Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №4
з дисципліни «Методи оптимізації та планування експерименту»

# «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії»

Виконав:
студент групи ІО-92
Калашніков Ілля
Залікова книжка № ІО-92ХХ
Варіант: 210
Перевірив:
Регіда П.Г.

Київ 2020

**Мета:** провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

**Завдання:**

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.
3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, 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.

**Варіант:**

| | | | | | | |
|---|---|---|---|---|---|---|
| 210 | -20 | 15 | -30 | 45 | -30 | -15 |

## Код програми:

```
import math
import numpy as np
from prettytable import PrettyTable
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial
from random import randint


while True:
    def cohren_teor(f1, 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)

    fisher_teor = partial(f.ppf, q=1 - 0.05)
    student_teor = partial(t.ppf, q=1 - 0.025)

    X1max = 15
    X1min = -20
    X2max = 45
    X2min = -30
    X3max = -15
    X3min = -30


    Xmax_average = (X1max + X2max + X3max) / 3
    Xmin_average = (X1min + X2min + X3min) / 3

    y_max = round(200 + Xmax_average)
```

```
    y_min = round(200 + Xmin_average)

    # матриця ПФЕ
    x0_factor = [1, 1, 1, 1, 1, 1, 1, 1]
    x1_factor = [-1, -1, 1, 1, -1, -1, 1, 1]
    x2_factor = [-1, 1, -1, 1, -1, 1, -1, 1]
    x3_factor = [-1, 1, 1, -1, 1, -1, -1, 1]
    x1x2_factor = [a * b for a, b in zip(x1_factor, x2_factor)]
    x1x3_factor = [a * b for a, b in zip(x1_factor, x3_factor)]
    x2x3_factor = [a * b for a, b in zip(x2_factor, x3_factor)]
    x1x2x3_factor = [a * b * c for a, b, c in zip(x1_factor, x2_factor, x3_factor)]

    m = 3

    y1, y2, y3 = [], [], []
    for i in range(0, 8):
        y1.append(randint(y_min, y_max))
        y2.append(randint(y_min, y_max))
        y3.append(randint(y_min, y_max))

    Y_row1 = [y1[0], y2[0], y3[0]]
    Y_row2 = [y1[1], y2[1], y3[1]]
    Y_row3 = [y1[2], y2[2], y3[2]]
    Y_row4 = [y1[3], y2[3], y3[3]]
    Y_row5 = [y1[4], y2[4], y3[4]]
    Y_row6 = [y1[5], y2[5], y3[5]]
    Y_row7 = [y1[6], y2[6], y3[6]]
    Y_row8 = [y1[7], y2[7], y3[7]]
    Y_row_arr = [Y_row1, Y_row2, Y_row3, Y_row4, Y_row5, Y_row6, Y_row7, Y_row8]
   # for i in range(len(Y_row_arr)):
   #     Y_row_av_arr.append(np.average(Y_row_arr[i]))
   # for l in range(len(Y_row_av_arr)):
   #     Y_row_av_arr[l] = round(Y_row_av_arr[l], 3)
    Y_row_av_arr = list(map(lambda x: np.average(x), Y_row_arr))
    Y_row_av_arr = list(map(lambda x: round(x,3), Y_row_av_arr))



    x0 = [1, 1, 1, 1, 1, 1, 1, 1]
    x1 = [15, 15, 45, 45, 15, 15, 45, 45]
    x2 = [-25, 10, -25, 10, -25, 10, -25, 10]
    x3 = [-45, 50, 50, -45, 50, -45, -45, 50]
    x1x2 = [a * b for a, b in zip(x1, x2)]
    x1x3 = [a * b for a, b in zip(x1, x3)]
    x2x3 = [a * b for a, b in zip(x2, x3)]
    x1x2x3 = [a * b * c for a, b, c in zip(x1, x2, x3)]

    list_for_solve_b = [x0_factor, x1_factor, x2_factor, x3_factor, x1x2_factor,
x1x3_factor, x2x3_factor,
                        x1x2x3_factor]
    list_for_solve_a = list(zip(x0, x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3))

    N = 8
    list_bi = []  # b with "i" index
    for k in range(N):
        S = 0
        for i in range(N):
            S += (list_for_solve_b[k][i] * Y_row_av_arr[i]) / N
        list_bi.append(round(S, 5))

    Dispersion = [0,0,0,0,0,0,0,0]
    for k in range(len(Dispersion)):
        for i in range(m):
            Dispersion[k] += pow(((Y_row_arr[k][i] - np.average(Y_row_arr[k]))),2) / m
    Dispersion_sum = sum(Dispersion)
    Dispersion_list = list(map(lambda x: round(x, 3), Dispersion))
```

```
    pt1 = PrettyTable()  # Table
    column_names1 = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3", "Y1",
"Y2", "Y3", "Y", "S^2"]
    pt1.add_column(column_names1[0], x0_factor)
    pt1.add_column(column_names1[1], x1_factor)
    pt1.add_column(column_names1[2], x2_factor)
    pt1.add_column(column_names1[3], x3_factor)
    pt1.add_column(column_names1[4], x1x2_factor)
    pt1.add_column(column_names1[5], x1x3_factor)
    pt1.add_column(column_names1[6], x2x3_factor)
    pt1.add_column(column_names1[7], x1x2x3_factor)
    pt1.add_column(column_names1[8], y1)
    pt1.add_column(column_names1[9], y2)
    pt1.add_column(column_names1[10], y3)
    pt1.add_column(column_names1[11], Y_row_av_arr)
    pt1.add_column(column_names1[12], Dispersion_list)
    print(pt1, "\n")

    print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1x2 + {}*x1x3 + {}*x2x3 + {}*x1x2x3
\n".format(list_bi[0], list_bi[1],list_bi[2], list_bi[3],list_bi[4],
list_bi[5],list_bi[6], list_bi[7]))

    pt2 = PrettyTable()  # Table
    pt2.add_column(column_names1[0], x0)
    pt2.add_column(column_names1[1], x1)
    pt2.add_column(column_names1[2], x2)
    pt2.add_column(column_names1[3], x3)
    pt2.add_column(column_names1[4], x1x2)
    pt2.add_column(column_names1[5], x1x3)
    pt2.add_column(column_names1[6], x2x3)
    pt2.add_column(column_names1[7], x1x2x3)
    pt2.add_column(column_names1[8], y1)
    pt2.add_column(column_names1[9], y2)
    pt2.add_column(column_names1[10], y3)
    pt2.add_column(column_names1[11], Y_row_av_arr)
    pt2.add_column(column_names1[12], Dispersion_list)
    print(pt2, '\n')

    list_ai = [round(i, 5) for i in solve(list_for_solve_a, Y_row_av_arr)]
    print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1x2 + {}*x1x3 + {}*x2x3 +
{}*x1x2x3".format(list_ai[0], list_ai[1],list_ai[2], list_ai[3],list_ai[4],
list_ai[5],list_ai[6], list_ai[7]))

    Gp = max(Dispersion) / Dispersion_sum  # Exp
    F1 = m - 1
    N = len(y1)
    F2 = N
    Gt = cohren_teor(F1, F2)  # Teor

    def cohren(g_prac, g_teor):
        return g_prac < g_teor

    print("\nGp = ", Gp, " Gt = ", Gt)
    if cohren(Gp, Gt):
        print("Дисперсія однорідна!\n")

        Dispersion_B = Dispersion_sum / N
        Dispersion_beta = Dispersion_B / (m * N)
        S_beta = math.sqrt(abs(Dispersion_beta))

        beta0 = 0
        beta1 = 0
        beta2 = 0
        beta3 = 0
        beta4 = 0
        beta5 = 0
        beta6 = 0
```

```
        beta7 = 0
        for i in range(len(x0_factor)):
            beta0 += (Y_row_av_arr[i] * x0_factor[i]) / N
            beta1 += (Y_row_av_arr[i] * x1_factor[i]) / N
            beta2 += (Y_row_av_arr[i] * x2_factor[i]) / N
            beta3 += (Y_row_av_arr[i] * x3_factor[i]) / N
            beta4 += (Y_row_av_arr[i] * x1x2_factor[i]) / N
            beta5 += (Y_row_av_arr[i] * x1x3_factor[i]) / N
            beta6 += (Y_row_av_arr[i] * x2x3_factor[i]) / N
            beta7 += (Y_row_av_arr[i] * x1x2x3_factor[i]) / N
        beta_list = [beta0, beta1, beta2, beta3, beta4, beta5, beta6, beta7]

        t0 = abs(beta0) / S_beta
        t1 = abs(beta1) / S_beta
        t2 = abs(beta2) / S_beta
        t3 = abs(beta3) / S_beta
        t4 = abs(beta4) / S_beta
        t5 = abs(beta5) / S_beta
        t6 = abs(beta6) / S_beta
        t7 = abs(beta7) / S_beta
        t_list = [t0, t1, t2, t3, t4, t5, t6, t7]

        F3 = F1 * F2
        d = 0
        T = student_teor(df=F3)

        def student(t_teor, t_pr):
            return t_pr < t_teor

        print("t табличне = ", T)
        for i in range(len(t_list)):
            if student(t_list[i], T):
                beta_list[i] = 0
                print("Гіпотеза підтверджена, beta{} = 0".format(i))
            else:
                print("Гіпотеза не підтверджена.\nbeta{} = {}".format(i, beta_list[i]))
                d += 1

        y_1 = beta_list[0] + beta_list[1] * x1[0] + beta_list[2] * x2[0] + beta_list[3]
* x3[0] + beta_list[4] * x1x2[0] \
              + beta_list[5] * x1x3[0] + beta_list[6] * x2x3[0] + beta_list[7] *
x1x2x3[0]
        y_2 = beta_list[0] + beta_list[1] * x1[1] + beta_list[2] * x2[1] + beta_list[3]
* x3[1] + beta_list[4] * x1x2[1] \
              + beta_list[5] * x1x3[1] + beta_list[6] * x2x3[1] + beta_list[7] *
x1x2x3[1]
        y_3 = beta_list[0] + beta_list[1] * x1[2] + beta_list[2] * x2[2] + beta_list[3]
* x3[2] + beta_list[4] * x1x2[2] \
              + beta_list[5] * x1x3[2] + beta_list[6] * x2x3[2] + beta_list[7] *
x1x2x3[2]
        y_4 = beta_list[0] + beta_list[1] * x1[3] + beta_list[2] * x2[3] + beta_list[3]
* x3[3] + beta_list[4] * x1x2[3] \
              + beta_list[5] * x1x3[3] + beta_list[6] * x2x3[3] + beta_list[7] *
x1x2x3[3]
        y_5 = beta_list[0] + beta_list[1] * x1[4] + beta_list[2] * x2[4] + beta_list[3]
* x3[4] + beta_list[4] * x1x2[4] \
              + beta_list[5] * x1x3[4] + beta_list[6] * x2x3[4] + beta_list[7] *
x1x2x3[4]
        y_6 = beta_list[0] + beta_list[1] * x1[5] + beta_list[2] * x2[5] + beta_list[3]
* x3[5] + beta_list[4] * x1x2[5] \
              + beta_list[5] * x1x3[5] + beta_list[6] * x2x3[5] + beta_list[7] *
x1x2x3[5]
        y_7 = beta_list[0] + beta_list[1] * x1[6] + beta_list[2] * x2[6] + beta_list[3]
* x3[6] + beta_list[4] * x1x2[6] \
              + beta_list[5] * x1x3[6] + beta_list[6] * x2x3[6] + beta_list[7] *
x1x2x3[6]
        y_8 = beta_list[0] + beta_list[1] * x1[7] + beta_list[2] * x2[7] + beta_list[3]
```

```
* x3[7] + beta_list[4] * x1x2[7] \
              + beta_list[5] * x1x3[7] + beta_list[6] * x2x3[7] + beta_list[7] *
x1x2x3[7]
        Y_counted_for_Student = [y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8]

        F4 = N - d
        Dispersion_ad = 0
        for i in range(len(Y_counted_for_Student)):
            Dispersion_ad += ((Y_counted_for_Student[i] - Y_row_av_arr[i]) ** 2) * m /
(N - d)
        Fp = Dispersion_ad / Dispersion_beta
        Ft = fisher_teor(dfn=F4, dfd=F3)

        def fisher(f_teor, f_prac):
            return f_teor > f_prac

        if fisher(Ft, Fp):
            print("Рівняння регресії адекватне")
            break
        else:
            print("Рівняння регресії неадекватне")
            break

    else:
        print("Дисперсія неоднорідна")
        m += 1


    print("Дисперсія неоднорідна!")
```

# Результати виконання:

| X0 | X1 | X2 | X3 | X1X2 | X1X3 | X2X3 | X1X2X3 | Y1 | Y2 | Y3 | Y | S^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -1 | -1 | -1 | 1 | 1 | 1 | -1 | 210 | 178 | 198 | 195.333 | 174.222 |
| 1 | -1 | 1 | 1 | -1 | -1 | 1 | -1 | 173 | 199 | 182 | 184.667 | 116.222 |
| 1 | 1 | -1 | 1 | -1 | 1 | -1 | -1 | 193 | 209 | 215 | 205.667 | 86.222 |
| 1 | 1 | 1 | -1 | 1 | -1 | -1 | -1 | 204 | 196 | 174 | 191.333 | 160.889 |
| 1 | -1 | -1 | 1 | 1 | -1 | -1 | 1 | 204 | 180 | 192 | 192.0 | 96.0 |
| 1 | -1 | 1 | -1 | -1 | 1 | -1 | 1 | 180 | 203 | 175 | 186.0 | 148.667 |
| 1 | 1 | -1 | -1 | -1 | -1 | 1 | 1 | 178 | 205 | 205 | 196.0 | 162.0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 203 | 207 | 188 | 199.333 | 66.889 |

y = 193.79162 + 4.29162*x1 + -3.45838*x2 + 1.62513*x3 + 0.70812*x1x2 + 2.79162*x1x3 + 0.04162*x2x3 + -0.45838*x1x2x3

| X0 | X1 | X2 | X3 | X1X2 | X1X3 | X2X3 | X1X2X3 | Y1 | Y2 | Y3 | Y | S^2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 15 | -25 | -45 | -375 | -675 | 1125 | 16875 | 210 | 178 | 198 | 195.333 | 174.222 |
| 1 | 15 | 10 | 50 | 150 | 750 | 500 | 7500 | 173 | 199 | 182 | 184.667 | 116.222 |
| 1 | 45 | -25 | 50 | -1125 | 2250 | -1250 | -56250 | 193 | 209 | 215 | 205.667 | 86.222 |
| 1 | 45 | 10 | -45 | 450 | -2025 | -450 | -20250 | 204 | 196 | 174 | 191.333 | 160.889 |
| 1 | 15 | -25 | 50 | -375 | 750 | -1250 | -18750 | 204 | 180 | 192 | 192.0 | 96.0 |
| 1 | 15 | 10 | -45 | 150 | -675 | -450 | -6750 | 180 | 203 | 175 | 186.0 | 148.667 |
| 1 | 45 | -25 | -45 | -1125 | -2025 | 1125 | 50625 | 178 | 205 | 205 | 196.0 | 162.0 |
| 1 | 45 | 10 | 50 | 450 | 2250 | 500 | 22500 | 203 | 207 | 188 | 199.333 | 66.889 |

y = 183.30595 + 0.29723*x1 + -0.28143*x2 + -0.07468*x3 + 0.00279*x1x2 + 0.00364*x1x3 + 0.00115*x2x3 + -4e-05*x1x2x3

```
Gp =  0.1723076923076923  Gt =  0.815948432359917
Дисперсія однорідна!

t табличне =  2.119905299221011
Гіпотеза підтверджена, beta0 = 0
Гіпотеза не підтверджена.
beta1 = 4.291625
Гіпотеза не підтверджена.
beta2 = -3.458375
Гіпотеза не підтверджена.
beta3 = 1.6251250000000006
Гіпотеза не підтверджена.
beta4 = 0.708124999999999
Гіпотеза не підтверджена.
beta5 = 2.791625
Гіпотеза не підтверджена.
beta6 = 0.0416249999999998
Гіпотеза не підтверджена.
beta7 = -0.4583750000000002
Рівняння регресії неадекватне
```

Висновок: Я провів дробовий трьохфакторний експеримент. Склав матрицю планування, знайшов коефіцієнти рівняння регресії та провів 3 статистичні перевірки. Кінцева мета роботи досягнута.