|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Практическое задание 1-2 | | |
| по дисциплине «Статистические методы анализа данных» | | |
| **Генерация экспериментальных данных. Оценивание параметров регрессионной модели по методу наименьших квадратов.** | | |
|  | | |
|  |  |  |
| Группа ПМ-13 | Вострецова екатерина |
| Группа ПМ-14 | зиянуров артём |
| Вариант 5 | хамитова екатерина |
|  |  |
|  |  |
| Преподаватели | Попов александр александрович |
|  |  |
| Новосибирск,2024 | | |

# **Постановка задачи**

1. В соответствии с вариантом задания выбрать имитационную модель объекта, диапазон изменения факторов, план эксперимента.

2. Написать программу по генерации экспериментальных данных. Полученные по программе данные оформить в виде одного или двух файлов унифицированной структуры, доступных для дальнейшей обработки. Построить графики зависимости незашумленного отклика от входных факторов.

3. Оформить отчет, включающий в себя постановку задачи, обоснование принятых решений по выбору модели, порождающей данные, графики зависимости незашумленного отклика от входных факторов, сгенерированную выборку наблюдений в виде таблицы, характеристики помехи, текст программы.

4. Спроектировать и сформировать программные модули по вычислению МНK-оценок параметров для заданной параметрической модели объекта. Предусмотреть достаточно простой способ настройки программы на необходимый вид (структуру) модели.

5. Пользуясь экспериментальными данными, полученными в лабораторной работе № 1, оценить параметры модели объекта.

6. Проверить адекватность полученной модели. В качестве можно взять величину дисперсии , которая использовалась при зашумлении отклика в лабораторной работе № 1. Число степеней свободы .

7. Включить в отчет постановочную часть в табличной форме выборку данных и в дополнения к ним значения а также значения , текст программы, принятые решения по проверке адекватности модели.

**Вариант 5**

Произвести моделирование объекта, о котором известно: число факторов – два. По первому фактору зависимость выхода близка к линейной (возрастающей), по второму фактору зависимость близка к параболической. Первый фактор в эксперименте может варьироваться на четырех уровнях (принимать только четыре разрешенных значений), а второй на пяти уровнях. Максимальное значение отклика приходится на внутреннюю точку области действия второго фактора.

# **Ход работы**

Построим линейную имитационную модель. Так как зависимость выхода по первому фактору близка к линейной, то выберем достаточно небольшое значение параметра при этом факторе.

Запишем уравнение и зададим области определения для обоих факторов в соответствии с заданными уровнями:

Рассмотрим графики зависимости:

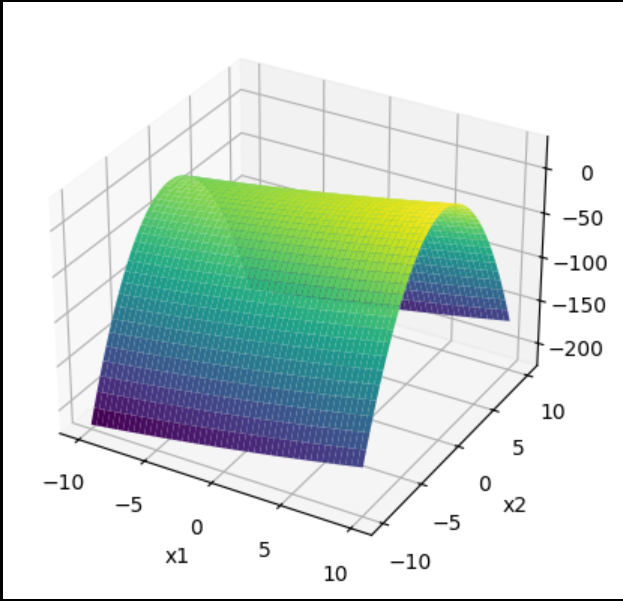


График зависимости незашумлённого отклика от фактора x1

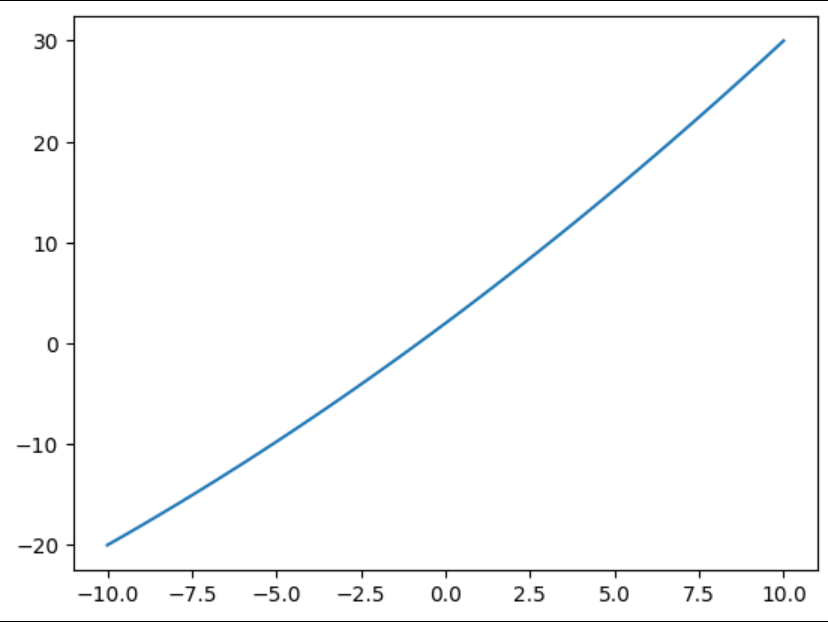
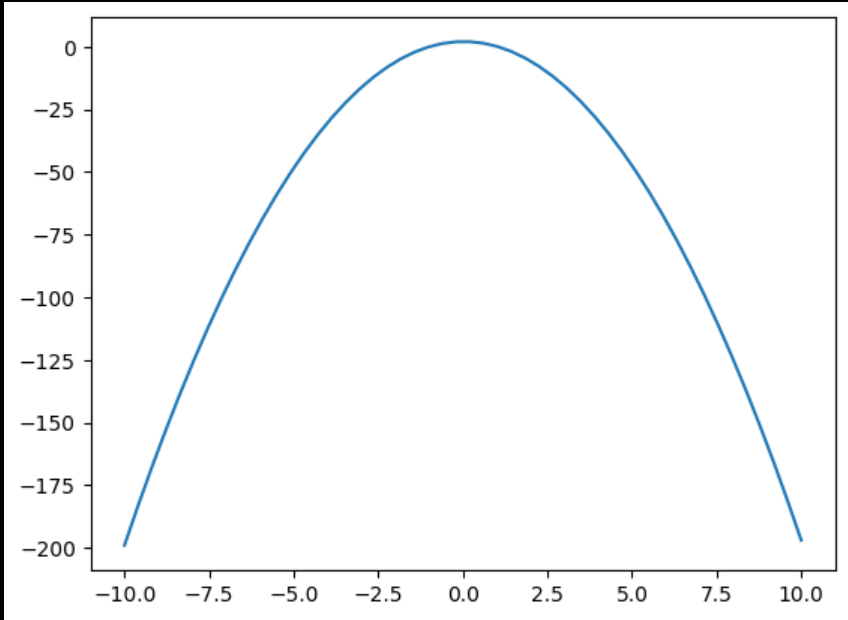
**

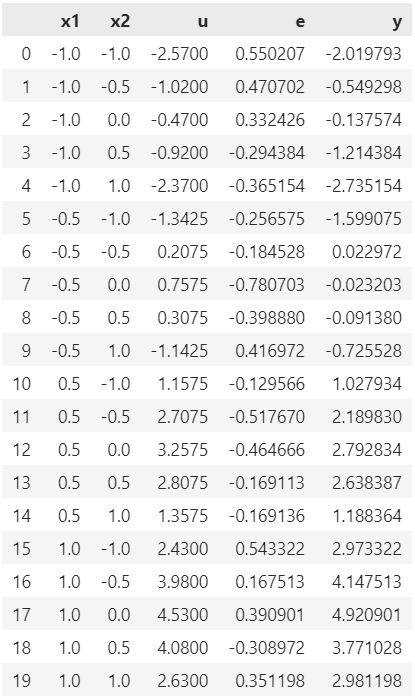
График зависимости незашумлённого отклика от фактора x2

**

**Установка значений параметров**

Значения ошибок наблюдений генерируются по нормальному закону с нулевым математическим ожиданием и дисперсией, а затем прибавляются к истинными значениям . Для нахождения значения стандартного отклонения необходимо найти мощность сигнала .

В результате получим таблицу:



**Оценка параметров при использовании метода наименьших квадратов**

С помощью метода наименьших квадратов можно получить решение нормального уравнения, представленного в виде вектора параметров модели:

Новая оценка σ:

Полученные оценки:





F = 2.1827388051822756

FT = 2.0102414060038662

F < FT: False

Изначально модель квадратичная, а мы пытаемся приблизить линейными базисными функциями, поэтому гипотеза об адекватности отвергается.

# **Код программы**

import numpy as np

import pandas as pd

import scipy.stats

from matplotlib import pyplot as plt

# функция u

def u\_func(x1, x2):

    return 2 + 2.5\*x1 + 0.0001\*x1\*x1 + 0.1\*x2 - 2\*x2\*x2

# создаем массивы x1, x2 и u и заполняем их

x1 = np.array([-1, -0.5, 0.5, 1])

x2 = np.array([-1, -0.5, 0, 0.5, 1])

u = np.array([])

for i in x1:

    for j in x2:

        u = np.append(u, u\_func(i, j))

u

# задаем параметры для дальнейших действий

n = len(u)

p = 0.1 # доля

u\_average = np.full(n, np.mean(u))

# считаем ошибку наблюдения

w\_sq = np.dot((u - u\_average).transpose(), (u - u\_average)) / (n - 1)#сигнал

d = p \* w\_sq#дисперсия

e = np.random.normal(0, d, n)

e

# генерируем y по формуле y = u + e

y = u + e

y

%matplotlib inline

# строим график зависимости незашумленного отклика от входных факторов

x1\_points = np.linspace(-10, 10, 50)

x2\_points = np.linspace(-10, 10, 50)

x1\_grid, x2\_grid = np.meshgrid(x1\_points, x2\_points)

u\_grid = u\_func(x1\_grid, x2\_grid)

fig = plt.figure()

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

ax.plot\_surface(x1\_grid, x2\_grid, u\_grid, cmap='viridis')

ax.set\_xlabel('x1')

ax.set\_ylabel('x2')

plt.show()

# построим еще пару простых графиков зависимости u от факторов

u\_grid = u\_func(x1\_points, 0)

plt.plot(x1\_points, u\_grid)

plt.show()

u\_grid = u\_func(0, x2\_points)

plt.plot(x2\_points, u\_grid)

plt.show()

# создаем дф с данными и формируем файл

df = pd.DataFrame({'x1': np.repeat(x1, len(x2)),

                   'x2': np.tile(x2, len(x1)),

                   'u': u,

                   'e': e,

                   'y': y})

df

df.to\_csv('lab\_1\_data.csv')

class MyLinearRegression:

    def \_\_init\_\_(self):

        self.coef\_ = None

        self.intercept\_ = None

        self.k = None

    def fit(self, X, y):

        X = np.array(X)

        y = np.array(y)

        X = np.hstack((np.ones((X.shape[0], 1)), X))

        self.k = np.linalg.inv(X.T @ X) @ X.T @ y

        self.coef\_ = self.k[::][1:]

        self.intercept\_ = self.k[::][0]

    def predict(self, X):

        X = np.hstack((np.ones((X.shape[0], 1)), X))

        y\_pred = X @ self.k

        return y\_pred

    def get\_param(self):

        return self.coef\_, self.intercept\_

LinModel = MyLinearRegression()

LinModel.fit(x,y)

y\_predict = LinModel.predict(x)

df['y\*']=y\_predict

LinModel.get\_param()

x = df[['x1','x2']]

y = df['y']

df['y-y\*']=df['y']-df['y\*']

df

from sklearn.linear\_model import LinearRegression

SKLinReg = LinearRegression()

SKLinReg.fit(x,y)

SKLinReg.coef\_, SKLinReg.intercept\_

sigma = np.sqrt(p\*w\_sq)

e\_predict = y - y\_predict

sigma\_sq\_predict = e\_predict.T @ e\_predict / (n - 3)

F = sigma\_sq\_predict/sigma\*\*2

FT = scipy.stats.f.ppf(q=1-0.05, dfn=9999, dfd=16)

print(sigma\_sq\_predict)

print("sigma\_e",sigma\_sq\_predict)

print('F =', F)

print('FT =', FT)

print('F < FT:', F < FT)