

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 № 2

**з дисципліни «Методи оптимізації та планування експерименту»
на тему «Проведення трьохфакторного експерименту
при використанні рівняння регресії з урахуванням ефекту взаємодії.»**

ВИКОНАВ:
студент II курсу ФІОТ
групи ІО-93
Поліщук М. С.
Варіант: 322

ПЕРЕВІРИВ:
Регіда П. Г.

Тема: Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії.

Мета: Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання:

Завдання на лабораторну роботу

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

$$y_{i\max} = 200 + x_{\text{ср}\max}$$

$$y_{i\min} = 200 + x_{\text{ср}\min}$$

$$\text{де } x_{\text{ср}\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}, \quad x_{\text{ср}\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$$

3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

Хід роботи:

322	-30	20	-20	40	-30	-15
-----	-----	----	-----	----	-----	-----

Лістинг програми:

```
import random, math
import numpy as np
from scipy.stats import f, t
from functools import partial

m = 3
N = 8
x1min, x2min, x3min = -30, -20, -30
x1max, x2max, x3max = 20, 40, -15

X_max = [x1max, x2max, x3max]
X_min = [x1min, x2min, x3min]

x_av_min = (x1min + x2min + x3min) / 3
x_av_max = (x1max + x2max + x3max) / 3
Y_max = int(round(200 + x_av_max, 0))
Y_min = int(round(200 + x_av_min, 0))
X0 = 1

X_matr = [
    [-1, -1, -1],
    [-1, -1, 1],
    [-1, 1, -1],
    [-1, 1, 1],
    [1, -1, -1],
```

```

        [1, -1, 1],
        [1, 1, -1],
        [1, 1, 1]
    ]
    x_for_beta = [
        [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]
    ]
    x_12_13_23 = [
        [1, 1, 1],
        [1, -1, -1],
        [-1, 1, -1],
        [-1, -1, 1],
        [-1, -1, 1],
        [-1, 1, -1],
        [1, -1, -1],
        [1, 1, 1],
    ]
    ]
    x_123 = [
        -1,
        1,
        1,
        -1,
        1,
        -1,
        -1,
        1
    ]
    ]
    X_matr_natur = [
        [10, -70, 60],
        [10, -70, 70],
        [10, -10, 60],
        [10, -10, 70],
        [60, -70, 60],
        [60, -70, 70],
        [60, -10, 60],
        [60, -10, 70],
    ]
    ]
    x_12_13_23_natur = [[X_matr_natur[j][0] * X_matr_natur[j][1], X_matr_natur[j][0]
    * X_matr_natur[j][2],
        X_matr_natur[j][1] * X_matr_natur[j][2]] for j in range(N)]
    x_123_natur = [X_matr_natur[j][0] * X_matr_natur[j][1] * X_matr_natur[j][2] for
    j in range(N)]

    flag = False
    while not flag:
        Y_matr = [[random.randint((Y_min), (Y_max)) for i in range(m)] for j in
        range(N)]

        Y_average = [sum(j) / m for j in Y_matr]

        results_nat = [
            sum(Y_average),
            sum([Y_average[j] * X_matr_natur[j][0] for j in range(N)]),
            sum([Y_average[j] * X_matr_natur[j][1] for j in range(N)]),
            sum([Y_average[j] * X_matr_natur[j][2] for j in range(N)]),
            sum([Y_average[j] * x_12_13_23_natur[j][0] for j in range(N)]),
            sum([Y_average[j] * x_12_13_23_natur[j][1] for j in range(N)]),
            sum([Y_average[j] * x_12_13_23_natur[j][2] for j in range(N)]),

```

```

        sum([Y_average[j] * x_123_natur[j] for j in range(N)]),
    ]

    mj0 = [N,
        sum([X_matr_natur[j][0] for j in range(N)]),
        sum([X_matr_natur[j][1] for j in range(N)]),
        sum([X_matr_natur[j][2] for j in range(N)]),
        sum([x_12_13_23_natur[j][0] for j in range(N)]),
        sum([x_12_13_23_natur[j][1] for j in range(N)]),
        sum([x_12_13_23_natur[j][2] for j in range(N)]),
        sum([x_123_natur[j] for j in range(N)]),
    ]

    mj1 = [sum([X_matr_natur[j][0] for j in range(N)]),
        sum([X_matr_natur[j][0] ** 2 for j in range(N)]),
        sum([x_12_13_23_natur[j][0] for j in range(N)]),
        sum([x_12_13_23_natur[j][1] for j in range(N)]),
        sum([(X_matr_natur[j][0] ** 2) * X_matr_natur[j][1] for j in
range(N)]),
        sum([(X_matr_natur[j][0] ** 2) * X_matr_natur[j][2] for j in
range(N)]),
        sum([x_123_natur[j] for j in range(N)]),
        sum([(X_matr_natur[j][0] ** 2) * x_12_13_23_natur[j][2] for j in
range(N)]),
    ]

    mj2 = [sum([X_matr_natur[j][1] for j in range(N)]),
        sum([x_12_13_23_natur[j][0] for j in range(N)]),
        sum([X_matr_natur[j][1] ** 2 for j in range(N)]),
        sum([x_12_13_23_natur[j][2] for j in range(N)]),
        sum([(X_matr_natur[j][1] ** 2) * X_matr_natur[j][0] for j in
range(N)]),
        sum([x_123_natur[j] for j in range(N)]),
        sum([(X_matr_natur[j][1] ** 2) * X_matr_natur[j][2] for j in
range(N)]),
        sum([(X_matr_natur[j][1] ** 2) * x_12_13_23_natur[j][1] for j in
range(N)]),
    ]

    mj3 = [sum([X_matr_natur[j][2] for j in range(N)]),
        sum([x_12_13_23_natur[j][1] for j in range(N)]),
        sum([x_12_13_23_natur[j][2] for j in range(N)]),
        sum([X_matr_natur[j][2] ** 2 for j in range(N)]),
        sum([x_123_natur[j] for j in range(N)]),
        sum([(X_matr_natur[j][2] ** 2) * X_matr_natur[j][0] for j in
range(N)]),
        sum([(X_matr_natur[j][2] ** 2) * X_matr_natur[j][1] for j in
range(N)]),
        sum([(X_matr_natur[j][2] ** 2) * x_12_13_23_natur[j][0] for j in
range(N)]),
    ]

    mj4 = [sum([x_12_13_23_natur[j][0] for j in range(N)]),
        sum([(X_matr_natur[j][0] ** 2) * X_matr_natur[j][1] for j in
range(N)]),
        sum([(X_matr_natur[j][1] ** 2) * X_matr_natur[j][0] for j in
range(N)]),
        sum([x_123_natur[j] for j in range(N)]),
        sum([x_12_13_23_natur[j][0] ** 2 for j in range(N)]),
        sum([(X_matr_natur[j][0] ** 2) * x_12_13_23_natur[j][2] for j in
range(N)]),
        sum([(X_matr_natur[j][1] ** 2) * x_12_13_23_natur[j][1] for j in
range(N)]),
        sum([(x_12_13_23_natur[j][0] ** 2) * X_matr_natur[j][2] for j in
range(N)]),
    ]

    mj5 = [sum([x_12_13_23_natur[j][1] for j in range(N)]),
        sum([(X_matr_natur[j][0] ** 2) * X_matr_natur[j][2] for j in
range(N)]),
        sum([x_123_natur[j] for j in range(N)]),

```

```

        sum([(X_matr_natur[j][2] ** 2) * X_matr_natur[j][0] for j in
range(N)]),
        sum([(X_matr_natur[j][0] ** 2) * x_12_13_23_natur[j][2] for j in
range(N)]),
        sum([x_12_13_23_natur[j][1] ** 2 for j in range(N)]),
        sum([(X_matr_natur[j][2] ** 2) * x_12_13_23_natur[j][0] for j in
range(N)]),
        sum([(x_12_13_23_natur[j][1] ** 2) * X_matr_natur[j][1] for j in
range(N)]),
    ]
    mj6 = [sum([x_12_13_23_natur[j][2] for j in range(N)]),
            sum([x_123_natur[j] for j in range(N)]),
            sum([(X_matr_natur[j][1] ** 2) * X_matr_natur[j][2] for j in
range(N)]),
            sum([(X_matr_natur[j][2] ** 2) * X_matr_natur[j][1] for j in
range(N)]),
            sum([(X_matr_natur[j][1] ** 2) * x_12_13_23_natur[j][1] for j in
range(N)]),
            sum([(X_matr_natur[j][2] ** 2) * x_12_13_23_natur[j][0] for j in
range(N)]),
            sum([(X_matr_natur[j][2] ** 2) * X_matr_natur[j][1] for j in
range(N)]),
            sum([(x_12_13_23_natur[j][2] ** 2) * X_matr_natur[j][0] for j in
range(N)]),
    ]
    mj7 = [sum([x_123_natur[j] for j in range(N)]),
            sum([(X_matr_natur[j][0] ** 2) * x_12_13_23_natur[j][2] for j in
range(N)]),
            sum([(X_matr_natur[j][1] ** 2) * x_12_13_23_natur[j][1] for j in
range(N)]),
            sum([(X_matr_natur[j][2] ** 2) * x_12_13_23_natur[j][0] for j in
range(N)]),
            sum([(x_12_13_23_natur[j][0] ** 2) * X_matr_natur[j][2] for j in
range(N)]),
            sum([(x_12_13_23_natur[j][1] ** 2) * X_matr_natur[j][1] for j in
range(N)]),
            sum([(x_12_13_23_natur[j][2] ** 2) * X_matr_natur[j][0] for j in
range(N)]),
            sum([x_123_natur[j] ** 2 for j in range(N)])
    ]

    B_nat1 = np.linalg.solve([mj0, mj1, mj2, mj3, mj4, mj5, mj6, mj7],
results_nat) # list of B's
    B_nat = list(B_nat1)

    B_norm = [
        sum(Y_average) / N,
        sum([Y_average[j] * X_matr[j][0] for j in range(N)]) / N,
        sum([Y_average[j] * X_matr[j][1] for j in range(N)]) / N,
        sum([Y_average[j] * X_matr[j][2] for j in range(N)]) / N,
        sum([Y_average[j] * x_12_13_23[j][0] for j in range(N)]) / N,
        sum([Y_average[j] * x_12_13_23[j][1] for j in range(N)]) / N,
        sum([Y_average[j] * x_12_13_23[j][2] for j in range(N)]) / N,
        sum([Y_average[j] * x_123[j] for j in range(N)]) / N,
    ]

    print("Матриця планування експерименту:")
    print("N      " + "x1  " + "x2  " + "x3      " + "Y1" + " " * 8 + "Y2" + " " * 8
+ "Y3")
    for i in range(N):
        print("{0:=d} {1:=4d} {2:=3d} {3:=3d} {4:=9.5f} {5:=9.5f}
{6:=9.5f}".format(i + 1,
X_matr[i][0],
X_matr[i][1],

```

```

X_matr[i][2],
Y_matr[i][0],
Y_matr[i][1],
Y_matr[i][2]))
    print('###' * 40, '\n')

    def criterion_of_Student(value, criterion, check):
        if check < criterion:
            return 0
        else:
            return value

    y1_nat = B_nat[0] + B_nat[1] * X_matr_natur[0][0] + B_nat[2] *
X_matr_natur[0][1] + B_nat[3] * X_matr_natur[0][2] + \
        B_nat[4] * x_12_13_23_natur[0][0] + B_nat[5] *
x_12_13_23_natur[0][1] + B_nat[6] * x_12_13_23_natur[0][2] + \
        B_nat[7] * x_123_natur[0]
    y1_norm = B_norm[0] + B_norm[1] * X_matr[0][0] + B_norm[2] * X_matr[0][1] +
B_norm[3] * X_matr[0][2] + B_norm[4] * \
        x_12_13_23[0][0] + B_norm[5] * x_12_13_23[0][1] + B_norm[6] *
x_12_13_23[0][2] + B_norm[7] * x_123[0]

    dx = [(X_max[i] - X_min[i]) / 2) for i in range(3)]
    A = [sum(Y_average) / len(Y_average), B_nat[0] * dx[0], B_nat[1] * dx[1],
B_nat[2] * dx[2]]

    S_kv = [(sum([(Y_matr[i][j] - Y_average[i]) ** 2) for j in range(m)]) / m)
for i in range(N)]

    Gp = max(S_kv) / sum(S_kv)

    f1 = m - 1
    f2 = N
    p = .95
    q = 1 - p
    # for N=8
    Gt_dict = {2: 5157, 3: 4377, 4: 3910, 5: 3595, 6: 3362, 7: 3185, 8: 3043, 9:
2926, 10: 2829, 16: 2462}

    def kohren(f1=f1, f2=f2, q=0.05):
        q1 = q / f1
        fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
        return fisher_value / (fisher_value + f1 - 1)

    Gt = kohren()

    if Gp < Gt:
        print('Дисперсії однорідні')

        flag = False
    else:
        print('Дисперсії неоднорідні')
        m += 1
    S_average = sum(S_kv) / N

    S2_beta_s = S_average / (N * m)

    S_beta_s = S2_beta_s ** .5

```

```

    beta = [(sum([x_for_beta[j][i] * Y_average[j] for j in range(N)]) / N) for i
in range(4)]
    ts = [(math.fabs(beta[i]) / S_beta_s) for i in range(4)]
    tabl_Stud = [
        12.71,
        4.303,
        3.182,
        2.776,
        2.571,
        2.447,
        2.365,
        2.306,
        2.262,
        2.228,
        2.201,
        2.179
    ]
    f3 = f1 * f2

    student = partial(t.ppf, q=1 - 0.025)
    criterion_of_St = student(df=f3)

    result_2 = [criterion_of_Student(B_nat[0], criterion_of_St, ts[0]) +
        criterion_of_Student(B_nat[1], criterion_of_St, ts[1]) *
X_matr_natur[i][0] +
        criterion_of_Student(B_nat[2], criterion_of_St, ts[2]) *
X_matr_natur[i][1] +
        criterion_of_Student(B_nat[3], criterion_of_St, ts[3]) *
X_matr_natur[i][2] for i in range(N)]

    znach_koef = []
    for i in ts:
        if i > criterion_of_St:
            znach_koef.append(i)
        else:
            pass

    d = len(znach_koef)
    f4 = N - d
    f3 = (m - 1) * N

    deviation_of_adequacy = (m / (N - d)) * sum([(result_2[i] - Y_average[i]) **
2 for i in range(N)])

    Fp = deviation_of_adequacy / S2_beta_s

    fisher = partial(f.ppf, q=1 - 0.05)
    Ft = fisher(dfn=f4, dfd=f3)

    print("Значення після критерія Стюдента:")
    print("Y1 = {0:.3f};   Y2 = {1:.3f};   Y3 = {2:.3f};   Y4 =
{3:.3f}.".format(result_2[0],
result_2[1],
result_2[2],
result_2[3]))
    print("Y1a = {0:.3f};   Y2a = {1:.3f};   Y3a = {2:.3f};   Y4a =
{3:.3f}.".format(Y_average[0],
Y_average[1],
Y_average[2],
Y_average[3]))

```

```

if Fp > Ft:
    print('Fp = {} > Ft = {}'.format(round(Fp, 3), Ft))
    print('Рівняння регресії неадекватно оригіналу при рівні значимості {}'.format(round(q, 2)))
else:
    print('Fp = {} < Ft = {}'.format(round(Fp, 3), Ft))
    print('Рівняння регресії адекватно оригіналу при рівні значимості {}'.format(round(q, 2)))
    flag = True

```

Результат виконання роботи:

```

Дисперсії однорідні
Значення після критерія Стюдента:
Y1 = 212.902;   Y2 = 212.902;   Y3 = 212.902;   Y4 = 212.902.
Y1a = 202.333;   Y2a = 189.667;   Y3a = 181.333;   Y4a = 190.000.
Fp = 256.76 > Ft = 2.6571966002210865
Рівняння регресії неадекватно оригіналу при рівні значимості 0.05
Матриця планування експерименту:
N   x1  x2  x3   Y1       Y2       Y3
1 -  1 -  1 -  1 173.00000 197.00000 185.00000
2 -  1 -  1   1 174.00000 180.00000 200.00000
3 -  1   1 -  1 204.00000 189.00000 174.00000
4 -  1   1   1 209.00000 188.00000 173.00000
5   1 -  1 -  1 177.00000 214.00000 175.00000
6   1 -  1   1 204.00000 183.00000 181.00000
7   1   1 -  1 190.00000 176.00000 201.00000
8   1   1   1 186.00000 204.00000 178.00000
#####

Дисперсії однорідні
Значення після критерія Стюдента:
Y1 = 188.550;   Y2 = 188.550;   Y3 = 188.550;   Y4 = 188.550.
Y1a = 185.000;   Y2a = 184.667;   Y3a = 189.000;   Y4a = 190.000.
Fp = 2.085 < Ft = 2.6571966002210865
Рівняння регресії адекватно оригіналу при рівні значимості 0.05

Process finished with exit code 0

```

Висновок: у даній лабораторній роботі я провів повний трьохфакторний експеримент. Знайшов рівняння регресії адекватне об'єкту.