**ИДЗ №8  
Задача регрессии в случае нелинейной зависимости y от коэффициентов**

**Вариант №7**

*Работу выполнили:*Батманов Даниил, P3207

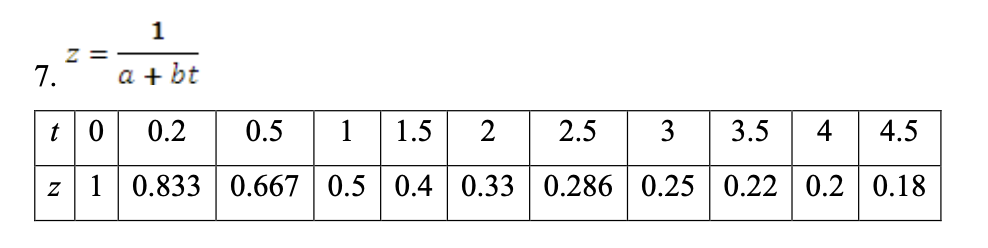
Шорников Сергей, P3211

**Цель работы:**

На основании анализа двумерной выборки:

Решить задачу регрессии в случае нелинейной зависимости y от коэффициентов.

**Исходные данные:**

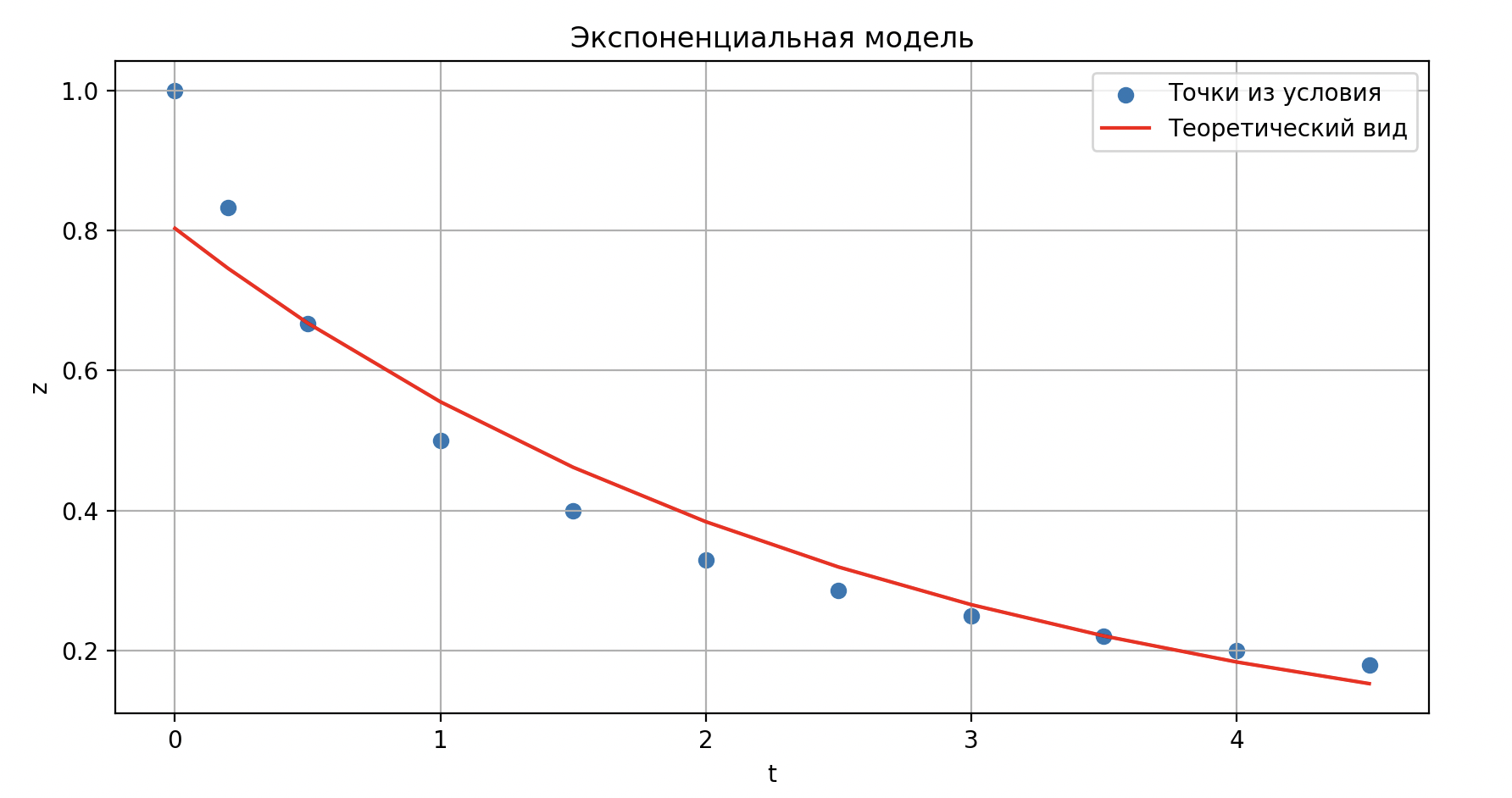


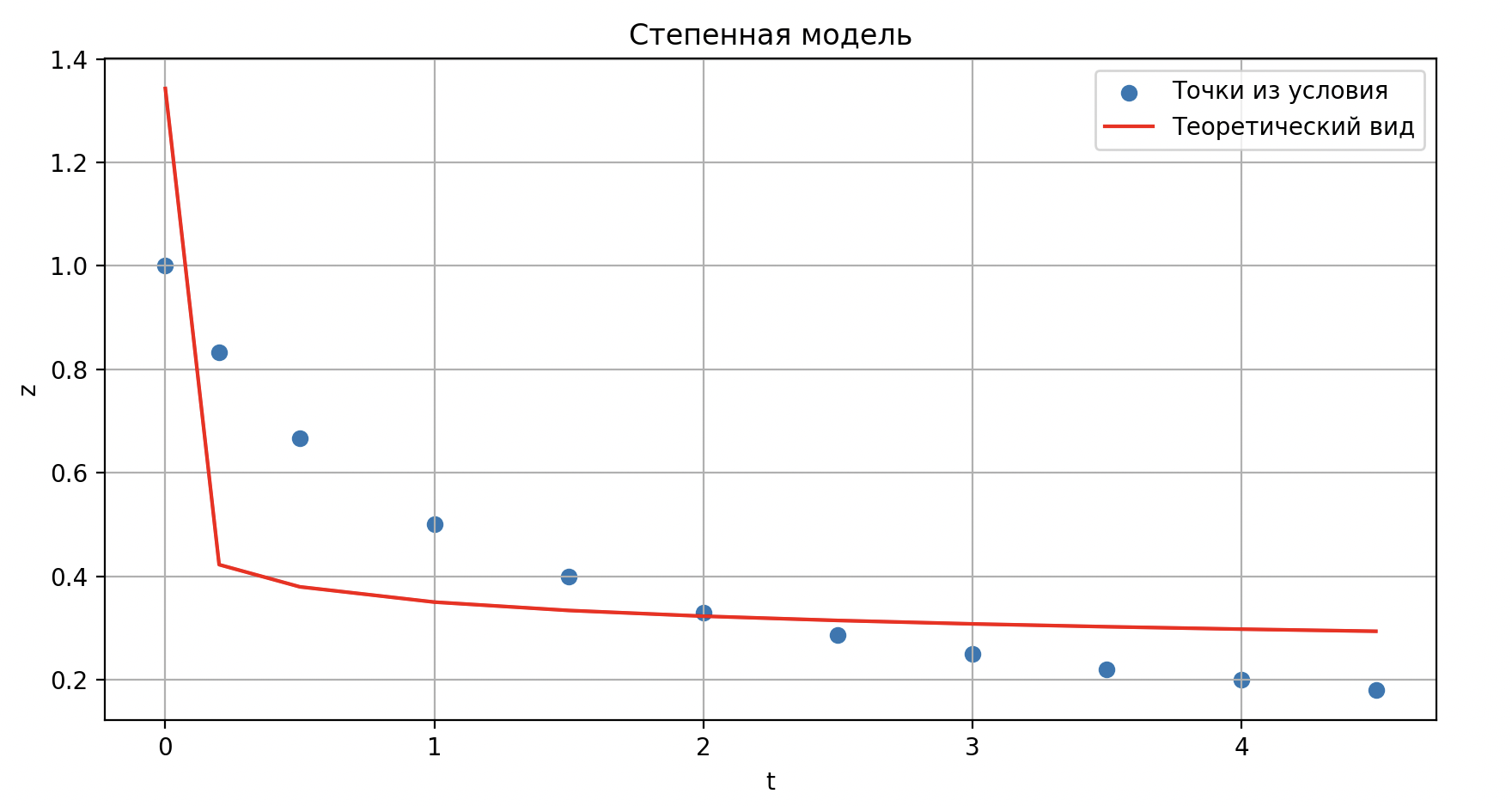
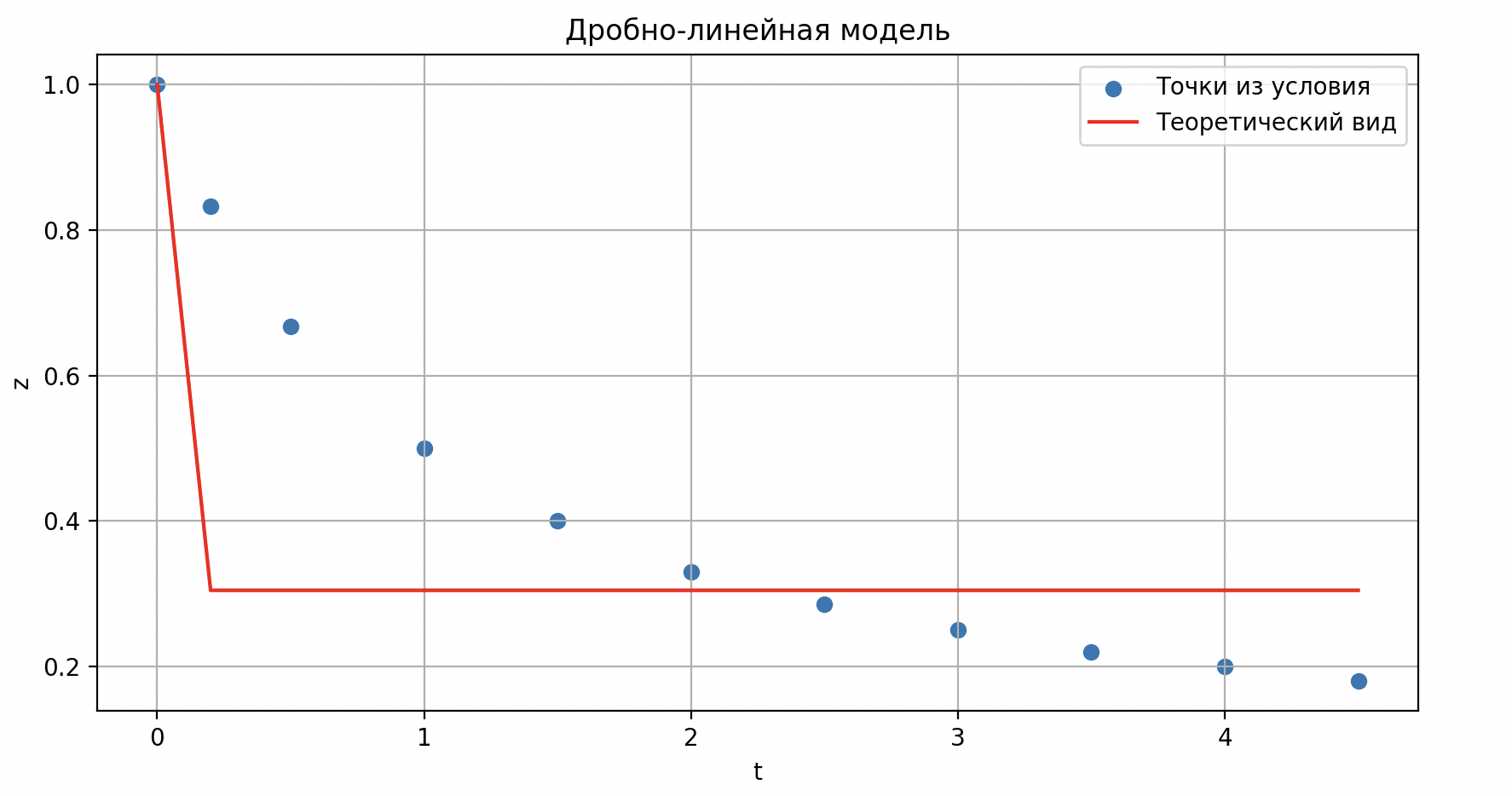
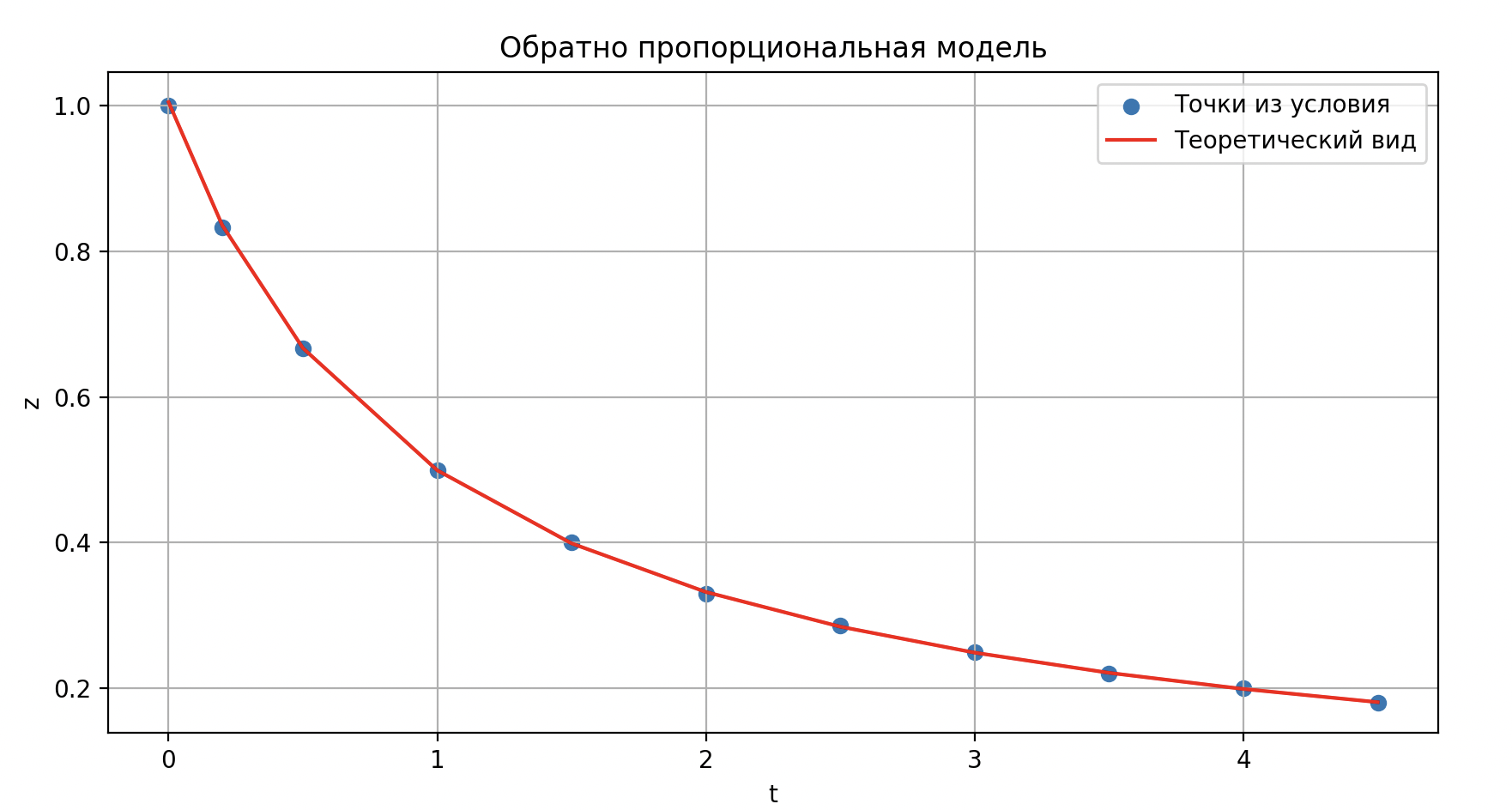
**Ход работы:**

**Исходный код:**

import numpy as np  
import matplotlib.pyplot as plt  
  
t\_start = np.array([0.00001, 0.2, 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5])  
z\_start = np.array([1, 0.833, 0.667, 0.5, 0.4, 0.33, 0.286, 0.25, 0.22, 0.2, 0.18])  
  
  
def draw\_graph(t: np.ndarray, z: np.ndarray, y\_theor: np.ndarray, title: str):  
 plt.figure(figsize=(10, 5))  
 plt.scatter(t, z, marker='o', label="Точки из условия")  
 plt.plot(t, y\_theor, label="Теоретический вид", color='red')  
 plt.title(title)  
 plt.xlabel('t')  
 plt.ylabel('z')  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
  
def perform\_regression(x: np.ndarray, y: np.ndarray) -> tuple:  
 \_x = np.mean(x)  
 \_y = np.mean(y)  
 \_xy = np.mean(x \* y)  
 \_x2 = np.mean(x \*\* 2)  
  
 b = (\_xy - \_x \* \_y) / (\_x2 - \_x \*\* 2)  
 a = \_y - b \* \_x  
 return a, b  
  
  
def exponential(t: np.ndarray, z: np.ndarray):  
 x = t  
 y = np.log(z)  
 a, b = perform\_regression(x, y)  
 y\_theory = np.exp(a + b \* x)  
 draw\_graph(t, z, y\_theory, "Экспоненциальная модель")  
 return np.sum(np.abs(z - y\_theory) \*\* 2)  
  
  
def power\_law(t: np.ndarray, z: np.ndarray):  
 x = np.log(t)  
 y = np.log(z)  
 a, b = perform\_regression(x, y)  
 y\_theory = np.exp(a) \* t \*\* b  
 draw\_graph(t, z, y\_theory, "Степенная модель")  
 return np.sum(np.abs(z - y\_theory) \*\* 2)  
  
  
def frac\_lin(t: np.ndarray, z: np.ndarray):  
 x = 1 / t  
 y = 1 / z  
 a, b = perform\_regression(x, y)  
 y\_theory = 1 / (a + b \* x)  
 draw\_graph(t, z, y\_theory, "Дробно-линейная модель")  
 return np.sum(np.abs(z - y\_theory) \*\* 2)  
  
  
def inverse\_prop(t: np.ndarray, z: np.ndarray):  
 x = t  
 y = 1 / z  
 a, b = perform\_regression(x, y)  
 y\_theory = 1 / (a + b \* x)  
 draw\_graph(t, z, y\_theory, "Обратно пропорциональная модель")  
 return np.sum(np.abs(z - y\_theory) \*\* 2)  
  
  
print("Сумма квадратов отклонений для экспоненциальной модели:", exponential(t\_start, z\_start))  
print("Сумма квадратов отклонений для степенной модели:", power\_law(t\_start, z\_start))  
print("Сумма квадратов отклонений для дробно-линейной модели:", frac\_lin(t\_start, z\_start))  
print("Сумма квадратов отклонений для обратно пропорциональной модели:", inverse\_prop(t\_start, z\_start))  
  
ss = [exponential(t\_start, z\_start), power\_law(t\_start, z\_start), frac\_lin(t\_start, z\_start), inverse\_prop(t\_start, z\_start)]  
names = ["экспоненциальная", "степенная", "дробно-линейная", "обратно пропорциональная"]  
  
print(f"Наилучшая модель - {names[ss.index(min(ss))]}")

**Графики:**



**Вывод программы:**

Сумма квадратов отклонений для экспоненциальной модели: 0.05849410857535555

Сумма квадратов отклонений для степенной модели: 0.42930437831088675

Сумма квадратов отклонений для дробно-линейной модели: 0.4953728997310387

Сумма квадратов отклонений для обратно пропорциональной модели: 4.642995664524906e-05

Наилучшая модель - обратно пропорциональная

**Вывод:**

В ходе лабораторной работы мы решили задачу регрессии в случае нелинейной зависимости y от коэффициентов.