Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №6

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

Виконала:

студентка II курсу ФІОТ

групи ІВ-91

Яременко Влада

Перевірив:

Регіда П.Г.

Київ – 2021

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

Завдання на лабораторну роботу:

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 $\overline{x_1}$, $\overline{x_2}$, $\overline{x_3}$.

3. Значення функції відгуку знайти за допомогою підстановки в формулу:

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

де f(x1, x2, x3) 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.

Варіант завдання:

Варіант	X_1		X_2		X_3	
	min	max	min	max	min	max
130	10	50	20	60	20	25
$f(x_1, x_2, x_3)$	`6.2 + 6.0 * X1 + 6.5 * X2 + 3.2 * X3 + 9.9 * X1 * X1 + 0.6 * X2 * X2 + 3.6 * X3 * X3 + 2.3 * X1 * X2 + 0.4 * X1 * X3 + 2.9 * X2 * X3 + 7.4 * X1 * X2 * X3`					

Довірча ймовірність дорівнює 0.95, а рівень значимості q = 0.05.

Роздруківка тексту програми:

```
from math import fabs
from random import randrange
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from prettytable import PrettyTable

# Виконала: Яременко Влада
# Група: ІВ-91
```

```
# Номер у списку:30
# Варіант: 130


m = 3
n = 15

x1min = 10
x1max = 50
x2min = 20
x2max = 60
x3min = 20
x3max = 25

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 = 6.2 + 6.0 * X1 + 6.5 * X2 + 3.2 * X3 + 9.9 * X1 * X1 + 0.6 * X2 * X2 + 3.6 *
X3 * X3 + 2.3 * X1 * X2 + \
        0.4 * X1 * X3 + 2.9 * X2 * X3 + 7.4 * X1 * X2 * X3 + randrange(0, 10) - 5
    return y


xn = [[-1, -1, -1, +1, +1, +1, -1, +1, +1, +1],
      [-1, -1, +1, +1, -1, -1, +1, +1, +1, +1],
      [-1, +1, -1, -1, +1, -1, +1, +1, +1, +1],
      [-1, +1, +1, -1, -1, +1, -1, +1, +1, +1],
      [+1, -1, -1, -1, -1, +1, +1, +1, +1, +1],
      [+1, -1, +1, -1, +1, -1, -1, +1, +1, +1],
      [+1, +1, -1, +1, -1, -1, -1, +1, +1, +1],
      [+1, +1, +1, +1, +1, +1, +1, +1, +1, +1],
      [-1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],
      [+1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],
      [0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],
      [0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],
      [0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929],
      [0, 0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929],
      [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

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:")
for i in range(15):
    print("{:.3f}".format(Y_average[i]), end=" ")

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 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[0], mx[1], mx[2], mx[3], mx[4], mx[5], mx[6], mx[7], mx[8], mx[9]],
    [mx[0], a(1, 1), a(1, 2), a(1, 3), a(1, 4), a(1, 5), a(1, 6), a(1, 7), a(1, 8),
a(1, 9), a(1, 10)],
    [mx[1], a(2, 1), a(2, 2), a(2, 3), a(2, 4), a(2, 5), a(2, 6), a(2, 7), a(2, 8),
a(2, 9), a(2, 10)],
    [mx[2], a(3, 1), a(3, 2), a(3, 3), a(3, 4), a(3, 5), a(3, 6), a(3, 7), a(3, 8),
a(3, 9), a(3, 10)],
    [mx[3], a(4, 1), a(4, 2), a(4, 3), a(4, 4), a(4, 5), a(4, 6), a(4, 7), a(4, 8),
a(4, 9), a(4, 10)],
    [mx[4], a(5, 1), a(5, 2), a(5, 3), a(5, 4), a(5, 5), a(5, 6), a(5, 7), a(5, 8),
a(5, 9), a(5, 10)],
    [mx[5], a(6, 1), a(6, 2), a(6, 3), a(6, 4), a(6, 5), a(6, 6), a(6, 7), a(6, 8),
a(6, 9), a(6, 10)],
    [mx[6], a(7, 1), a(7, 2), a(7, 3), a(7, 4), a(7, 5), a(7, 6), a(7, 7), a(7, 8),
a(7, 9), a(7, 10)],
    [mx[7], a(8, 1), a(8, 2), a(8, 3), a(8, 4), a(8, 5), a(8, 6), a(8, 7), a(8, 8),
a(8, 9), a(8, 10)],
    [mx[8], a(9, 1), a(9, 2), a(9, 3), a(9, 4), a(9, 5), a(9, 6), a(9, 7), a(9, 8),
a(9, 9), a(9, 10)],
    [mx[9], a(10, 1), a(10, 2), a(10, 3), a(10, 4), a(10, 5), a(10, 6), a(10, 7),
a(10, 8), a(10, 9), a(10, 10)]]

det2 = [my, find_known(1), find_known(2), find_known(3), find_known(4),
find_known(5), find_known(6), find_known(7),
        find_known(8), find_known(9), find_known(10)]

beta = solve(det1, det2)
print("\nОтримане рівняння регресії:")
print("{:.3f} + {:.3f} * X1 + {:.3f} * X2 + {:.3f} * X3 + {:.3f} * X1X2 + {:.3f} *
X1X3 + {:.3f} * X2X3"
          "+ {:.3f} * X1X2X3 + {:.3f} * X11^2 + {:.3f} * X22^2 + {:.3f} * X33^2 = ŷ"
          .format(*beta))
y_i = [0] * 15
print("Експериментальні значення:")
for k in range(15):
    y_i[k] = 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]
for i in range(15):
    print("{:.3f}".format(y_i[i]), end=" ")
print("\n\n------------------------------ Перевірка за критерієм Кохрена -----------
--------------------")
Gp = max(dispersions) / sum(dispersions)
Gt = 0.3346
print("Gp =", Gp)
if Gp < Gt:
    print("Дисперсія однорідна")
else:
    print("Дисперсія неоднорідна")
```

```
print("\n------------------ Перевірка значущості коефіцієнтів за критерієм Стьюдента
------------------")
sb = sum(dispersions) / len(dispersions)
sbs = (sb / (15 * m)) ** 0.5

F3 = (m - 1) * n
coefs1 = []
coefs2 = []
d = 11
res = [0] * 11
for j in range(11):
    t_pract = 0
    for i in range(15):
        if j == 0:
            t_pract += Y_average[i] / 15
        else:
            t_pract += Y_average[i] * xn[i][j - 1]
        res[j] = beta[j]
    if fabs(t_pract / sbs) < t.ppf(q=0.975, df=F3):
        coefs2.append(beta[j])
        res[j] = 0
        d -= 1
    else:
        coefs1.append(beta[j])
print("Значущі коефіцієнти регресії:", [round(i, 3) for i in coefs1])
print("Незначущі коефіцієнти регресії:", [round(i, 3) for i in coefs2])
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("Значення з отриманими коефіцієнтами:")
for i in range(15):
    print("{:.3f}".format(y_st[i]), end=" ")

print("\n\n------------------------ Перевірка адекватності за критерієм Фішера -----
--------------------")
Sad = m * sum([(y_st[i] - Y_average[i]) ** 2 for i in range(15)]) / (n - d)
Fp = Sad / sb
F4 = n - d
print("Fp =", Fp)
if Fp < f.ppf(q=0.95, dfn=F4, dfd=F3):
    print("Рівняння регресії адекватне при рівні значимості 0.05")
else:
    print("Рівняння регресії неадекватне при рівні значимості 0.05")
```

Результати роботи програми:

Матриця планування з натуралізованими коефіцієнтами X:

X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1X1	X2X2	X3X3
10	20	20	200	200	400	4000	100	400	400
10	20	25	200	250	500	5000	100	400	625
10	60	20	600	200	1200	12000	100	3600	400
10	60	25	600	250	1500	15000	100	3600	625
50	20	20	1000	1000	400	20000	2500	400	400
50	20	25	1000	1250	500	25000	2500	400	625
50	60	20	3000	1000	1200	60000	2500	3600	400
50	60	25	3000	1250	1500	75000	2500	3600	625
-4.6	40.0	22.5	-184.0	-103.5	900.0	-4140.0	21.16	1600.0	506.25
64.6	40.0	22.5	2584.0	1453.5	900.0	58140.0	4173.16	1600.0	506.25
30.0	5.4	22.5	162.0	675.0	121.5	3645.0	900.0	29.16	506.25
30.0	74.6	22.5	2238.0	675.0	1678.5	50355.0	900.0	5565.16	506.25
30.0	40.0	18.175	1200.0	545.25	727.0	21810.0	900.0	1600.0	330.331
30.0	40.0	26.825	1200.0	804.75	1073.0	32190.0	900.0	1600.0	719.581
30.0	40.0	22.5	1200.0	675.0	900.0	27000.0	900.0	1600.0	506.25

Матриця планування Y:

Y1	Y2	Y3
34231.2	34233.2	34227.2
42765.2	42763.2	42768.2
98854.2	98846.2	98850.2
122763.2	122766.2	122767.2
178792.2	178793.2	178786.2
217006.2	217002.2	217004.2
483891.2	483892.2	483891.2
596683.2	596682.2	596686.2
-25188.015999999996	-25188.015999999996	-25193.015999999996
484189.18399999995	484190.18399999995	484192.18399999995
39014.246	39015.246	39010.246
397727.7459999999	397723.7459999999	397725.7459999999
178039.95025	178039.95025	178039.95025
257389.13025	257387.13025	257393.13025
217645.7	217647.7	217653.7

```
Середні значення відгуку за рядками:
34230.533 42765.533 98850.200 122765.533 178790.533 217004.200 483891.533 596683.867 -25189.683 484190.517 39013.246 397725.746 178039.950 257389.797 217649.033
Отримане рівняння регресії:
-13.540 + 6.023 * X1 + 6.199 * X2 + 5.243 * X3 + 2.305 * X1X2 + 0.397 * X1X3 + 2.904 * X2X3+ 7.400 * X1X2X3 + 9.900 * X11^2 + 0.602 * X22^2 + 3.555 * X33^2 = ŷ
Експериментальні значення:
34229.848 42766.080 98848.736 122765.302 178789.779 217004.678 483890.001 596683.567 -25189.104 484191.256 39013.005 397727.304 178042.034 257389.031 217649.024

------------------------------ Перевірка за критерієм Кохрена ------------------------------
Gp = 0.16149068322981372
Дисперсія однорідна

------------------ Перевірка значущості коефіцієнтів за критерієм Стьюдента ------------------
Значущі коефіцієнти регресії: [-13.54, 6.023, 6.199, 5.243, 2.305, 0.397, 2.904, 7.4, 9.9, 0.602, 3.555]
Незначущі коефіцієнти регресії: []
Значення з отриманими коефіцієнтами:
34229.848 42766.080 98848.736 122765.302 178789.779 217004.678 483890.001 596683.567 -25189.104 484191.256 39013.005 397727.304 178042.033 257389.029 217649.024

------------------------- Перевірка адекватності за критерієм Фішера -------------------------
Fp = 2.278529501370433
Рівняння регресії адекватне при рівні значимості 0.05
```

Висновок: У ході лабораторної роботи був проведений трьохфакторний експеримент, в результаті якого отримано адекватну модель – рівняння регресії, за допомогою рототабельного композиційного плану.