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 №5

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

Виконала:
студентка ІІ курсу ФІОТ
групи ІВ-91
Сайко Сабріна
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 x1, x2, x3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1; +l ; -l ; 0 для $x1$, $x2$, $x3$.

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.

Варіант завдання:

Варіант	X1		X2		X3	
124	-20	15	-15	35	-15	-10
$f(x_1, x_2, x_3)$	8,8+8,3*x1+4,9*x2+1,2*x3+0,5*x1*x1+0,4*x2*x2+9,7*x3*x3+2,1*x1*x2+0,8*x1*x3+5,4**x2*x3+1,1*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


def round_matrix(matrix, n_to_round=3):
    for i in range(len(matrix)):
        matrix[i] = list(matrix[i])
        for j in range(len(matrix[i])):
            matrix[i][j] = round(matrix[i][j], n_to_round)
```

```
    return matrix


m = 3
n = 15

x1min = -20
x1max = 15
x2min = -15
x2max = 35
x3min = -15
x3max = -10

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

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

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


def function(X1, X2, X3):
    y = 8.8 + 8.3 * X1 + 4.9 * X2 + 1.2 * X3 + 0.5 * X1 * X1 + 0.4 * X2 * X2
+ 9.7 * X3 * X3 + 2.1 * X1 * X2 + \
        0.8 * X1 * X3 + 5.4 * X2 * X3 + 1.1 * X1 * X2 * X3 + randrange(0, 10)
- 5
    return y


Y = round_matrix([[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)])

planning_matrix_y = PrettyTable()
planning_matrix_y.title = 'Матриця планування Y'
planning_matrix_y.field_names = ['Y1', 'Y2', 'Y3']
planning_matrix_y.add_rows(Y)
print(planning_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[0], beta[1], beta[2], beta[3], beta[4], beta[5], beta[6],
beta[7], beta[8], beta[9], beta[10]))
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\nПеревірка 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("\nПеревірка 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])
if len([round(i, 3) for i in coefs2]) != 0:
    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\nПеревірка адекватності за критерієм Фішера")
S = m * sum([(y_st[i] - Y_average[i]) ** 2 for i in range(15)]) / (n - d)
Fp = S / sb
F4 = n - d
print("Fp =", Fp)
```

```python
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
-20	-15	-15	300	300	225	-4500	400	225	225
-20	-15	-10	300	200	150	-3000	400	225	100
-20	35	-15	-700	300	-525	10500	400	1225	225
-20	35	-10	-700	200	-350	7000	400	1225	100
15	-15	-15	-225	-225	225	3375	225	225	225
15	-15	-10	-225	-150	150	2250	225	225	100
15	35	-15	525	-225	-525	-7875	225	1225	225
15	35	-10	525	-150	-350	-5250	225	1225	100
-32.775	10.0	-12.5	-327.75	409.688	-125.0	4096.875	1074.201	100.0	156.25
27.775	10.0	-12.5	277.75	-347.188	-125.0	-3471.875	771.451	100.0	156.25
-2.5	-33.25	-12.5	83.125	31.25	415.625	-1039.062	6.25	1105.562	156.25
-2.5	53.25	-12.5	-133.125	31.25	-665.625	1664.062	6.25	2835.562	156.25
-2.5	10.0	-16.825	-25.0	42.062	-168.25	420.625	6.25	100.0	283.081
-2.5	10.0	-8.175	-25.0	20.438	-81.75	204.375	6.25	100.0	66.831
-2.5	10.0	-12.5	-25.0	31.25	-125.0	312.5	6.25	100.0	156.25

Матриця планування Y

Y1	Y2	Y3
-639.2	-638.2	-638.2
-682.7	-687.7	-684.7
10352.8	10351.8	10351.8
6162.3	6158.3	6160.3
6703.8	6696.8	6700.8
3916.8	3913.8	3915.8
-7502.2	-7504.2	-7504.2
-4819.2	-4819.2	-4813.2
5332.53	5329.53	5337.53
-1973.855	-1974.855	-1978.855
3069.069	3075.069	3067.069
871.481	866.481	868.481
2337.155	2336.155	2344.155
462.835	460.835	469.835
1225.05	1226.05	1217.05

```
Середні значення відгуку за рядками:
-638.533 -685.033 10352.133 6160.300 6700.467 3915.467 -7503.533 -4817.200 5333.197 -1975.855 3070.402 868.814 2339.155 464.502 1222.717
Отримане рівняння регресії:
```

```
-3.996 + 8.761 * X1 + 4.898 * X2 + -1.034 * X3 + 2.089 * X1X2 + 0.837 * X1X3 + 5.400 * X2X3+ 1.099 * X1X2X3 + 0.498 * X11^2 + 0.400 * X22^2 + 9.608 * X33^2 = y
Експериментальні значення:
-639.353 -685.582 10352.415 6160.853 6699.018 3914.290 -7503.880 -4817.275 5333.069 -1974.530 3072.274 868.140 2340.068 464.786 1222.708

Перевірка за критерієм Кохрена
Gp = 0.16043956043956045
Дисперсія однорідна

Перевірка значущості коефіцієнтів за критерієм Стьюдента
Значущі коефіцієнти регресії: [-3.996, 8.761, 4.898, -1.034, 2.089, 0.837, 5.4, 1.099, 0.498, 0.4, 9.608]
Значення з отриманими коефіцієнтами:
-639.353 -685.582 10352.415 6160.853 6699.018 3914.290 -7503.880 -4817.275 5333.068 -1974.530 3072.274 868.140 2340.064 464.782 1222.708

Перевірка адекватності за критерієм Фішера
Fp = 1.2906683843557125
Рівняння регресії адекватне при рівні значимості 0.05
```

Висновок: У ході лабораторної роботи я змоделювала трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи рототабельний композиційний план.