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 № 4

з дисципліни «Методи оптимізації та планування експерименту»

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

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІО-92
Долготьор Алевтина
Варіант: 208

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

```
from random import randint
import numpy as np

# Cramer's rule
def cramer(arr, ins, pos):
    matrix = np.insert(np.delete(arr, pos, 1), pos, ins, 1)

    return np.linalg.det(matrix) / np.linalg.det(arr)

# Method to get dispersion
def getDispersion(y, y_r):
    return [round(sum([(y[i][j] - y_r[i]) ** 2 for j in range(len(y[i]))]) /
3, 3) for i in range(len(y_r))]

# Cochran criteria
def cochran(disp, m):
    Gp = max(disp) / sum(disp)
    Gt = [.6798, .5157, .4377, .3910, .3595, .3362, .3185, .3043, .2926,
.2829]

    return [round(Gp, 4), Gt[m - 2]]

# Student criteria
def student(disp, m, y_r, x_nT):
    table = {
        8: 2.306,
        16: 2.120,
        24: 2.064,
        'inf': 1.960
    }

    N = len(y_r)

    Sb = sum(disp) / len(y_r)
    Sbeta = (Sb / (m * N)) ** (1 / 2)

    beta = [sum([y_r[j] * x_nT[i][j] for j in range(N)]) / N for i in
range(N)]
    t = [abs(beta[i]) / Sbeta for i in range(len(beta))]

    f3 = N * (m - 1)

    if f3 > 30:
        t_t = table['inf']
    elif f3 > 0:
        t_t = table[f3]
    else:
        return

    result = []
    for i in t:
        if i < t_t:
            result.append(False)
        else:
            result.append(True)

    return result
```

```python
# Fisher criteria
def fisher(y_r, y_st, b_det, disp, m):
    table = {
        8: [5.3, 4.5, 4.1, 3.8, 3.7, 3.6, 3.3],
        16: [4.5, 3.6, 3.2, 3.0, 2.9, 2.7, 2.4],
        24: [4.3, 3.4, 3.0, 2.8, 2.6, 2.5, 2.2],
    }

    N = len(y_r)
    Sb = sum(disp) / N
    d = 0
    for b in b_det:
        if b:
            d += 1

    f4 = N - d
    f3 = N * (m - 1)
    Sad = (m / f4) * sum([(y_st[i] - y_r[i]) ** 2 for i in range(N)])
    Fap = Sad / Sb
    Ft = table[f3][f4 - 1]

    if Fap < Ft:
        return f"\nРівняння регресії адекватно оригіналу:\nFap < Ft:
{round(Fap, 2)} < {Ft}"
    else:
        return f"\nРівняння регресії неадекватно оригіналу: \nFap > Ft:
{round(Fap, 2)} > {Ft}"

# Main function
def experiment(m, min_x1, max_x1, min_x2, max_x2, min_x3, max_x3):
    y_min = round((min_x1 + min_x2 + min_x3) / 3) + 200
    y_max = round((max_x1 + max_x2 + max_x3) / 3) + 200

    x_norm = np.array([
        [+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]
    ])

    x = np.array([
        [min_x1, min_x2, min_x3, min_x1 * min_x2, min_x1 * min_x3, min_x2 *
min_x3, min_x1 * min_x2 * min_x3],
        [min_x1, max_x2, max_x3, min_x1 * max_x2, min_x1 * max_x3, max_x2 *
max_x3, min_x1 * max_x2 * max_x3],
        [max_x1, min_x2, max_x3, max_x1 * min_x2, max_x1 * max_x3, min_x2 *
max_x3, max_x1 * min_x2 * max_x3],
        [max_x1, max_x2, min_x3, max_x1 * max_x2, max_x1 * min_x3, max_x2 *
min_x3, max_x1 * max_x2 * min_x3],
        [min_x1, min_x2, max_x3, min_x1 * min_x2, min_x1 * max_x3, min_x2 *
max_x3, min_x1 * min_x2 * max_x3],
        [min_x1, max_x2, min_x3, min_x1 * max_x2, min_x1 * min_x3, max_x2 *
```

```
min_x3, min_x1 * max_x2 * min_x3],
        [max_x1, min_x2, min_x3, max_x1 * min_x2, max_x1 * min_x3, min_x2 *
min_x3, max_x1 * min_x2 * min_x3],
        [max_x1, max_x2, max_x3, max_x1 * max_x2, max_x1 * max_x3, max_x2 *
max_x3, max_x1 * max_x2 * max_x3]
    ])

    x_normT = x_norm.T
    xT = x.T

    N = len(x)
    y = [[randint(y_min, y_max) for _ in range(m)] for _ in range(N)]
    y_r = [round(sum(y[i]) / len(y[i]), 2) for i in range(N)]

    disp = getDispersion(y, y_r)
    cochran_cr = cochran(disp, m)

    # Get coefficients
    x1 = xT[0]
    x2 = xT[1]
    x3 = xT[2]
    yi = np.array(y_r)

    # Solve SoLE by Cramer's rule
    b_delta = [[N, sum(x1), sum(x2), sum(x3), sum(x1 * x2), sum(x1 * x3),
sum(x2 * x3), sum(x1 * x2 * x3)],
               [sum(x1), sum(x1 ** 2), sum(x1 * x2), sum(x1 * x3), sum(x1 ** 2
* x2), sum(x1 ** 2 * x3),
                sum(x1 * x2 * x3), sum(x1 ** 2 * x2 * x3)],
               [sum(x2), sum(x1 * x2), sum(x2 ** 2), sum(x2 * x3), sum(x1 * x2
** 2), sum(x1 * x2 * x3),
                sum(x2 ** 2 * x3), sum(x1 * x2 ** 2 * x3)],
               [sum(x3), sum(x1 * x3), sum(x2 * x3), sum(x3 ** 2), sum(x1 * x2
* x3), sum(x1 * x3 ** 2),
                sum(x2 * x3 ** 2), sum(x1 * x2 * x3 ** 2)],
               [sum(x1 * x2), sum(x1 ** 2 * x2), sum(x1 * x2 ** 2), sum(x1 *
x2 * x3), sum(x1 ** 2 * x2 ** 2),
                sum(x1 ** 2 * x2 * x3), sum(x1 * x2 ** 2 * x3), sum(x1 ** 2 *
x2 ** 2 * x3)],
               [sum(x1 * x3), sum(x1 ** 2 * x3), sum(x1 * x2 * x3), sum(x1 *
x3 ** 2), sum(x1 ** 2 * x2 * x3),
                sum(x1 ** 2 * x3 ** 2), sum(x1 * x2 * x3 ** 2), sum(x1 ** 2 *
x2 * x3 ** 2)],
               [sum(x2 * x3), sum(x1 * x2 * x3), sum(x2 ** 2 * x3), sum(x2 *
x3 ** 2), sum(x1 * x2 ** 2 * x3),
                sum(x1 * x2 * x3 ** 2), sum(x2 ** 2 * x3 ** 2), sum(x1 * x2 **
2 * x3 ** 2)],
               [sum(x1 * x2 * x3), sum(x1 ** 2 * x2 * x3), sum(x1 * x2 ** 2 *
x3), sum(x1 * x2 * x3 ** 2),
                sum(x1 ** 2 * x2 ** 2 * x3), sum(x1 ** 2 * x2 * x3 ** 2),
sum(x1 * x2 ** 2 * x3 ** 2),
                sum(x1 ** 2 * x2 ** 2 * x3 ** 2)]]

    b_set = [sum(yi), sum(yi*x1), sum(yi*x2), sum(yi*x3), sum(yi*x1*x2),
sum(yi*x1*x3), sum(yi*x2*x3), sum(yi*x1*x2*x3)]
    b = [cramer(b_delta, b_set, i) for i in range(N)]
```

```
    x1_norm = x_normT[0]
    x2_norm = x_normT[1]
    x3_norm = x_normT[2]

    b_norm = [sum(yi)/N, sum(yi*x1_norm/N), sum(yi*x2_norm)/N,
sum(yi*x3_norm)/N,
              sum(yi*x1_norm*x2_norm)/N, sum(yi*x1_norm*x3_norm)/N,
sum(yi*x2_norm*x3_norm)/N, sum(yi*x1*x2*x3_norm)/N]

    b_det = student(disp, m, y_r, x_normT)
    b_cut = b.copy()

    # Simplified equations
    if b_det is None:
        return
    else:
        for i in range(N):
            if not b_det[i]:
                b_cut[i] = 0

        y_st = [round(sum([b_cut[0]] + [x[i][j] * b_cut[j + 1] for j in
range(N - 1)]), 2) for i in range(N)]

    # Calculate F-test
    fisher_cr = fisher(y_r, y_st, b_det, disp, m)

    # Print out results
    print(f"\nМатриця планування для m = {m}:")
    for i in range(m):
        print(f"Y{i + 1} - {np.array(y).T[i]}")

    print(f"\nСередні значення функції відгуку за рядками:\nY_R: {y_r}")
    print(f"\nКоефіцієнти рівняння регресії:")
    for i in range(len(b)):
        print(f"b{i} = {round(b[i], 3)}")

    if cochran_cr[0] < cochran_cr[1]:
        print(f"\nЗа критерієм Кохрена дисперсія однорідна:\nGp < Gt -
{cochran_cr[0]} < {cochran_cr[1]}")
    else:
        print(f"\nЗа критерієм Кохрена дисперсія неоднорідна:\nGp > Gt -
{cochran_cr[0]} > {cochran_cr[1]}"
              f"\nСпробуйте збільшити кількість експериментів.")
        return

    print(f"\nЗа критерієм Стьюдента коефіцієнти ", end="")
    for i in range(len(b_det)):
        if not b_det[i]:
            print(f"b{i} ", end="")
    print("приймаємо незначними")

    print(f"\nОтримані функції відгуку зі спрощеними коефіцієнтами:\nY_St -
{y_st}")
    print(fisher_cr)

    return True
```

```python
if __name__ == '__main__':
    Min_x1, Max_x1 = -5, 15
    Min_x2, Max_x2 = -35, 10
    Min_x3, Max_x3 = -35, 10

    M = 3

    experiment(M, Min_x1, Max_x1, Min_x2, Max_x2, Min_x3, Max_x3)
```

Результати виконання програми:

```
Матриця планування для m = 3:
Y1 - [182 212 207 201 209 197 197 211]
Y2 - [182 191 195 200 177 207 187 178]
Y3 - [193 182 209 196 198 189 191 211]

Середні значення функції відгуку за рядками:
Y_R: [185.67, 195.0, 203.67, 199.0, 194.67, 197.67, 191.67, 200.0]

Коефіцієнти рівняння регресії:
b0 = 196.132
b1 = 0.259
b2 = 0.04
b3 = 0.016
b4 = -0.005
b5 = 0.004
b6 = -0.006
b7 = 0.0

За критерієм Кохрена дисперсія однорідна:
Gp < Gt - 0.3375 < 0.5157

За критерієм Стьюдента коефіцієнти b1 b2 b3 b4 b5 b6 b7 приймаємо незначними

Отримані функції відгуку зі спрощеними коефіцієнтами:
Y_St - [196.13, 196.13, 196.13, 196.13, 196.13, 196.13, 196.13, 196.13]

Рівняння регресії адекватно оригіналу:
Fap < Ft: 1.03 < 2.4

Process finished with exit code 0
```