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

ЛАБОРАТОРНА РОБОТА № 6

з дисципліни «Методи наукових досліджень»

на тему «Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

ВИКОНАВ:
студент 2 курсу
групи ІВ-91
Бойко М. І.
Залікова – 9102

ПЕРЕВІРИВ:
ас. Регіда П. Г.

Київ – 2021

Лабораторна робота №6

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

Завдання:

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.

Група: ІВ-91

Номер у списку: 2

Варіант – 102

№ варіанту	x_1		x_2		x_3		$f(x_1, x_2, x_3)$
	min	max	min	max	min	max	
102	20	70	-20	40	70	80	2,9+3,9*x1+6,7*x2+4,4*x3+9,7*x1*x1+0,8*x2*x2+7,5*x3*x3+1,0*x1*x2+1,0*x1*x3+0,8*x2*x3+4,2*x1*x2*x3

Роздруківка коду програми:

```
import sys
from functools import partial
from random import randrange
from pyDOE2 import ccdesign
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

x1min = 20
x1max = 70
x2min = -20
x2max = 40
x3min = 70
x3max = 80

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

```
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.f3 = self.f1 * self.f2
        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)
        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


def function(X1, X2, X3):
    y = 2.9 + 3.9 * X1 + 6.7 * X2 + 4.4 * X3 + 9.7 * X1 * X1 + 0.8 * X2 * X2
+ 7.5 * X3 * X3 + 1.0 * X1 * X2 + \
        1.0 * X1 * X3 + 0.8 * X2 * X3 + 4.2 * 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\nПеревіримо 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("\nПеревіримо 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("\nПеревіримо 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('Fp =', F_p)
print('F_t =', f_t)
```

```
if len(final_k) == 2:
    print('Математична модель не адекватна')
else:
    print('Математична модель адекватна')
```

Приклад роботи програми:

Матриця планування з натуралізованими коефіцієнтами X:

X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1X1	X2X2	X3X3
20	-20	70	-400	1400	-1400	-28000	400	400	4900
20	-20	80	-400	1600	-1600	-32000	400	400	6400
20	40	70	800	1400	2800	56000	400	1600	4900
20	40	80	800	1600	3200	64000	400	1600	6400
70	-20	70	-1400	4900	-1400	-98000	4900	400	4900
70	-20	80	-1400	5600	-1600	-112000	4900	400	6400
70	40	70	2800	4900	2800	196000	4900	1600	4900
70	40	80	2800	5600	3200	224000	4900	1600	6400
1.75	10.0	75.0	17.5	131.25	750.0	1312.5	3.062	100.0	5625.0
88.25	10.0	75.0	882.5	6618.75	750.0	66187.5	7788.062	100.0	5625.0
45.0	-41.9	75.0	-1885.5	3375.0	-3142.5	-141412.5	2025.0	1755.61	5625.0
45.0	61.9	75.0	2785.5	3375.0	4642.5	208912.5	2025.0	3831.61	5625.0
45.0	10.0	66.35	450.0	2985.75	663.5	29857.5	2025.0	100.0	4402.322
45.0	10.0	83.65	450.0	3764.25	836.5	37642.5	2025.0	100.0	6997.323
45.0	10.0	75.0	450.0	3375.0	750.0	33750.0	2025.0	100.0	5625.0

Матриця планування Y:

Y1	Y2	Y3
-76512.1	-76517.1	-76512.1
-81982.1	-81979.1	-81978.1
282203.9	282205.9	282207.9
327617.9	327624.9	327621.9
-324175.1	-324171.1	-324170.1
-371134.1	-371133.1	-371140.1
919554.9	919546.9	919550.9
1049461.9	1049461.9	1049461.9
48966.18125	48965.18125	48962.18125
404643.53125000006	404642.53125000006	404642.53125000006
-531492.842	-531497.842	-531499.842
953124.4180000001	953129.4180000001	953127.4180000001
182642.30874999997	182643.30874999997	182647.30874999997
235798.82875	235796.82875	235793.82875
208656.4	208659.4	208657.4

Середні значення відгуку за рядками:

[-76513.76666666668, -81979.76666666668, 282205.9, 327621.5666666667, -324172.1, -371135.7666666666, 919550.9, 1049461.9, 48964.51458333334, 404642.86458333343, -531496.842, 953127.0846666667, 182644.30874999997, 235796.49541666664, 208657.7333333333]

Отримане рівняння регресії:

125.97183203393588 + 3.7894996940576187 * X1 + 6.12439912945988 * X2 + 1.09750072768066 * X3 + 1.017444444469586 * X1X2 + 1.000000000149285 * X1X3 + 0.8074444444632423 * X2X3+ 4.199766666666341 * X1X2X3 + 9.700892973119263 * X11^2 + 0.8009895684728173 * X22^2 + 7.522259483225089 * X33^2 = ŷ

Експериментальні значення:

[-76514.94328778161, -81981.13461119532, 282206.3081422351, 327621.7834854955, -324172.56103463296, -371136.41902463546, 919552.0237287241, 1049462.8324053858, 48965.5050477861, 404642.200512469, -531494.8467068909, 953125.4157670464, 182644.25076231564, 235796.87979513893, 208657.73101854368]

```
Перевіримо за критерієм Кохрена:
Gp = 0.1509431292362988
З ймовірністю 0.95 дисперсії однорідні.

Перевіримо значущості коефіцієнтів за критерієм Стьюдента:

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


Перевіримо адекватності за критерієм Фішера:
Fp = 1.2142037552471731e+20
F_t = 2.6896275736914177
Математична модель адекватна
```