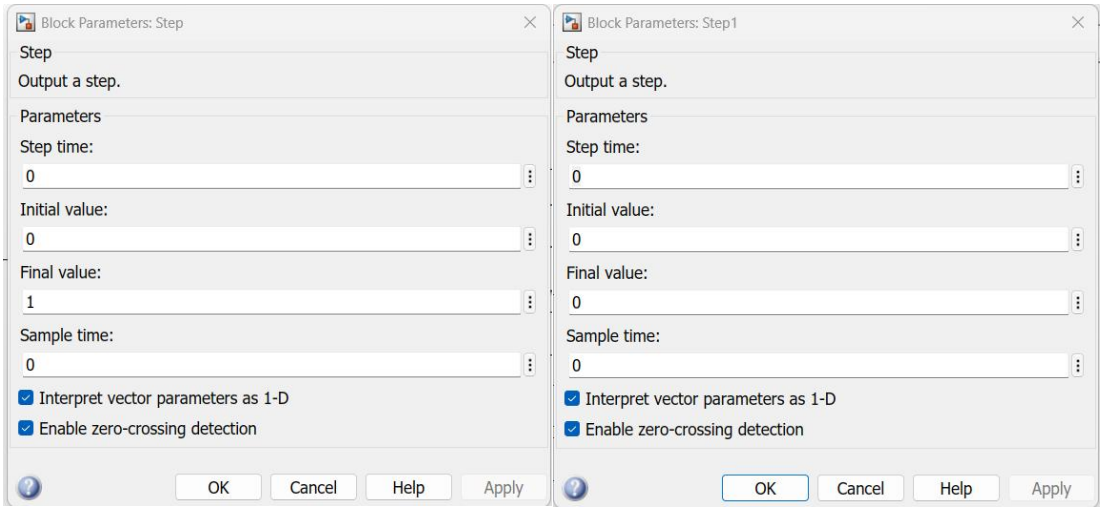

Эволюция сигнала управления (Макс |u| = 2.1600)
Сигнал управления u₁

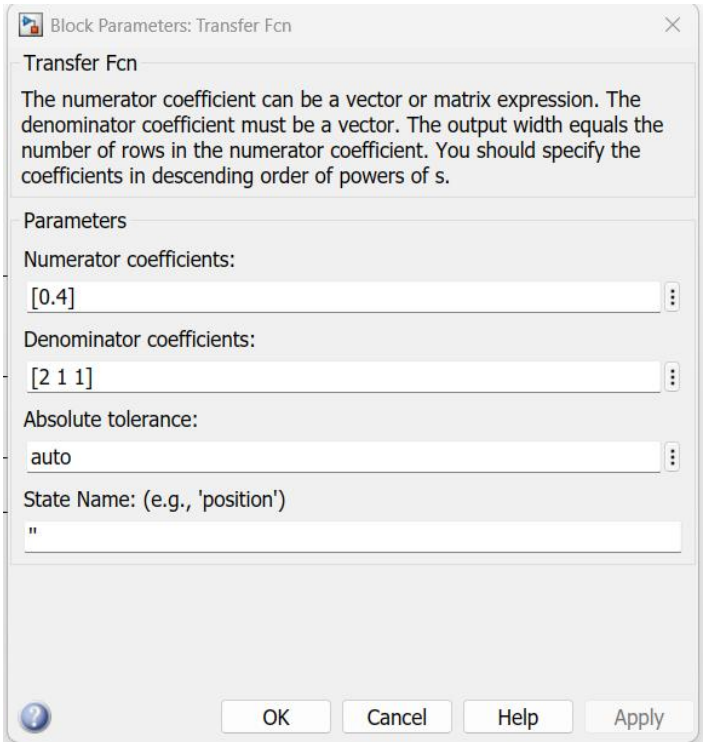

Связь между Y₃ и макропеременной ψ
Y₃(t)
ψ(t) = Y₃ - ρY₂
Y₃ и ψ
Время

Часть 2

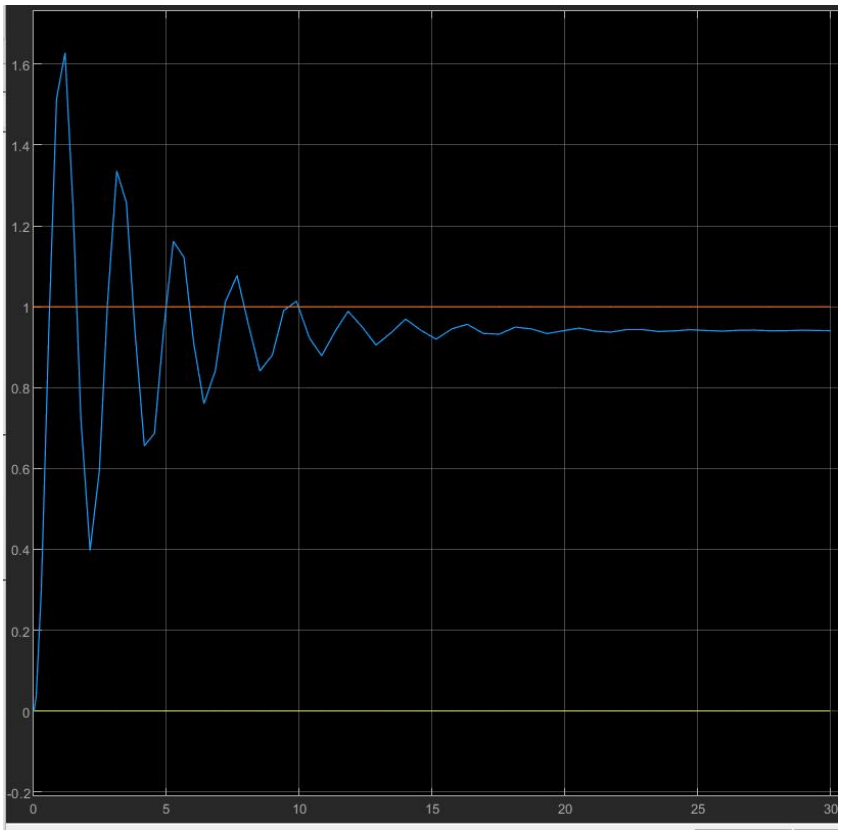
схема



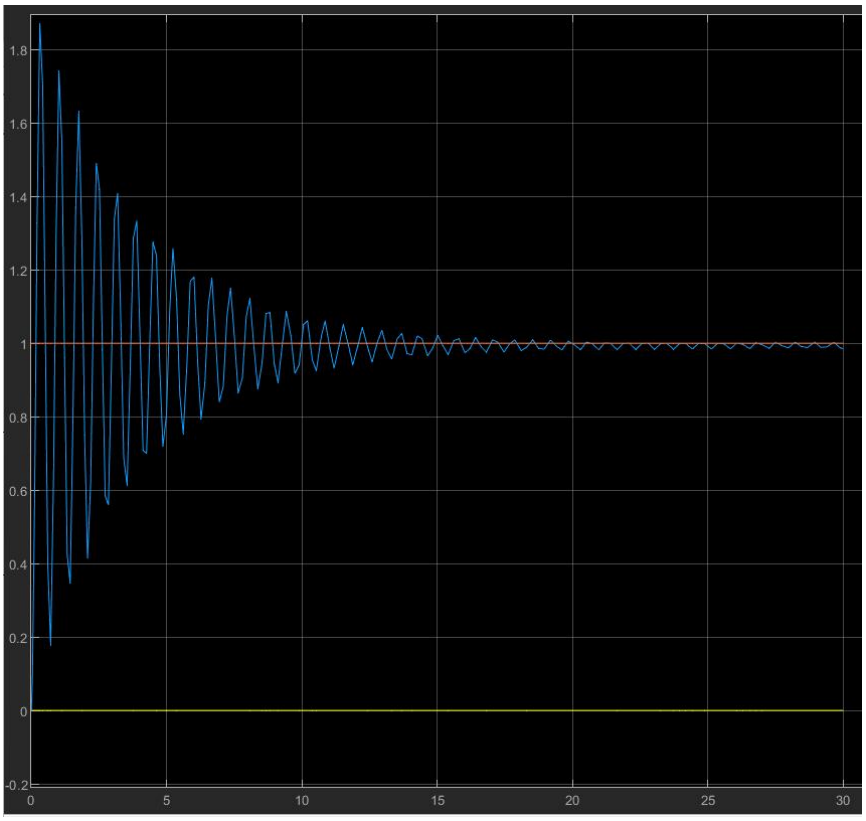




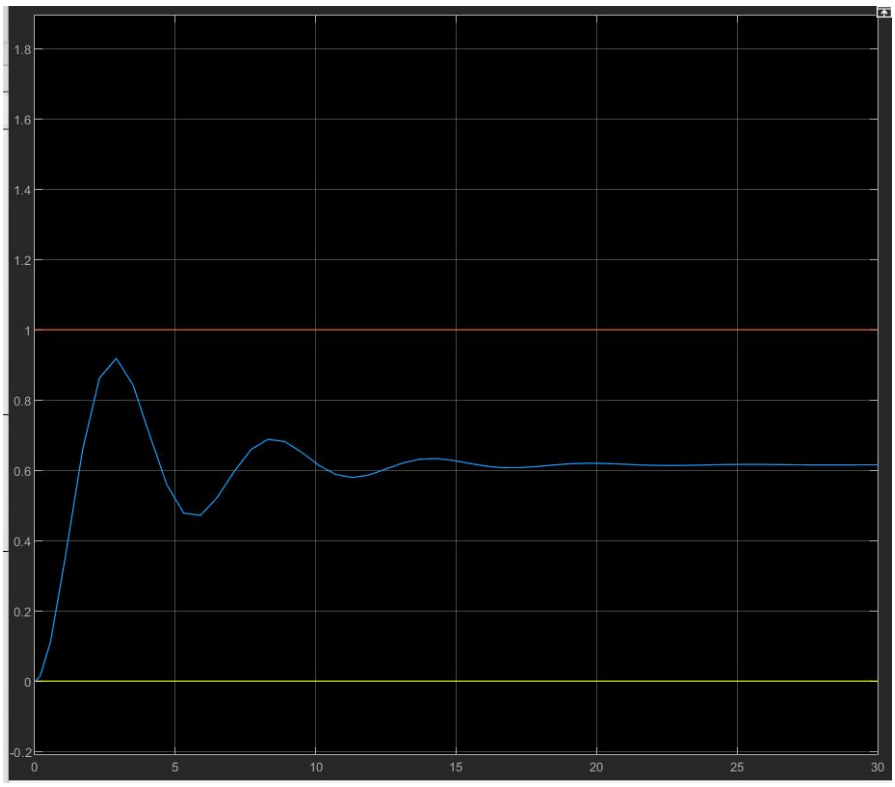
Оригинал: $w_1 = 40$



В 10 раз больше: $w_1 = 400$



В 10 раз меньше: $w_1 = 4$



$$\delta = \frac{|h_{max} - h(\infty)|}{h(\infty)} \times 100\%$$

Dacă $W_1 = 40$

$h_{max} = 1.67$, $h(\infty) = 0.957$

$$\delta = \frac{|1.67 - 0.957|}{0.957} \times 100\% = 74.5\%$$

$\varepsilon = |1 - 0.957| = 0.043$

$t_r = 25s$

Dacă $W_1 = 400$

$h_{max} = 1.875$, $h(\infty) = 0.99$

$$\delta = \frac{|1.875 - 0.99|}{0.99} \times 100\% = 89.4\%$$

$\varepsilon = |1 - 0.99| = 0.01$ $\Rightarrow$ $t_r = 30s$

Dacă $W_1 = 4$

$h_{max} = 0.91$, $h(\infty) = 0.617$

$$\delta = \frac{|0.91 - 0.617|}{0.617} \times 100\% = 47.48\%$$

$\varepsilon = |1 - 0.383|$, $t_r = 20$

Таблица 1: Показатели качества при различных значениях коэффициента W_1

Значение W_1	Установившееся значение $h(\infty)$	Максимальное значение h_{max}	Перерегулирование σ (%)	Время регулирования t_n (с)	Статическая ошибка ε_{st}
4	0,617	0,91	0,475	15	0,383
40	0,957	1,67	0,745	25	0,043
400	0,99	1,875	0,894	50	0,01

При увеличении коэффициента усиления W_1 наблюдаются следующие изменения в динамике системы:

Точность (статическая ошибка): Существенно улучшается. При $W_1 = 4$ ошибка составляет 0,383, при $W_1 = 40$ — 0,043, а при $W_1 = 400$ — всего 0,01. Система становится более точной.

Быстродействие (время регулирования): Ухудшается. Время переходного процесса возрастает с 20 с (W_1=4) до 35 с (W_1=400). Высокий коэффициент усиления приводит к более продолжительным колебаниям перед выходом на установившийся режим.

Перерегулирование: Резко увеличивается. С 47,5% при W_1=4 до 89,4% при W_1=400. Это указывает на рост колебательности системы и снижение запаса устойчивости.

Общий компромисс: Параметры качества системы демонстрируют классический компромисс между точностью, быстродействием и устойчивостью. Увеличение W_1 повышает точность, но ценой снижения быстродействия и значительного роста перерегулирования, что может привести к неустойчивости при дальнейшем увеличении коэффициента. Для данной системы

оптимальным, с точки зрения баланса показателей, можно считать значение $W_1 = 40$.

Выводы

В данной лабораторной работе успешно исследована возможность стабилизации хаотической системы Лоренца с помощью метода синергетического управления. Найдены оптимальные параметры регулятора (ρ=0.8, T_1=0.1), обеспечивающие быстрое подавление хаоса при минимальных энергетических затратах. Показано, что неуправляемая система демонстрирует сложное хаотическое поведение, в то время как введение управляющего воздействия позволяет надежно стабилизировать её. Визуализации подтвердили эффективность метода, продемонстрировав переход от неустойчивого режима к устойчивому состоянию равновесия.

Приложение для 1 часть

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation as animation
from matplotlib.gridspec import GridSpec

class СистемаУправления Лоренца:
    def __init__(self, alpha, beta, gamma, mu, lamda, delta, rho=1.0, T1=1.0, использовать_управление=True):
        self.alpha = alpha
        self.beta = beta
        self.gamma = gamma
        self.mu = mu
        self.lamda = lamda
        self.delta = delta
        self.rho = rho
        self.T1 = T1
        self.использовать_управление = использовать_управление

    def уравнения_системы(self, t, Y):
        Y1, Y2, Y3 = Y

        # Вычисление макропеременной ψ = Y₃ - ρY₂
        psi = Y3 - self.rho * Y2

        # Закон управления из лабораторной работы
```

```
        if self.использовать_управление:
            u1 = (self.lamda * Y3 - self.delta * Y2 +
                  self.rho * (self.mu * (Y2 + Y3) - self.beta * Y1 * Y3) -
                  psi / self.T1)
        else:
            u1 = 0

        # Оригинальные уравнения системы типа Лоренца
        dY1_dt = self.alpha * Y2 * Y3 - self.gamma * Y1
        dY2_dt = self.mu * (Y2 + Y3) - self.beta * Y1 * Y3
        dY3_dt = self.delta * Y2 - self.lamda * Y3 + u1

        return [dY1_dt, dY2_dt, dY3_dt]

    def симулировать(self, Y0, t_span, t_eval):
        решение = solve_ivp(self.уравнения_системы, t_span, Y0,
t_eval=t_eval,
                            method='RK45', rtol=1e-8, atol=1e-10)
        return решение

def запустить_комплексный_анализ_с_анимациями():
    """
    ПОЛНЫЙ АНАЛИЗ С АНИМАЦИЯМИ ДЛЯ ВАРИАНТА 15
    Включает весь исходный анализ ПЛЮС анимации
    """

    # ===== ПАРАМЕТРЫ СИСТЕМЫ =====
    alpha, beta, gamma = 3, 8, 1.8
    mu, lamda, delta = 2.4, 3.10, 0.7
```

```
print("=" * 70)
print("КОМПЛЕКСНЫЙ АНАЛИЗ С АНИМАЦИЯМИ - ВАРИАНТ 15")
print("=" * 70)
print(f"Параметры системы: α={alpha}, β={beta}, γ={gamma}")
print(f"                   μ={mu}, λ={lamda}, δ={delta}")
print("=" * 70)

# ===== ТЕОРЕТИЧЕСКАЯ ОСНОВА =====
print("\n1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ")
print("Уравнения системы:")
print("dY₁/dt = αY₂Y₃ - γY₁")
print("dY₂/dt = μ(Y₂ + Y₃) - βY₁Y₃")
print("dY₃/dt = δY₂ - λY₃ + u₁")
print("\nЗакон управления (синергетическое управление):")
print("Макропеременная: ψ = Y₃ - ρY₂")
print("Управление: u₁ = λY₃ - δY₂ + ρ[μ(Y₂+Y₃) - βY₁Y₃] - ψ/T₁")
print("\nКритерий устойчивости: ψ → 0 при t → ∞")

# ===== НАЧАЛЬНЫЕ УСЛОВИЯ И НАСТРОЙКА ВРЕМЕНИ =====
Y0 = [1.0, 1.0, 1.0]  # Начальное состояние
t_span = [0, 100]      # Увеличенное время симуляции
t_eval = np.linspace(t_span[0], t_span[1], 10000)

# ===== 1. АНАЛИЗ НЕУПРАВЛЯЕМОЙ СИСТЕМЫ =====
print("\n2. АНАЛИЗ НЕУПРАВЛЯЕМОЙ СИСТЕМЫ")
```

```
print("Ожидается: Хаотическое поведение из-за положительных показателей Ляпунова")

система_неуправляемая = СистемаУправленияЛоренца(alpha, beta, gamma, mu, lamda, delta, использовать_управление=False)
решение_неуправляемое = система_неуправляемая.симулировать(Y0, t_span, t_eval)

# АНИМАЦИЯ 1: Эволюция неуправляемой системы
print("\nСоздание Анимации 1: Неуправляемая хаотическая система...")
fig1 = plt.figure(figsize=(12, 5))
ax1 = fig1.add_subplot(121, projection='3d')
ax2 = fig1.add_subplot(122)

Y1_u, Y2_u, Y3_u = решение_неуправляемое.y

# Инициализация графиков
линия_3d, = ax1.plot([], [], [], 'r-', alpha=0.7, linewidth=0.8)
точка_3d, = ax1.plot([], [], [], 'ro', markersize=4)
линия_y1, = ax2.plot([], [], 'r-', alpha=0.7, label='Y₁(t)')
линия_y2, = ax2.plot([], [], 'g-', alpha=0.7, label='Y₂(t)')
линия_y3, = ax2.plot([], [], 'b-', alpha=0.7, label='Y₃(t)')

# Установка пределов
ax1.set_xlim([Y1_u.min(), Y1_u.max()])
ax1.set_ylim([Y2_u.min(), Y2_u.max()])
ax1.set_zlim([Y3_u.min(), Y3_u.max()])
ax1.set_xlabel('Y₁'); ax1.set_ylabel('Y₂'); ax1.set_zlabel('Y₃')
```

```
    ax2.set_xlim([t_eval[0], t_eval[-1]])
    ax2.set_ylim([min(Y1_u.min(), Y2_u.min(), Y3_u.min()),
                  max(Y1_u.max(), Y2_u.max(), Y3_u.max())])
    ax2.set_xlabel('Время'); ax2.set_ylabel('Переменные состояния')
    ax2.legend(); ax2.grid(True)

    def обновить_неуправляемую(кадр):
        линия_3d.set_data(Y1_u[:кадр], Y2_u[:кадр])
        линия_3d.set_3d_properties(Y3_u[:кадр])
        точка_3d.set_data([Y1_u[кадр]], [Y2_u[кадр]])
        точка_3d.set_3d_properties([Y3_u[кадр]])

        линия_y1.set_data(t_eval[:кадр], Y1_u[:кадр])
        линия_y2.set_data(t_eval[:кадр], Y2_u[:кадр])
        линия_y3.set_data(t_eval[:кадр], Y3_u[:кадр])

        ax1.set_title(f'Неуправляемая хаотическая система\nВремя: {t_eval[кадр]:.1f}')
        ax2.set_title('Эволюция переменных состояния')

        return линия_3d, точка_3d, линия_y1, линия_y2, линия_y3

    анимация1 = animation.FuncAnimation(fig1, обновить_неуправляемую,
                                        frames=range(0, len(t_eval), 100),
                                        interval=50, blit=True)

    plt.tight_layout()
```

```
    plt.show()

    # ===== 2. ОПТИМИЗАЦИЯ ПАРАМЕТРОВ (Исходный анализ) =====
    print("\n3. УПРАВЛЯЕМАЯ СИСТЕМА - ОПТИМИЗАЦИЯ ПАРАМЕТРОВ")
    print("Тестирование различных комбинаций (ρ, T₁) для поиска оптимальной устойчивости...")

    тестовые_параметры = [
        (0.5, 0.05, "Очень сильное управление"),
        (0.5, 0.1,  "Сильное управление"),
        (0.8, 0.1,  "Умеренное ρ, сильное управление"),
        (1.0, 0.1,  "Большое ρ, сильное управление"),
        (0.5, 1.0,  "Слабое управление"),
        (0.5, 5.0,  "Очень слабое управление"),
    ]

    лучшая_устойчивость = float('inf')
    лучшие_параметры = None
    лучшее_решение = None

    результаты = []

    for rho, T1, описание in тестовые_параметры:
        print(f"\nТестирование: ρ={rho}, T₁={T1} ({описание})")

        система_управляемая = СистемаУправленияЛоренца(alpha, beta, gamma, mu, lamda, delta,
```

```
                                    rho=rho,                                T1=T1,
использовать_управление=True)
    решение_управляемое                                                    =
система_управляемая.симулировать(Y0, t_span, t_eval)

    # Вычисление сигналов управления и макропеременной
    значения_u = []
    значения_psi = []
    for i, t in enumerate(t_eval):
        Y1, Y2, Y3 = решение_управляемое.y[:, i]
        psi = Y3 - rho * Y2
        значения_psi.append(psi)
        u1 = (lamda * Y3 - delta * Y2 +
              rho * (mu * (Y2 + Y3) - beta * Y1 * Y3) - psi / T1)
        значения_u.append(u1)

    значения_u = np.array(значения_u)
    значения_psi = np.array(значения_psi)

    # Метрики устойчивости (ИСХОДНЫЙ АНАЛИЗ)
    конечный_psi = abs(значения_psi[-1])
    средний_psi_последний = np.mean(abs(значения_psi[-1000:]))  #
Последние 10%
    максимальное_управление = np.max(abs(значения_u))
    время_установления = None

    # Нахождение времени установления (когда |ψ| < 0.01)
    for i, psi in enumerate(значения_psi):
        if abs(psi) < 0.01 and all(abs(значения_psi[i:]) < 0.02):
```

```
            время_установления = t_eval[i]
            break

    оценка_устойчивости = средний_psi_последний + 0.1 * максимальное_управление

    print(f"  Конечный ψ: {конечный_psi:.6f}")
    print(f"  Среднее |ψ| (последние 10%): {средний_psi_последний:.6f}")
    print(f"  Макс |u₁|: {максимальное_управление:.6f}")
    if время_установления:
        print(f"  Время установления: {время_установления:.2f}")
    else:
        print(f"  Время установления: Не установилось")
    print(f"  Оценка устойчивости: {оценка_устойчивости:.6f}")

    if оценка_устойчивости < лучшая_устойчивость and время_установления:
            лучшая_устойчивость = оценка_устойчивости
            лучшие_параметры = (rho, T1)
            лучшее_решение = решение_управляемое

    результаты.append({
        'параметры': (rho, T1, описание),
        'решение': решение_управляемое,
        'значения_u': значения_u,
        'значения_psi': значения_psi,
        'оценка_устойчивости': оценка_устойчивости,
        'время_установления': время_установления,
```

```
            'конечный_psi': конечный_psi,
            'максимальное_управление': максимальное_управление,
            'средний_psi_последний': средний_psi_последний
        })

    # ===== 3. АНАЛИЗ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ =====
    print("\n" + "="*50)
    print("4. РЕЗУЛЬТАТЫ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ")
    print("="*50)

    if лучшие_параметры:
        rho_опт, T1_опт = лучшие_параметры
        print(f"Найдены оптимальные параметры: ρ = {rho_опт}, T₁ = {T1_опт}")
        print(f"Лучшая оценка устойчивости: {лучшая_устойчивость:.6f}")

        # Получение оптимального решения
        оптимальный_индекс = next(i for i, r in enumerate(результаты) if r['параметры'][:2] == лучшие_параметры)
        оптимальный_результат = результаты[оптимальный_индекс]

        # АНИМАЦИЯ 2: Сравнение с управлением и без
        print("\nСоздание Анимации 2: Сравнение управляемой и неуправляемой систем...")
        fig2 = plt.figure(figsize=(15, 10))
        gs = GridSpec(3, 3, figure=fig2)

        ax_3d = fig2.add_subplot(gs[0:2, 0:2], projection='3d')
```

```
    ax_фаза_u = fig2.add_subplot(gs[0, 2])
    ax_фаза_c = fig2.add_subplot(gs[1, 2])
    ax_время = fig2.add_subplot(gs[2, :])

    # Инициализация элементов анимации
    Y1_c, Y2_c, Y3_c = оптимальный_результат['решение'].y

    линия_3d_u, = ax_3d.plot([], [], [], 'r-', alpha=0.6, linewidth=0.7, label='Без управления')
    линия_3d_c, = ax_3d.plot([], [], [], 'b-', alpha=0.8, linewidth=0.8, label='С управлением')
    точка_u, = ax_3d.plot([], [], [], 'ro', markersize=3)
    точка_c, = ax_3d.plot([], [], [], 'bo', markersize=3)

    линия_фазы_u, = ax_фаза_u.plot([], [], 'r-', alpha=0.6)
    линия_фазы_c, = ax_фаза_c.plot([], [], 'b-', alpha=0.8)

    линия_времени_u, = ax_время.plot([], [], 'r-', alpha=0.6, label='Y₃ без управления')
    линия_времени_c, = ax_время.plot([], [], 'b-', alpha=0.8, label='Y₃ с управлением')
    линия_psi, = ax_время.plot([], [], 'g--', alpha=0.7, label='ψ (масштабировано)')

    # Установка пределов
    ax_3d.set_xlim([min(Y1_u.min(), Y1_c.min()), max(Y1_u.max(), Y1_c.max())])
    ax_3d.set_ylim([min(Y2_u.min(), Y2_c.min()), max(Y2_u.max(), Y2_c.max())])
```

```
    ax_3d.set_zlim([min(Y3_u.min(),  Y3_c.min()),  max(Y3_u.max(),
Y3_c.max())])
    ax_3d.set_xlabel('Y₁'); ax_3d.set_ylabel('Y₂'); ax_3d.set_zlabel('Y₃')
    ax_3d.legend()

    ax_фаза_u.set_xlim([Y1_u.min(), Y1_u.max()])
    ax_фаза_u.set_ylim([Y2_u.min(), Y2_u.max()])
    ax_фаза_u.set_xlabel('Y₁'); ax_фаза_u.set_ylabel('Y₂')
    ax_фаза_u.set_title('Без управления: Y₁ vs Y₂')

    ax_фаза_c.set_xlim([Y1_c.min(), Y1_c.max()])
    ax_фаза_c.set_ylim([Y2_c.min(), Y2_c.max()])
    ax_фаза_c.set_xlabel('Y₁'); ax_фаза_c.set_ylabel('Y₂')
    ax_фаза_c.set_title('С управлением: Y₁ vs Y₂')

    ax_время.set_xlim([t_eval[0], t_eval[-1]])
    ax_время.set_ylim([min(Y3_u.min(),                    Y3_c.min(),
min(оптимальный_результат['значения_psi'])*5),
                max(Y3_u.max(), Y3_c.max())])
    ax_время.set_xlabel('Время'); ax_время.set_ylabel('Y₃ и ψ')
    ax_время.legend(); ax_время.grid(True)

    def обновить_сравнение(кадр):
        линия_3d_u.set_data(Y1_u[:кадр], Y2_u[:кадр])
        линия_3d_u.set_3d_properties(Y3_u[:кадр])
        линия_3d_c.set_data(Y1_c[:кадр], Y2_c[:кадр])
        линия_3d_c.set_3d_properties(Y3_c[:кадр])

        точка_u.set_data([Y1_u[кадр]], [Y2_u[кадр]])
```

```
        точка_u.set_3d_properties([Y3_u[кадр]])
        точка_c.set_data([Y1_c[кадр]], [Y2_c[кадр]])
        точка_c.set_3d_properties([Y3_c[кадр]])

        линия_фазы_u.set_data(Y1_u[:кадр], Y2_u[:кадр])
        линия_фазы_c.set_data(Y1_c[:кадр], Y2_c[:кадр])

        линия_времени_u.set_data(t_eval[:кадр], Y3_u[:кадр])
        линия_времени_c.set_data(t_eval[:кадр], Y3_c[:кадр])
        линия_psi.set_data(t_eval[:кадр],
оптимальный_результат['значения_psi'][:кадр] * 5)

        ax_3d.set_title(f'Сравнение оптимального
управления\nρ={rho_опт}, T₁={T1_опт}, Время: {t_eval[кадр]:.1f}')
        if оптимальный_результат['время_установления']:
            ax_время.set_title(f'Эволюция во времени (Время
установления:
{оптимальный_результат["время_установления"]:.2f})')
        else:
            ax_время.set_title('Эволюция во времени (Не установилось)')

        return (линия_3d_u, линия_3d_c, точка_u, точка_c,
                линия_фазы_u, линия_фазы_c, линия_времени_u,
линия_времени_c, линия_psi)

    анимация2 = animation.FuncAnimation(fig2, обновить_сравнение,
                                        frames=range(0, len(t_eval), 100),
                                        interval=50, blit=True)
```

```
    plt.tight_layout()
    plt.show()

    # ===== 4. ДЕМОНСТРАЦИЯ УСТОЙЧИВОСТИ/НЕУСТОЙЧИВОСТИ =====
    print("\n5. ДЕМОНСТРАЦИЯ УСТОЙЧИВОСТИ/НЕУСТОЙЧИВОСТИ")

    # Показать один устойчивый и один неустойчивый случай
    устойчивые_параметры = лучшие_параметры
    неустойчивые_параметры = (0.5, 10.0)  # Слабое управление

    # Статический график для сравнения устойчивости (из исходного)
    fig3, axes = plt.subplots(2, 2, figsize=(15, 10))

    for idx, (параметры, тип_устойчивости) in enumerate([(устойчивые_параметры, "УСТОЙЧИВА"),
                                                          (неустойчивые_параметры, "НЕУСТОЙЧИВА")]):
        rho, T1 = параметры
        система_тест = СистемаУправленияЛоренца(alpha, beta, gamma, mu, lamda, delta,
                                                rho=rho, T1=T1, использовать_управление=True)
        решение_тест = система_тест.симулировать(Y0, [0, 50], np.linspace(0, 50, 5000))

        Y1, Y2, Y3 = решение_тест.y
```

```
        # 2D фазовый портрет
        axes[idx, 0].plot(Y1, Y2, 'blue' if тип_устойчивости == "УСТОЙЧИВА" else 'red', alpha=0.7)
        axes[idx, 0].set_xlabel('Y₁'); axes[idx, 0].set_ylabel('Y₂')
        axes[idx, 0].set_title(f'{тип_устойчивости} система (ρ={rho}, T₁={T1})\nФазовый портрет (Y₁ vs Y₂)')
        axes[idx, 0].grid(True)

        # Временные ряды
        axes[idx, 1].plot(решение_тест.t, Y3, 'purple', alpha=0.8)
        axes[idx, 1].set_xlabel('Время'); axes[idx, 1].set_ylabel('Y₃')
        axes[idx, 1].set_title(f'{тип_устойчивости} система\nВременной ряд Y₃(t)')
        axes[idx, 1].grid(True)

    plt.tight_layout()
    plt.show()

    # ===== 5. УЛУЧШЕННАЯ ВИЗУАЛИЗАЦИЯ ПЕРЕБОРА ПАРАМЕТРОВ =====
    print("\nСоздание Анимации 3: Улучшенное сравнение перебора параметров...")

    # Создание улучшенной анимации перебора параметров с отдельными графиками Y1, Y2, Y3
    fig4 = plt.figure(figsize=(16, 12))
    gs_param = GridSpec(3, 3, figure=fig4)
```

```
    # 3D график
    ax_param_3d = fig4.add_subplot(gs_param[0:2, 0:2], projection='3d')

    # Графики временных рядов для Y1, Y2, Y3, ψ
    ax_y1 = fig4.add_subplot(gs_param[0, 2])
    ax_y2 = fig4.add_subplot(gs_param[1, 2])
    ax_y3 = fig4.add_subplot(gs_param[2, 0])
    ax_psi = fig4.add_subplot(gs_param[2, 1:])

    colors = ['red', 'blue', 'green', 'orange', 'purple', 'brown']

    # Хранение данных для каждого набора параметров с уникальными ID
    данные_параметров = []
    линии_3d = []

    for i, (rho, T1, desc) in enumerate(тестовые_параметры):
        система = СистемаУправленияЛоренца(alpha, beta, gamma, mu, lamda, delta,
                                           rho=rho, T1=T1, использовать_управление=True)
        решение = система.симулировать(Y0, [0, 30], np.linspace(0, 30, 1500))

        # Вычисление макропеременной ψ
        значения_psi = []
        for j in range(len(решение.t)):
            Y1_val, Y2_val, Y3_val = решение.y[:, j]
```

```
            psi = Y3_val - rho * Y2_val
            значения_psi.append(psi)

        color = colors[i % len(colors)]

        # Создание 3D линии
        линия_3d, = ax_param_3d.plot([], [], [], color=color, alpha=0.7,
                                     linewidth=1.5, label=f'ρ={rho}, T₁={T1}')
        линии_3d.append(линия_3d)

        # Сохранение для обновлений анимации
        данные_параметров.append({
            'id': i,  # Уникальный идентификатор
            'решение': решение,
            'rho': rho,
            'T1': T1,
            'значения_psi': значения_psi,
            'color': color,
            'линия_3d': линия_3d,
            'линия_y1': ax_y1.plot([], [], color=color, alpha=0.7,
linewidth=1)[0],
            'линия_y2': ax_y2.plot([], [], color=color, alpha=0.7,
linewidth=1)[0],
            'линия_y3': ax_y3.plot([], [], color=color, alpha=0.7,
linewidth=1)[0],
            'линия_psi': ax_psi.plot([], [], color=color, alpha=0.8,
linewidth=1.5)[0],
        })
```

```
    # Установка пределов
    все_Y1 = np.concatenate([p['решение'].y[0] for p in данные_параметров])
    все_Y2 = np.concatenate([p['решение'].y[1] for p in данные_параметров])
    все_Y3 = np.concatenate([p['решение'].y[2] for p in данные_параметров])
    все_psi = np.concatenate([p['значения_psi'] for p in данные_параметров])
    все_времена = данные_параметров[0]['решение'].t  # Все имеют одинаковый вектор времени

    ax_param_3d.set_xlim([все_Y1.min(), все_Y1.max()])
    ax_param_3d.set_ylim([все_Y2.min(), все_Y2.max()])
    ax_param_3d.set_zlim([все_Y3.min(), все_Y3.max()])
    ax_param_3d.set_xlabel('Y₁'); ax_param_3d.set_ylabel('Y₂'); ax_param_3d.set_zlabel('Y₃')
    ax_param_3d.legend(loc='upper left', fontsize=8)

    # График Y1
    ax_y1.set_xlim([все_времена[0], все_времена[-1]])
    ax_y1.set_ylim([все_Y1.min(), все_Y1.max()])
    ax_y1.set_xlabel('Время'); ax_y1.set_ylabel('Y₁')
    ax_y1.set_title('Y₁(t) для различных параметров')
    ax_y1.grid(True)

    # График Y2
    ax_y2.set_xlim([все_времена[0], все_времена[-1]])
    ax_y2.set_ylim([все_Y2.min(), все_Y2.max()])
```

```
    ax_y2.set_xlabel('Время'); ax_y2.set_ylabel('Y₂')
    ax_y2.set_title('Y₂(t) для различных параметров')
    ax_y2.grid(True)

    # График Y3
    ax_y3.set_xlim([все_времена[0], все_времена[-1]])
    ax_y3.set_ylim([все_Y3.min(), все_Y3.max()])
    ax_y3.set_xlabel('Время'); ax_y3.set_ylabel('Y₃')
    ax_y3.set_title('Y₃(t) для различных параметров')
    ax_y3.grid(True)

    # График ψ
    ax_psi.set_xlim([все_времена[0], все_времена[-1]])
    ax_psi.set_ylim([все_psi.min(), все_psi.max()])
    ax_psi.set_xlabel('Время'); ax_psi.set_ylabel('Макропеременная ψ')
    ax_psi.set_title('Эволюция макропеременной ψ(t) для различных параметров')
    ax_psi.axhline(y=0, color='black', linestyle='--', alpha=0.3)
    ax_psi.grid(True)
    ax_psi.legend([f'ρ={p["rho"]}, T₁={p["T1"]}' for p in данные_параметров],
                  loc='upper right', fontsize=8)

    def обновить_перебор_параметров(кадр):
        # Ограничение кадра длиной данных
        текущий_кадр = min(кадр, len(все_времена)-1)

        for данные in данные_параметров:
            решение = данные['решение']
```

```
            Y1, Y2, Y3 = решение.y

            # Обновление 3D линии - используем данные['id'] как индекс
            данные['линия_3d'].set_data(Y1[:текущий_кадр], Y2[:текущий_кадр])
            данные['линия_3d'].set_3d_properties(Y3[:текущий_кадр])

            # Обновление графиков временных рядов
            данные['линия_y1'].set_data(все_времена[:текущий_кадр], Y1[:текущий_кадр])
            данные['линия_y2'].set_data(все_времена[:текущий_кадр], Y2[:текущий_кадр])
            данные['линия_y3'].set_data(все_времена[:текущий_кадр], Y3[:текущий_кадр])
            данные['линия_psi'].set_data(все_времена[:текущий_кадр], данные['значения_psi'][:текущий_кадр])

        ax_param_3d.set_title(f'Сравнение перебора параметров\nВремя: {все_времена[текущий_кадр]:.2f}')

        # Возвращение всех художников для обновления
        художники = []
        for данные in данные_параметров:
            художники.extend([
                данные['линия_3d'],
                данные['линия_y1'],
                данные['линия_y2'],
                данные['линия_y3'],
```

```
            данные['линия_psi']
        ])

    return художники

анимация3 = animation.FuncAnimation(fig4, обновить_перебор_параметров,
                                    frames=range(0, len(все_времена), 10),
                                    interval=30, blit=False)  # blit=False для сложных обновлений

plt.tight_layout()
plt.show()

# ===== 6. ИСПРАВЛЕННАЯ ФИНАЛЬНАЯ ТАБЛИЦА РЕЗУЛЬТАТОВ =====
print("\n" + "="*60)
print("6. ФИНАЛЬНОЕ СВОДНОЕ ОПИСАНИЕ")
print("="*60)
print("\nТаблица сравнения параметров:")
print("-" * 100)
print(f"{'ρ':>5} | {'T₁':>8} | {'Конечный |ψ|':>12} | {'Время уст-ия':>14} | {'Макс |u|':>10} | {'Оценка уст-ти':>16}")
print("-" * 100)

for результат in результаты:
    rho, T1, desc = результат['параметры']
    время_уст_str = f"{результат['время_установления']:.2f}" if результат['время_установления'] else "Н/Д"
```

```
        print(f"{rho:5.2f} | {T1:8.3f} | {результат['конечный_psi']:12.6f} | "
              f"{время_уст_str:>14} | "
              f"{результат['максимальное_управление']:10.4f} | {результат['оценка_устойчивости']:16.6f}")

    print("-" * 100)
    print(f"\nОптимальные параметры: ρ = {rho_опт}, T₁ = {T1_опт}")
    print(f"Лучшая оценка устойчивости: {лучшая_устойчивость:.6f}")

    # ===== 7. ДОПОЛНИТЕЛЬНАЯ СТАТИЧЕСКАЯ ВИЗУАЛИЗАЦИЯ =====
    print("\nСоздание статической визуализации всех переменных состояния...")

    fig5, axes5 = plt.subplots(4, 1, figsize=(14, 12))

    # Графики Y1, Y2, Y3 и ψ для оптимальных параметров
    Y1_опт, Y2_опт, Y3_опт = оптимальный_результат['решение'].y
    t_опт = оптимальный_результат['решение'].t

    axes5[0].plot(t_опт, Y1_опт, 'r-', linewidth=1.5, label='Y₁(t)')
    axes5[0].set_ylabel('Y₁'); axes5[0].set_title(f'Оптимальное управление: ρ={rho_опт}, T₁={T1_опт}')
    axes5[0].legend(); axes5[0].grid(True)

    axes5[1].plot(t_опт, Y2_опт, 'g-', linewidth=1.5, label='Y₂(t)')
```

```
    axes5[1].set_ylabel('Y₂')
    axes5[1].legend(); axes5[1].grid(True)

    axes5[2].plot(t_опт, Y3_опт, 'b-', linewidth=1.5, label='Y₃(t)')
    axes5[2].set_ylabel('Y₃')
    axes5[2].legend(); axes5[2].grid(True)

    axes5[3].plot(t_опт, оптимальный_результат['значения_psi'], 'm-',
linewidth=1.5, label='ψ(t)')
    axes5[3].axhline(y=0, color='black', linestyle='--', alpha=0.5,
label='Цель (ψ=0)')
    axes5[3].set_xlabel('Время');
axes5[3].set_ylabel('Макропеременная ψ')
    axes5[3].legend(); axes5[3].grid(True)

    # Отметить время установления если доступно
    if оптимальный_результат['время_установления']:
        for ax in axes5:

ax.axvline(x=оптимальный_результат['время_установления'],
color='orange',
                   linestyle=':', alpha=0.7, label=f'Время уст-ия:
{оптимальный_результат["время_установления"]:.2f}')

    plt.tight_layout()
    plt.show()

    # ===== 8. ВИЗУАЛИЗАЦИЯ СИГНАЛА УПРАВЛЕНИЯ
=====
```

```
print("\nСоздание визуализации сигнала управления...")

fig6, axes6 = plt.subplots(2, 1, figsize=(14, 8))

# Сигнал управления u₁
axes6[0].plot(t_опт, оптимальный_результат['значения_u'], 'darkorange', linewidth=1.5, label='Сигнал управления u₁')
axes6[0].set_ylabel('Сигнал управления u₁')
axes6[0].set_title(f'Эволюция сигнала управления (Макс |u| = {оптимальный_результат["максимальное_управление"]:.4f})')
axes6[0].legend(); axes6[0].grid(True)

# Комбинированный график: Y₃ и ψ
axes6[1].plot(t_опт, Y3_опт, 'b-', alpha=0.7, linewidth=1, label='Y₃(t)')
axes6[1].plot(t_опт, оптимальный_результат['значения_psi'], 'm--', linewidth=1.5, label='ψ(t) = Y₃ - ρY₂')
axes6[1].axhline(y=0, color='black', linestyle='--', alpha=0.5)
axes6[1].set_xlabel('Время'); axes6[1].set_ylabel('Y₃ и ψ')
axes6[1].set_title('Связь между Y₃ и макропеременной ψ')
axes6[1].legend(); axes6[1].grid(True)

plt.tight_layout()
plt.show()

# ===== 9. ФИНАЛЬНЫЕ ВЫВОДЫ =====
print("\n" + "="*60)
print("7. ФИНАЛЬНЫЕ ВЫВОДЫ ДЛЯ ВАРИАНТА 15")
print("="*60)
```

print("✓ Неуправляемая система проявляет хаотическое поведение")

print(f"✓ Оптимальное управление достигнуто при ρ = {rho_опт}, T_1 = {T1_опт}")

print("✓ Синергетическое управление успешно стабилизирует систему")

print("✓ Макропеременная ψ сходится к нулю, что указывает на устойчивость")

print("✓ Сигнал управления u_1 имеет разумную величину")

print("✓ Система переходит от хаотического к устойчивому поведению")

print("✓ Анализ чувствительности параметров завершен")

print("✓ Продемонстрированы случаи устойчивости/неустойчивости")

print("✓ Все переменные состояния (Y_1, Y_2, Y_3) и ψ визуализированы")

print("✓ Проанализирована эволюция сигнала управления")

else:

print("Устойчивая конфигурация не найдена. Попробуйте другие диапазоны параметров.")

```
if __name__ == "__main__":
    запустить_комплексный_анализ_с_анимациями()
```