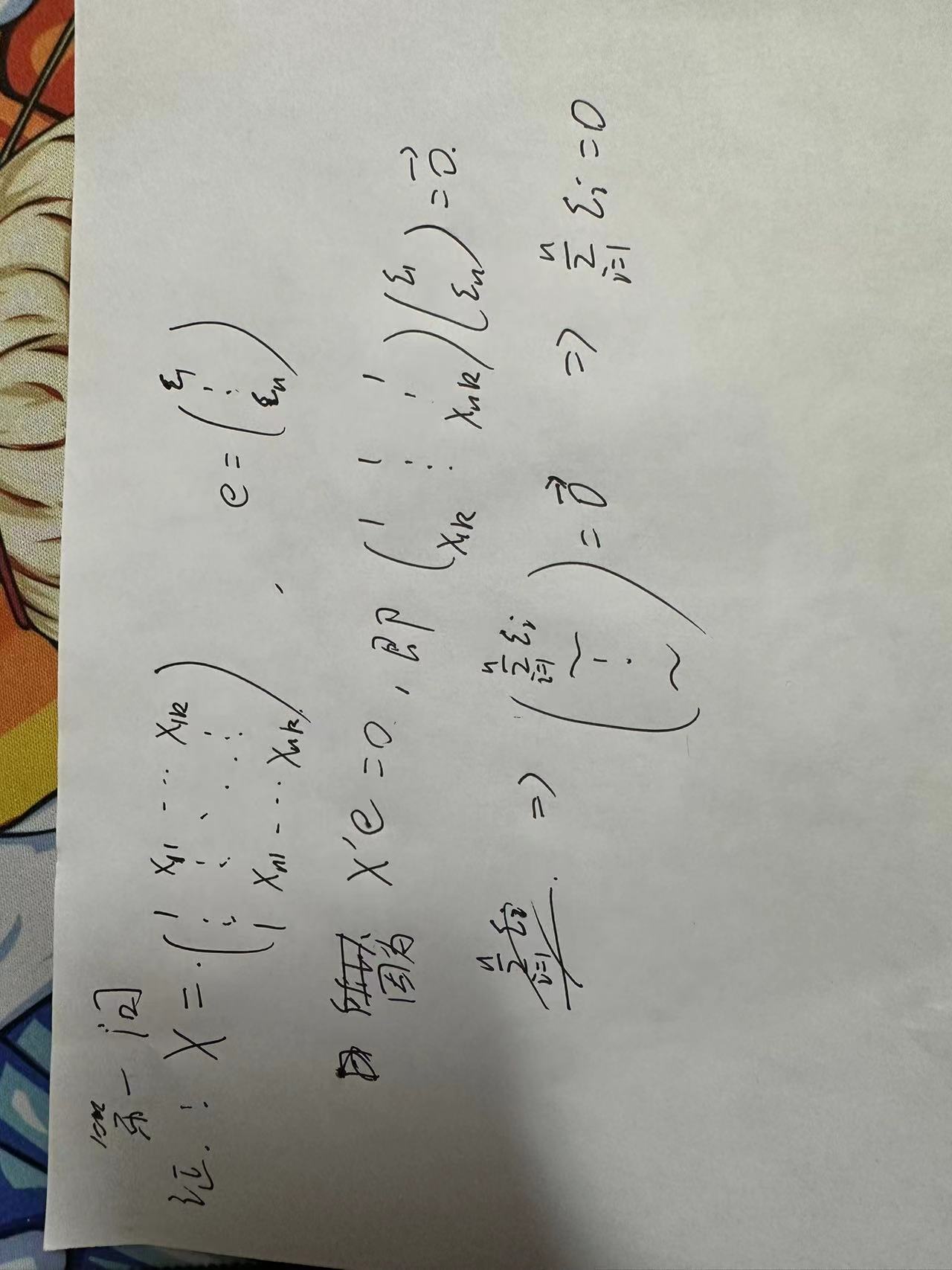
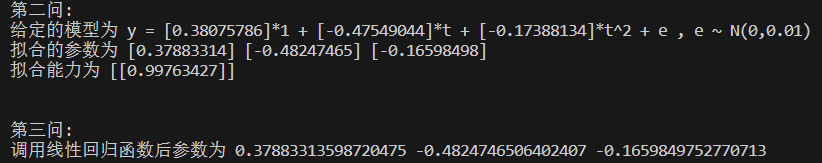
题目：





代码（此代码为python代码，在此以文本形式列出，建议使用同附带的可执行py文件，文件内代码一致）：

import random

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# y = a\*x0 + b\*x1 + c\*x2 + e , x0 = 1, x1 = t, x2 = t^2

# Generate three random integers as a, b, c between -0.5 and 0.5

a = np.random.random(1)-0.5

b = np.random.random(1)-0.5

c = np.random.random(1)-0.5

beta = np.vstack((a, b, c))

print("第二问:")

print(f"给定的模型为 y = {a}\*1 + {b}\*t + {c}\*t^2 + e , e ~ N(0,0.01)")

# Generate T between 1 and 10

T = np.random.random(50)

T = np.sort(T)

T = np.vstack((np.ones(50),T,T\*\*2))

T = np.transpose(T)

# Generate E = [e1, e2, e3, ... , e50]

E = [random.gauss(0, 0.01) for \_ in range(50)]

E = np.array(E)

E = E.reshape(50,1)

# Generate Y\_real

Y = np.dot(T,beta) + E

# calculate beta vertor : beta\_fitting = (X'X)^(-1)X'Y

beta\_fitting = np.dot(np.linalg.inv(np.dot(np.transpose(T),T)), np.dot(np.transpose(T),Y))

Y\_fitting = np.dot(T,beta\_fitting)

print(f"拟合的参数为 {beta\_fitting[0]} {beta\_fitting[1]} {beta\_fitting[2]}")

fitting\_ability = np.dot(np.transpose(Y\_fitting),Y\_fitting)/np.dot(np.transpose(Y),Y)

print(f"拟合能力为 {fitting\_ability}")

# 调用python自带工具进行线性回归

model = LinearRegression().fit(T, Y)

print("\n")

print("第三问:")

print(f"调用线性回归函数后参数为 {model.intercept\_[0]} {model.coef\_[0][1]} {model.coef\_[0][2]}")

Y\_fitting2 = np.dot(T, np.transpose(model.coef\_)) + model.intercept\_

plt.ylabel('y')

plt.xlabel('t')

plt.scatter(T[:,1],Y)

# 第二问计算结果可视化

plt.plot(T[:,1], Y\_fitting, color='r')

# 第三问计算结果可视化

plt.plot(T[:,1], Y\_fitting2, color='r')

plt.show()