Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №6
З дисципліни «Методи наукових досліджень»
За темою:
«Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІВ-91
Гришин О.С.
Номер у списку - 07

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

Мета: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

Завдання:

Завдання до лабораторної роботи:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x_1, x_2, x_3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ l; - l; 0 для $\bar{x}_1, \bar{x}_2, \bar{x}_3$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + random(10)-5,$$

де $f(x_1, x_2, x_3)$ вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів.
5. Зробити висновки по виконаній роботі.

Програмний код

Main.py

```
import sys
from functools import partial
from random import randrange
from pyDOE2 import ccdesign
from Criterium import Criteria
import numpy as np
from numpy.linalg import solve
from prettytable import PrettyTable
from scipy.stats import f, t

if len(sys.argv) == 1:
    m = 3
else:
    m = sys.argv[1]
n = 15

# VAR 107

x1min = -5
x1max = 15
x2min = -15
x2max = 35
x3min = 15
x3max = 30

x01 = (x1max + x1min) / 2
x02 = (x2max + x2min) / 2
x03 = (x3max + x3min) / 2
deltax1 = x1max - x01
deltax2 = x2max - x02
deltax3 = x3max - x03


def function(X1, X2, X3):
    y = 3.9 + 5.6 * X1 + 7.9 * X2 + 7.3 * X3 + 2.0 * X1 * X1 + 0.5 * X2 * X2
+ 4.2 * X3 * X3 + 1.5 * X1 * X2 + \
        0.1 * X1 * X3 + 9.9 * X2 * X3 + 5.3 * X1 * X2 * X3 + randrange(0, 10)
- 5
    return y
```

```
def add_sq_nums(x):
    for i in range(len(x)):
        x[i][4] = x[i][1] * x[i][2]
        x[i][5] = x[i][1] * x[i][3]
        x[i][6] = x[i][2] * x[i][3]
        x[i][7] = x[i][1] * x[i][3] * x[i][2]
        x[i][8] = x[i][1] ** 2
        x[i][9] = x[i][2] ** 2
        x[i][10] = x[i][3] ** 2
    return x


if n > 14:
    no = n - 14
else:
    no = 1
xn = ccdesign(3, center=(0, no))
xn = np.insert(xn, 0, 1, axis=1)

for i in range(4, 11):
    xn = np.insert(xn, i, 0, axis=1)

l = 1.215

for i in range(len(xn)):
    for j in range(len(xn[i])):
        if xn[i][j] < -1 or xn[i][j] > 1:
            if xn[i][j] < 0:
                xn[i][j] = -l
            else:
                xn[i][j] = l

x_norm = add_sq_nums(xn)

x1 = [x1min, x1min, x1min, x1min, x1max, x1max, x1max, x1max, -1.73 * deltax1
+ x01, 1.73 * deltax1 + x01, x01, x01,
      x01, x01, x01]
x2 = [x2min, x2min, x2max, x2max, x2min, x2min, x2max, x2max, x02, x02, -1.73
* deltax2 + x02, 1.73 * deltax2 + x02,
      x02, x02, x02]
x3 = [x3min, x3max, x3min, x3max, x3min, x3max, x3min, x3max, x03, x03, x03,
x03, -1.73 * deltax3 + x03,
      1.73 * deltax3 + x03, x03]
x1x2 = [0] * 15
x1x3 = [0] * 15
x2x3 = [0] * 15
x1x2x3 = [0] * 15
x1kv = [0] * 15
x2kv = [0] * 15
x3kv = [0] * 15
for i in range(15):
    x1x2[i] = x1[i] * x2[i]
    x1x3[i] = x1[i] * x3[i]
    x2x3[i] = x2[i] * x3[i]
    x1x2x3[i] = x1[i] * x2[i] * x3[i]
    x1kv[i] = x1[i] ** 2
    x2kv[i] = x2[i] ** 2
    x3kv[i] = x3[i] ** 2

list_for_a = list(zip(x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3, x1kv, x2kv,
x3kv))

for i in range(len(list_for_a)):
```

```
    list_for_a[i] = list(list_for_a[i])
    for j in range(len(list_for_a[i])):
        list_for_a[i][j] = round(list_for_a[i][j], 3)

planning_matrix_x = PrettyTable()
planning_matrix_x.field_names = ['X1', 'X2', 'X3', 'X1X2', 'X1X3', 'X2X3',
'X1X2X3', 'X1X1', 'X2X2', 'X3X3']
print("Матриця планування з натуралізованими коефіцієнтами X:")
planning_matrix_x.add_rows(list_for_a)
print(planning_matrix_x)

Y = [[function(list_for_a[j][0], list_for_a[j][1], list_for_a[j][2]) for i in
range(m)] for j in range(15)]

planing_matrix_y = PrettyTable()
planing_matrix_y.field_names = ['Y1', 'Y2', 'Y3']
print("Матриця планування Y:")
planing_matrix_y.add_rows(Y)
print(planing_matrix_y)

Y_average = []
for i in range(len(Y)):
    Y_average.append(np.mean(Y[i], axis=0))
print("Середні значення відгуку за рядками:")
print(Y_average)

dispersions = []
for i in range(len(Y)):
    a = 0
    for k in Y[i]:
        a += (k - np.mean(Y[i], axis=0)) ** 2
    dispersions.append(a / len(Y[i]))


def find_known(num):
    a = 0
    for j in range(15):
        a += Y_average[j] * list_for_a[j][num - 1] / 15
    return a


def get_exp_val(k):
    return beta[0] + beta[1] * list_for_a[k][0] + beta[2] * list_for_a[k][1]
+ beta[3] * list_for_a[k][2] + \
           beta[4] * list_for_a[k][3] + beta[5] * list_for_a[k][4] + beta[6]
* list_for_a[k][5] + beta[7] * \
           list_for_a[k][6] + beta[8] * list_for_a[k][7] + beta[9] *
list_for_a[k][8] + beta[10] * list_for_a[k][9]


def a(first, second):
    a = 0
    for j in range(15):
        a += list_for_a[j][first - 1] * list_for_a[j][second - 1] / 15
    return a


my = sum(Y_average) / 15
mx = []
for i in range(10):
    number_lst = []
    for j in range(15):
        number_lst.append(list_for_a[j][i])
    mx.append(sum(number_lst) / len(number_lst))
```

```
det1 = [
    [1, *mx],
    *[[mx[row - 1], *[a(row, col) for col in range(1, 11)]] for row in
range(1, 11)]
]

det2 = [my, *[find_known(num) for num in range(1, 11)]]

beta = solve(det1, det2)

print("\nОтримане рівняння регресії:")
print("{} + {} * X1 + {} * X2 + {} * X3 + {} * X1X2 + {} * X1X3 + {} * X2X3+
{} * X1X2X3 + {} * X11^2 + {} * X22^2 + "
      "{} * X33^2 = ŷ "
      .format(*beta))

print("Експериментальні значення:")
y_i = [get_exp_val(k) for k in range(15)]
print(y_i)

criteria = Criteria(list_for_a, Y, n, m)


print("\n\n------------------------------ Перевірка за критерієм Кохрена ---
----------------------------")
G_kr = criteria.cohren()
Gp = criteria.criteria_cochrana(Y_average)
print(f'Gp = {Gp}')
if Gp < G_kr:
    print(f'З ймовірністю {1 - criteria.q} дисперсії однорідні.')
else:
    print("Необхідно збільшити кількість дослідів")
    m += 1
    new_com = [sys.argv[0] + f" {m}"]
    print([sys.executable] + new_com)

print("\n------------------ Перевірка значущості коефіцієнтів за критерієм
Стьюдента ------------------")
student = partial(t.ppf, q=1 - criteria.q)
t_student = student(df=criteria.f3)
ts = criteria.criteria_studenta(xn[:, 1:], Y_average)
res = [t for t in ts if t > t_student]
final_k = [beta[i] for i in range(len(ts)) if ts[i] in res]
print('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з
рівняння.'.format(
    [round(i, 3) for i in beta if i not in final_k]))
d = len(final_k)
y_st = []
for i in range(15):
    y_st.append(res[0] + res[1] * x1[i] + res[2] * x2[i] + res[3] * x3[i] +
res[4] * x1x2[i] + res[5] *
                x1x3[i] + res[6] * x2x3[i] + res[7] * x1x2x3[i] + res[8] *
x1kv[i] + res[9] *
                x2kv[i] + res[10] * x3kv[i])

print("\n\n------------------------- Перевірка адекватності за критерієм
Фішера -------------------------")
F_p = criteria.criteria_fishera(Y_average, y_st, d)
f4 = n - d
fisher = partial(f.ppf, q=1-criteria.q)
f_t = fisher(dfn=f4, dfd=criteria.f3)

print('\nПеревірка адекватності за критерієм Фішера')
```

```
print('Fp =', F_p)
print('F_t =', f_t)
if len(final_k) == 2:
    print('Математична модель не адекватна експериментальним даним')
else:
    print('Математична модель адекватна експериментальним даним')
```

Criteria.py

```
from scipy.stats import f


class Criteria:
    def __init__(self, x, y, n, m):
        self.x = x
        self.y = y

        self.n = n
        self.m = m
        self.f1 = self.m - 1
        self.f2 = self.n
        self.q = 0.05
        self.q1 = self.q / self.f1

    def s_kv(self, y_aver):
        res = []
        for i in range(self.n):
            s = sum([(y_aver[i] - self.y[i][j]) ** 2 for j in range(self.m)])
/ self.m
            res.append(round(s, 3))
        return res

    def criteria_cochrana(self, y_aver):
        S_kv = self.s_kv(y_aver)
        Gp = max(S_kv) / sum(S_kv)
        print('\nПеревірка за критерієм Кохрена')
        return Gp

    def cohren(self):
        fisher_value = f.ppf(q=1 - self.q1, dfn=self.f2, dfd=(self.f1 - 1) *
self.f2)
        return fisher_value / (fisher_value + self.f1 - 1)

    def bs(self, x, y_aver):
        res = [sum(y_aver) / self.n]

        for i in range(len(x[0])):
            b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / self.n
            res.append(b)

        return res

    def criteria_studenta(self, x, y_aver):
        S_kv = self.s_kv(y_aver)
        s_kv_aver = sum(S_kv) / self.n

        s_Bs = (s_kv_aver / self.n / self.m) ** 0.5
        Bs = self.bs(x, y_aver)
        ts = [round(abs(B) / s_Bs, 3) for B in Bs]

        return ts

    def criteria_fishera(self, y_aver, y_new, d):
        S_ad = self.m / (self.n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i
```

```
in range(len(self.y))])
        S_kv = self.s_kv(y_aver)
        S_kv_aver = sum(S_kv) / self.n

        return S_ad / S_kv_aver
```

Результати роботи програми

Матриця планування з натуралізованими коефіцієнтами X:

X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1X1	X2X2	X3X3
-5	-15	15	75	-75	-225	1125	25	225	225
-5	-15	30	75	-150	-450	2250	25	225	900
-5	35	15	-175	-75	525	-2625	25	1225	225
-5	35	30	-175	-150	1050	-5250	25	1225	900
15	-15	15	-225	225	-225	-3375	225	225	225
15	-15	30	-225	450	-450	-6750	225	225	900
15	35	15	525	225	525	7875	225	1225	225
15	35	30	525	450	1050	15750	225	1225	900
-12.3	10.0	22.5	-123.0	-276.75	225.0	-2767.5	151.29	100.0	506.25
22.3	10.0	22.5	223.0	501.75	225.0	5017.5	497.29	100.0	506.25
5.0	-33.25	22.5	-166.25	112.5	-748.125	-3740.625	25.0	1105.562	506.25
5.0	53.25	22.5	266.25	112.5	1198.125	5990.625	25.0	2835.562	506.25
5.0	10.0	9.525	50.0	47.625	95.25	476.25	25.0	100.0	90.726
5.0	10.0	35.475	50.0	177.375	354.75	1773.75	25.0	100.0	1258.476
5.0	10.0	22.5	50.0	112.5	225.0	1125.0	25.0	100.0	506.25

Матриця планування Y:

Y1	Y2	Y3
4914.4	4913.4	4914.4
11583.4	11586.4	11582.4
-7020.6	-7014.6	-7018.6
-12791.6	-12795.6	-12793.6
-18839.6	-18845.6	-18848.6
-35994.6	-35994.6	-35995.6
50223.4	50221.4	50230.4
100124.4	100127.4	100123.4
-9991.325	-9995.325	-9993.325
32747.785000000003	32748.785000000003	32748.785000000003
-24806.36875	-24812.36875	-24811.36875
48230.23125	48230.23125	48231.23125
4210.342625	4211.342625	4203.342625
18765.102625	18764.102625	18759.102625
10778.65	10776.65	10776.65

```
Середні значення відгуку за рядками:
[4914.066666666667, 11584.066666666666, -7017.933333333334, -12793.6, -18844.6, -35994.93333333333, 50225.06666666667, 100125.06666666665, -

Отримане рівняння регресії:
4.786971483786538 + 5.64820009024805 * X1 + 7.836419125856932 * X2 + 7.217549876693587 * X3 + 1.5056666666656853 * X1X2 + 0.0948888888866610
Експериментальні значення:
[4914.274144425908, 11585.114458649934, -7017.725855573832, -12792.552208016965, -18844.294389372182, -35993.78740848234, 50225.37227729439,
```

```
-------------------------------- Перевірка за критерієм Кохрена --------------------------------
Gp = 0.19999999999999998
З ймовірністю 0.95 дисперсії однорідні.

------------------ Перевірка значущості коефіцієнтів за критерієм Стьюдента ------------------

Коефіцієнти [] статистично незначущі, тому ми виключаємо їх з рівняння.


------------------------- Перевірка адекватності за критерієм Фішера -------------------------

Перевірка адекватності за критерієм Фішера
Fp = 4.301617458681541e+16
F_t = 2.6896275736914177
Математична модель адекватна експериментальним даним
```