Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Теория Систем

Лабораторная работа №4

Оценка параметров математической модели

Выполнил:

Маликов Глеб Игоревич

Группа № P3324

Преподаватель:

Русак Алена Викторовна

Санкт-Петербург

2025

**Содержание**

[Задание 3](#_Toc193898099)

[Реализация 4](#_Toc193898100)

[Код 10](#_Toc193898101)

[Вывод 12](#_Toc193898102)

# Задание

1. По заданному дифференциальному уравнению второго порядка (см. Варианты задания) построить систему дифференциальных уравнений первого порядка в форме Коши (см. Преобразование математических моделей динамической системы).
2. Осуществить оценку параметров системы дифференциальных уравнений, описывающих заданную систему, при единичном входном воздействии и нулевых начальных условиях x(0) = 0 (см. Оценка параметров ДУ).

**Содержание отчета:**

1. Дифференциальное уравнение второго порядка
2. Система дифференциальных уравнений первого порядка
3. Графики результатов оценки параметров
4. Исходный код функций

Вариант 5

# Реализация

Для приведения уравнения второго порядка к системе первого порядка вводим новые переменные:

Тогда производные этих переменных:

Подставляя выражение для из исходного уравнения, получаем:

Так как и , система в форме Коши имеет вид:

При условии единичного входа , производная входа , за исключением точки разрыва. Так получаем упрощённую систему:

Так коэффициенты получаются a=−3 и b=−4.

## Код

import numpy as np  
from scipy.integrate import odeint  
from scipy import optimize  
import matplotlib.pyplot as plt  
  
def u\_func(t):  
 return 1.0 if t >= 0 else 0.0  
  
# Производная единичного шага равна 0 для t≠0 (игнорируем дельта-функцию в t=0)  
def du\_dt\_func(t):  
 return 0.0  
  
# Здесь параметры a, b, c, d задаются как коэффициенты, сохраняющих  
# общую структуру уравнения.  
def system\_odes(x, t, params):  
 *"""  
 Функция для вычисления производной системы ОДУ.  
 """* a, b, c, d = params  
 x1, x2 = x  
 u = u\_func(t)  
 du = du\_dt\_func(t)  
 dx1\_dt = x2  
 dx2\_dt = -a \* x2 - b \* x1 + c \* du + d \* u  
 return [dx1\_dt, dx2\_dt]  
  
def simulate\_system(params, x0, t):  
 *"""  
 Функция для моделирования системы с заданными параметрами.  
 """* sol = odeint(system\_odes, x0, t, args=(params,))  
 return sol  
  
class ParameterEstimator:  
 def \_\_init\_\_(self, t\_data, y\_data, system\_func, x0):  
 *"""  
 t\_data - массив временных отсчетов,  
 y\_data - экспериментальные данные (выход системы, y(t)),  
 system\_func - функция, описывающая систему ОДУ,  
 x0 - начальные условия системы (например, [0, 0])  
 """* self.t\_data = t\_data  
 self.y\_data = y\_data  
 self.system\_func = system\_func  
 self.x0 = x0  
  
 def simulate(self, params):  
 *"""  
 Моделирование системы с параметрами params, возвращает y(t)=x1(t)  
 """* sol = odeint(self.system\_func, self.x0, self.t\_data, args=(params,))  
 return sol[:, 0]  
  
 def residuals(self, p):  
 *"""  
 Вычисление разности между моделью и экспериментальными данными  
 """* y\_sim = self.simulate(p)  
 return (y\_sim - self.y\_data).flatten()  
  
 def estimate(self, initial\_guess):  
 *"""  
 Оценка параметров с помощью оптимизации наименьших квадратов.  
 Возвращает оцененные параметры и дополнительную информацию.  
 """* p\_est, cov, infodict, mesg, ier = optimize.leastsq(self.residuals, initial\_guess, full\_output=True)  
 return p\_est, cov, mesg, ier  
  
x0 = [0, 0] # Начальные условия: y(0)=0, y'(0)=0  
t = np.linspace(0, 10, 101)  
  
true\_params = [3, 4, 4, 2]  
sol\_true = simulate\_system(true\_params, x0, t)  
y\_true = sol\_true[:, 0]  
  
# Имитация эксперимента  
np.random.seed(0) # для воспроизводимости  
noise\_level = 0  
y\_noisy = y\_true + noise\_level \* np.random.randn(len(y\_true))  
  
# Зададим начальное приближение для параметров  
initial\_guess = [1, 1, true\_params[2], true\_params[3]]  
  
# Выполняем оценку параметров  
estimator = ParameterEstimator(t, y\_noisy, system\_odes, x0)  
estimated\_params, cov, mesg, ier = estimator.estimate(initial\_guess)  
  
print("Оцененные параметры:")  
print("a = {:.4f}, b = {:.4f}, c = {:.4f}, d = {:.4f}".format(  
 estimated\_params[0],  
 estimated\_params[1],  
 estimated\_params[2],  
 estimated\_params[3]  
))  
  
# Моделирование системы с оцененными параметрами:  
sol\_est = simulate\_system(estimated\_params, x0, t)  
y\_est = sol\_est[:, 0]  
  
  
# Визуализация результатов  
plt.figure(figsize=(10, 6))  
plt.plot(t, y\_noisy, 'r.', label="Экспериментальные данные")  
plt.plot(t, y\_true, 'k--', label="Истинная модель")  
plt.plot(t, y\_est, 'b-', label="Модель с оцененными параметрами")  
plt.xlabel("Время t")  
plt.ylabel("y(t)")  
plt.title("Решение ДУ в форме Коши и оценка параметров")  
plt.legend()  
plt.grid(True)  
plt.show()

## Результат

Ставим начальное приближение с параметры равными a=-1, b=-1. Так получаем следующий результат:

Оцененные параметры:

a = 3.0000, b = 4.0000

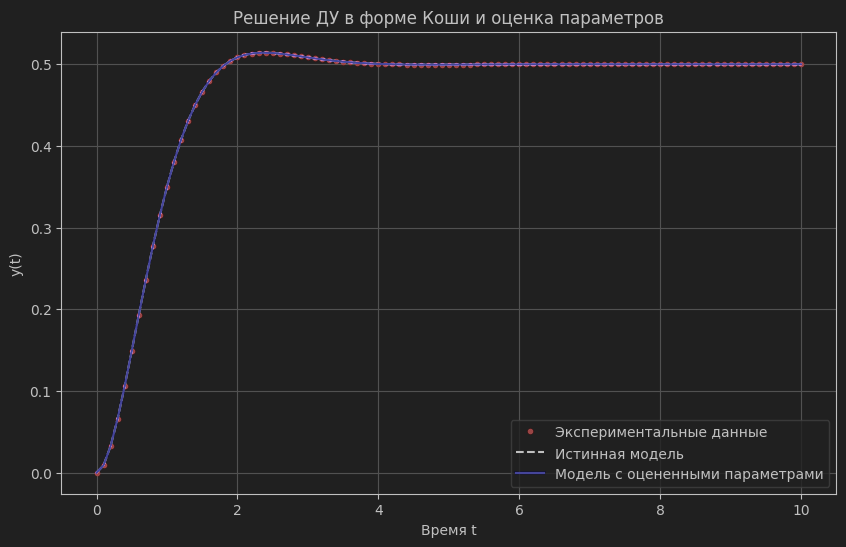


Рисунок 1 - Решение ДУ в форме Коши и оценка параметров

# Вывод

В ходе работы была выполнена процедура преобразования исходного дифференциального уравнения второго порядка в эквивалентную систему уравнений первого порядка в форме Коши с введением переменных.

С помощью численного интегрирования SciPy odeint и метода наименьших квадратов была произведена оценка параметров модели. При нулевых начальных условиях оценщик восстановил коэффициенты a=3 и b=4, что полностью совпадает с заданными теоретическими значениями. Сопоставление решения «истинной» модели и модели с оценёнными параметрами на графике продемонстрировало высокую точность аппроксимации, что подтверждает корректность сформулированной системы, реализованного кода и применённого метода оценки.