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 оптимізації та планування»

Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами

ВИКОНАВ:
Студент II курсу ФІОТ
Групи ІВ-92
Залога А.С.

ПЕРЕВІРИВ:
Регіда П.Г.

Київ 2021 р.

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

Варіант завдання:

208	-30	0	10	60	10	35	8,0+5,3*x1+0,5*x2+5,6*x3+3,2*x1*x1+0,7*x2*x2+4,1*x3*x3+8,9*x1*x2+0,5*x1*x3+1,5*x2*x3+1,2*x1*x2*x3

Лістинг 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


def function(X1, X2, X3):
    y = 8.0 + 5.3 * X1 + 0.5 * X2 + 5.6 * X3 + 3.2 * X1 * X1 + 0.7 * X2 * X2
+ 4.1 * X3 * X3 + 8.9 * X1 * X2 + \
        0.5 * X1 * X3 + 1.5 * X2 * X3 + 1.2 * X1 * X2 * X3 + randrange(0, 10)
- 5
    return y


def main(n, m):
    x1min = -30
    x1max = 0
    x2min = 10
    x2max = 60
    x3min = 10
    x3max = 35

    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)

    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])
    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")
    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")
```

```
main(15, 3)
```

Результати роботи програми:

```
E:\Study\Lab6_MND\venv\Scripts\python.exe E:\Study\Lab6_MND\lab6.py
```

Матриця планування з натуралізованими коефіцієнтами X

X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1X1	X2X2	X3X3
-30	10	10	-300	-300	100	-3000	900	100	100
-30	10	35	-300	-1050	350	-10500	900	100	1225
-30	60	10	-1800	-300	600	-18000	900	3600	100
-30	60	35	-1800	-1050	2100	-63000	900	3600	1225
0	10	10	0	0	100	0	0	100	100
0	10	35	0	0	350	0	0	100	1225
0	60	10	0	0	600	0	0	3600	100
0	60	35	0	0	2100	0	0	3600	1225
-40.95	35.0	22.5	-1433.25	-921.375	787.5	-32248.125	1676.903	1225.0	506.25
10.95	35.0	22.5	383.25	246.375	787.5	8623.125	119.902	1225.0	506.25
-15.0	-8.25	22.5	123.75	-337.5	-185.625	2784.375	225.0	68.062	506.25
-15.0	78.25	22.5	-1173.75	-337.5	1760.625	-26409.375	225.0	6123.062	506.25
-15.0	35.0	0.875	-525.0	-13.125	30.625	-459.375	225.0	1225.0	0.766
-15.0	35.0	44.125	-525.0	-661.875	1544.375	-23165.625	225.0	1225.0	1947.016
-15.0	35.0	22.5	-525.0	-337.5	787.5	-11812.5	225.0	1225.0	506.25

Матриця планування Y

Y1	Y2	Y3
-3004.0	-2996.0	-3001.0
-7249.5	-7244.5	-7252.5
-31123.0	-31126.0	-31130.0
-78500.5	-78496.5	-78500.5
700.0	695.0	695.0
5824.5	5830.5	5830.5
3926.0	3920.0	3924.0
10930.5	10929.5	10922.5
-42497.435	-42497.435	-42498.435
18591.461	18584.461	18585.461
6885.081	6890.081	6890.081
-32494.044	-32489.044	-32492.044
-3657.836	-3648.836	-3655.836
-20732.261	-20731.261	-20728.261
-14106.875	-14113.875	-14113.875

```
Середні значення відгуку за рядками:
-3000.333 -7248.833 -31126.333 -78499.167 696.667 5828.500 3923.333 10927.500 -42497.768 18587.128 6888.414 -32491.711 -3654.169 -20730.594 -14111.542
Отримане рівняння регресії:
4.052 + 5.142 * X1 + 0.535 * X2 + 5.636 * X3 + 8.903 * X1X2 + 0.508 * X1X3 + 1.498 * X2X3+ 1.200 * X1X2X3 + 3.202 * X11^2 + 0.700 * X22^2 + 4.104 * X33^2 = ŷ
Експериментальні значення:
-2999.562 -7247.820 -31125.738 -78498.329 695.950 5828.026 3922.440 10926.850 -42499.569 18588.767 6888.130 -32491.588 -3653.969 -20730.956 -14111.541

Перевірка за критерієм Кохрена
Gp = 0.1303501945525292
Дисперсія однорідна

Перевірка значущості коефіцієнтів за критерієм Стьюдента
Значущі коефіцієнти регресії: [4.052, 5.142, 0.535, 5.636, 8.903, 0.508, 1.498, 1.2, 3.202, 0.7, 4.104]
Незначущі коефіцієнти регресії: []
Значення з отриманими коефіцієнтами:
-2999.562 -7247.820 -31125.738 -78498.329 695.950 5828.026 3922.440 10926.850 -42499.571 18588.769 6888.130 -32491.587 -3653.971 -20730.957 -14111.541

Перевірка адекватності за критерієм Фішера
Fp = 1.0681327205868631
Рівняння регресії адекватне при рівні значимості 0.05

Process finished with exit code 0
```

Висновок:
В даній лабораторній роботі я провів трьохфакторний експеримент і отримав адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.