|  |  |  |
| --- | --- | --- |
|  | | |
| 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** | | |
|  | | |
|  | Бригада 3 | ПМ-13 Буданцев Дмитрий |
| Вариант 3 | ПМ-13 Форкин кирилл |
|  |  |
|  |  |
|  |  |
|  |  |
| Преподаватель | Попов Александр Александрович |
|  |  |
| Новосибирск, 2024 | | |

**Вариант 3**

Провести моделирование объекта, о котором известно: число действующих факторов – три; по всем факторам зависимость выхода близка к линейной, взаимодействия первого фактора со вторым и третьим существенны, т. е. соответствующие параметры при регрессорах , значительно отличаются от нулевого значения.

1. **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. 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.
   4. Пользуясь экспериментальными данными, полученными в лабораторной работе №1, оценить параметры модели объекта.
   5. Проверить адекватность полученной модели.
2. **Ход работы**

Построим линейную имитационную модель в соответствии с вариантом задания. Так как по всем трём факторам зависимость выхода близка к линейной, параметры при регрессорах , и будут близки к единице. Так как взаимодействия первого фактора со вторым и третьим существенны, поэтому соответствующие параметры при регрессорах р , значительно отличаются от нулевого значения.

Количество оцениваемых параметров модели равно пяти. Необходимое число экспериментов в соответствии с эмпирической оценкой должно быть не менее 20. В нашем случае будет 27 экспериментов. Точки расположили равномерно в соответствующих диапазонах возможного изменения факторов :

**Графики зависимости незашумлённого отклика от входных факторов**

|  |  |  |
| --- | --- | --- |
| Зависимость от : | Зависимость от : | Зависимость от : |
|  | Изображение выглядит как текст, линия, График, снимок экрана  Автоматически созданное описание | Изображение выглядит как текст, линия, снимок экрана, График  Автоматически созданное описание |

**Выбор характеристик случайных величин**

Учитываем требование о нормальном распределении ошибки наблюдения . Моделируемая помеха должна иметь нулевое математическое ожидание и дисперсию . Дисперсию будет равна доле от от мощности сигнала . Долю взяли равной 0.12 (12%). Мощность сигнала рассчитывается по следующей формуле:

Дисперсия равна:

Учитывая, что

Получаем значения, которые будут использоваться в работе далее:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **x1** | **x2** | **x3** | **u** | **eta** | **y** |
| -1 | -1 | -1 | 44 | 5,63 | 49,63 |
| -1 | -1 | 0 | 17 | -13,4 | 3,6 |
| -1 | -1 | 1 | -10 | 1,51 | -8,49 |
| -1 | 0 | -1 | 26 | -1,13 | 24,87 |
| -1 | 0 | 0 | -1 | 14 | 13 |
| -1 | 0 | 1 | -28 | -11,74 | -39,74 |
| -1 | 1 | -1 | 8 | 3,72 | 11,72 |
| -1 | 1 | 0 | -19 | 12,17 | -6,83 |
| -1 | 1 | 1 | -46 | -8,85 | -54,85 |
| 0 | -1 | -1 | -5 | -4,82 | -9,82 |
| 0 | -1 | 0 | -2 | 7,79 | 5,79 |
| 0 | -1 | 1 | 1 | 4,67 | 5,67 |
| 0 | 0 | -1 | -3 | 3,45 | 0,45 |
| 0 | 0 | 0 | 0 | -11,94 | -11,94 |
| 0 | 0 | 1 | 3 | -8,5 | -5,5 |
| 0 | 1 | -1 | -1 | -3,12 | -4,12 |
| 0 | 1 | 0 | 2 | 6,64 | 8,64 |
| 0 | 1 | 1 | 5 | -3,79 | 1,21 |
| 1 | -1 | -1 | -54 | -3,62 | -57,62 |
| 1 | -1 | 0 | -21 | -10,89 | -31,89 |
| 1 | -1 | 1 | 12 | 0,00 | 12 |
| 1 | 0 | -1 | -32 | 4,88 | -27,12 |
| 1 | 0 | 0 | 1 | -24,21 | -23,21 |
| 1 | 0 | 1 | 34 | -11,41 | 22,59 |
| 1 | 1 | -1 | -10 | 16,94 | 6,94 |
| 1 | 1 | 0 | 23 | 12,79 | 35,79 |
| 1 | 1 | 1 | 56 | 3,73 | 59,73 |

**Вычисление МНК-оценки**

МНК-оценка вычисляется по формуле:

Вектор остатков вычисляется по формуле:

|  |  |  |
| --- | --- | --- |
| **y** | **y^** | **y-y^** |
| 49,63 | 47,93 | 1,7 |
| 3,6 | 17,68 | -14,09 |
| -8,49 | -12,56 | 4,07 |
| 24,87 | 30,01 | -5,14 |
| 13 | -0,24 | 13,24 |
| -39,74 | -30,49 | -9,25 |
| 11,72 | 12,09 | -0,36 |
| -6,83 | -18,16 | 11,33 |
| -54,85 | -48,41 | -6,44 |
| -9,82 | -4,84 | -4,98 |
| 5,79 | -4,97 | 10,75 |
| 5,67 | -5,09 | 10,77 |
| 0,45 | 0,13 | 0,32 |
| -11,94 | 0 | -11,94 |
| -5,5 | -0,13 | -5,37 |
| -4,12 | 5,09 | -9,21 |
| 8,64 | 4,97 | 3,67 |
| 1,21 | 4,84 | -3,62 |
| -57,62 | -57,6 | -0,02 |
| -31,89 | -27,61 | -4,28 |
| 12 | 2,37 | 9,62 |
| -27,12 | -29,75 | 2,63 |
| -23,21 | 0,24 | -23,45 |
| 22,59 | 30,23 | -7,64 |
| 6,94 | -1,9 | 8,83 |
| 35,79 | 28,09 | 7,7 |
| 59,73 | 58,08 | 1,65 |

А несмещённая оценка неизвестной дисперсии равна:

**Проверка модели на адекватность**

Модель не признаётся неадекватной ( не отвергается), если соотношение

истинно.

В качестве оценки дисперсии взята оценка дисперсии , которая использовалась для зашумления отклика:

Квантиль распределения Фишера при уровне значимости , , равен:

Гипотеза не отвергается.

1. **Код программы**

import pandas as pd

import numpy as np

from random import uniform

import matplotlib.pyplot as plt

import os

from math import sqrt

def Ita(Teta : np.array, x: np.array) -> float:

    """

    Имитационная модель. Линейная по параметрам тета.\n

    param Teta - (np.array) - весовые переменные (вклад входной переменной)\n

    param x - (np.array) - входной вектор\n

    return: (float)

    """

    if x.shape[0] != 3 or Teta.shape[0] != 5:

        raise ValueError("Размер не соответствует условию")

    return Teta[0]\*x[0] + Teta[1]\*x[1] + Teta[2]\*x[2] + Teta[3]\*x[0]\*x[1] + Teta[4]\*x[0]\*x[2]

def y\_response(x : np.array, Teta : np.array, eta : float) -> float:

    """

    Функция отклика\n

    param x - (np.array) - входной вектор\n

    param Teta - (np.array) - весовые переменные (вклад входной переменной)\n

    param eta - (float) - Ошибка наблюдения

    """

    if x.shape[0] != 3 or Teta.shape[0] != 5:

        raise ValueError("Размер не соответствует условию")

    return Ita(Teta, x) + eta

def generator\_x(x\_min : float = 0, x\_max : float = 1, len\_x : int = 3) -> np.array:

    """

    Генератор вектора x по равномерному закону заданной длины\n

    param x\_min : (float) - минимальное значение\n

    param x\_max : (float) - максимальное значение\n

    param len\_x : (int) - длина вектора

    return: (np.array)

    """

    if x\_min > x\_max:

        raise ValueError("x\_min > x\_max")

    return np.array([uniform(x\_min, x\_max) for i in range(len\_x)])

def generator\_sample\_x(x\_min : float = 0, x\_max : float = 1, len\_x : int = 3) -> np.array:

    """

    Сгенерировать значения равномерно по отрезку

    param x\_min : (float) - минимальное значение\n

    param x\_max : (float) - максимальное значение\n

    param len\_x : (int) - длина вектора

    return: (np.array)

    """

    return np.arange(x\_min, x\_max+1e-7, (x\_max - x\_min)/(len\_x - 1))

def generator\_deviation(eta : float, len\_list : int) -> np.array:

    if len\_list < 0:

        raise ValueError("Длина не может быть отрицательной")

    if len\_list == 0:

        return np.array([])

    return np.random.normal(0, sqrt(eta), len\_list)

x1 = generator\_sample\_x(-1, 1, 3)

x = np.linspace(-1, 1)

y = [Ita(np.array([1, 2, 3, 20, 30]), np.array([0, 0, x3])) for x3 in x]

plt.plot(x, y)

plt.show()

def generator\_x(x\_min : float = 0, x\_max : float = 1, len\_x : int = 3) -> np.array:

    """

    Генератор вектора x по равномерному закону заданной длины\n

    param x\_min : (float) - минимальное значение\n

    param x\_max : (float) - максимальное значение\n

    param len\_x : (int) - длина вектора

    return: (np.array)

    """

    if x\_min > x\_max:

        raise ValueError("x\_min > x\_max")

    return np.array([uniform(x\_min, x\_max) for i in range(len\_x)])

def generator\_sample\_x(x\_min : float = 0, x\_max : float = 1, len\_x : int = 3) -> np.array:

    """

    Сгенерировать значения равномерно по отрезку

    param x\_min : (float) - минимальное значение\n

    param x\_max : (float) - максимальное значение\n

    param len\_x : (int) - длина вектора

    return: (np.array)

    """

    return np.arange(x\_min, x\_max, (x\_max - x\_min)/(len\_x - 1))

x1 = generator\_sample\_x(-1, 1, 3)

x1

def generator\_x(x\_min : float = 0, x\_max : float = 1, len\_x : int = 3) -> np.array:

    """

    Генератор вектора x по равномерному закону заданной длины\n

    param x\_min : (float) - минимальное значение\n

    param x\_max : (float) - максимальное значение\n

    param len\_x : (int) - длина вектора

    return: (np.array)

    """

    if x\_min > x\_max:

        raise ValueError("x\_min > x\_max")

    return np.array([uniform(x\_min, x\_max) for i in range(len\_x)])

def Calc\_eta(list\_u : np.array, procent : float) -> float:

    if procent <= 0 and procent > 1:

        raise ValueError("Процент находится в диапозоне от 0 до 1")

    mean\_u = np.ones(list\_u.shape[0]) \* list\_u.mean()

    temp = list\_u - mean\_u

    w = temp.T @ temp / (list\_u.shape[0] - 1)

    return procent\* w

def gen\_test() -> list:

    x1 = generator\_sample\_x(-1, 1, 3)

    x2 = generator\_sample\_x(-1, 1, 3)

    x3 = generator\_sample\_x(-1, 1, 3)

    result = []

    for x in x1:

        for y in x2:

            for z in x3:

                result.append(np.array([x, y, z]))

    return result

x\_test = gen\_test()

list\_u = np.array([Ita(np.array([1, 2, 3, 20, 30]), x) for x in x\_test])

eta = Calc\_eta(list\_u, 0.12)

list\_deviation = generator\_deviation(eta, len(list\_u))

list\_deviation\_1 = generator\_deviation(eta, len(list\_u))

y\_test = np.array([y\_response(x\_test[i], np.array([1, 2, 3, 20, 30]), list\_deviation\_1[i]) for i in range(len(x\_test))])

df = pd.DataFrame(columns=np.array(["x1", "x2", "x3", "u", "eta", "y"]))

for i in range(len(x\_test)):

    df.loc[len(df)] = np.array([x\_test[i][0], x\_test[i][1], x\_test[i][2], list\_u[i], list\_deviation\_1[i], y\_test[i]])

with pd.ExcelWriter(os.path.abspath(".\Results.xlsx")) as writter:

    df.to\_excel(writter, sheet\_name="Results", float\_format="%.2f", index=False)

def calc\_matrix\_X(list\_x : list) -> np.matrix:

    return np.matrix([[x[0], x[1], x[2], x[0]\*x[1], x[0]\*x[2]] for x in list\_x])

X = calc\_matrix\_X(x\_test)

teta\_calc = (np.linalg.inv(X.T @ X) @ X.T) @ y\_test

teta\_calc = np.asarray(teta\_calc)[0]

teta\_calc

I = np.eye(len(x\_test))

y\_calc = X @ teta\_calc

y\_calc = np.asarray(y\_calc)[0]

e\_calc = y\_test - y\_calc

e\_calc = np.asarray(e\_calc)

eta\_calc = e\_calc.T @ e\_calc / (27 - 5)

F = eta\_calc/eta

df2 = pd.DataFrame(columns=np.array(["y", "y^", "y-y^"]))

for i in range(len(y\_test)):

    df2.loc[len(df2)] = np.array([y\_test[i], y\_calc[i], e\_calc[i]])

with pd.ExcelWriter(os.path.abspath(".\Result\_2.xlsx")) as writter:

    df2.to\_excel(writter, sheet\_name="Result2", index=False, float\_format="%.2f")