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 № 3

з дисципліни «МНД» на тему
«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ
ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ.»

ВИКОНАВ:
студент II курсу ФІОТ
групи ІВ-91
Чопик Н.О.
Залікова - 9130

ПЕРЕВІРИВ:
ас. Регіда П. Г.

Київ – 2021

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

Завдання:

1. Скласти матрицю планування для дробового трьохфакторного експерименту. Провести експеримент в усіх точках факторного простору, повторивши N експериментів, де N – кількість експериментів (рядків матриці планування) в усіх точках факторного простору – знайти значення функції відгуку У. Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі (випадковим чином).

$y_{max} = 200 + x_{cp\ max}$;

$y_{min} = 200 + x_{cp\ min}$

де $x_{cp\,max} = \frac{x_{1max} + x_{2max} + x_{3max}}{3}$, $x_{cp\,min} = \frac{x_{1min} + x_{2min} + x_{3min}}{3}$

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.

3. Провести 3 статистичні перевірки.

4. Написати комп'ютерну програму, яка усе це виконує.

Варіант: 128

№ варіанта	X1		X2		X3	
	min	max	min	max	min	Max
128	-15	30	30	80	30	35

Лістинг програми:

```
from random import randint
from scipy import linalg
from scipy.stats import t as criterion_t
from scipy.stats import f as criterion_f
from math import sqrt


def get_x_matrix(planning_matrix, x_list):
    result = []
    for i in range(len(planning_matrix)):
        row = []
        for j in range(1, len(planning_matrix[i])):
            if planning_matrix[i][j] == -1:
                row.append(x_list[j - 1][0])
            else:
                row.append(x_list[j - 1][1])
        result.append(row)
    return result


def get_average(matrix):
    return [round(sum(matrix[k]) / m, 4) for k in range(n)]


def get_dispersion(matrix):
    return [round(sum([((i - get_average(matrix)[j]) ** 2) for i in matrix[j]]) / m,
4) for j in range(n)]


def get_mx_my(matrix_x, mat_y):
    mx = [sum(matrix_x[i][k] for i in range(n)) / n for k in range(m)]
    my = sum(get_average(mat_y)) / n
    return mx, my


def get_ai_aii(transposed_x_matrix, matrix_y):
    ai = [round(sum(transposed_x_matrix[k][i] * get_average(matrix_y)[i] for i in
range(n)) / n, 4) for k in range(m)]
    aii = [round(sum(transposed_x_matrix[k][i] ** 2 for i in range(n)) / n, 4) for k
in range(m)]
    return ai, aii


def find_cf():
    tran = transposed_x_matrix
    mx, my = get_mx_my(matrix_x, matrix_y)
    ai, aii = get_ai_aii(transposed_x_matrix, matrix_y)

    a12 = a21 = (tran[0][0] * tran[1][0] + tran[0][1] * tran[1][1] + tran[0][2] *
tran[1][2] + tran[0][3] * tran[1][3]) / n
    a13 = a31 = (tran[0][0] * tran[2][0] + tran[0][1] * tran[2][1] + tran[0][2] *
tran[2][2] + tran[0][3] * tran[2][3]) / n
    a23 = a32 = (tran[1][0] * tran[2][0] + tran[1][1] * tran[2][1] + tran[1][2] *
tran[2][2] + tran[1][3] * tran[2][3]) / n

    delta_b = linalg.det(
        [[1, mx[0], mx[1], mx[2]],
         [mx[0], aii[0], a12, a13],
         [mx[1], a12, aii[1], a32],
         [mx[2], a13, a23, aii[2]]])
```

```
    b0 = linalg.det([[my, mx[0], mx[1], mx[2]],
                     [ai[0], aii[0], a12, a13],
                     [ai[1], a12, aii[1], a32],
                     [ai[2], a13, a23, aii[2]]]) / delta_b

    b1 = linalg.det([[1, my, mx[1], mx[2]],
                     [mx[0], ai[0], a12, a13],
                     [mx[1], ai[1], aii[1], a32],
                     [mx[2], ai[2], a23, aii[2]]]) / delta_b

    b2 = linalg.det([[1, mx[0], my, mx[2]],
                     [mx[0], aii[0], ai[0], a13],
                     [mx[1], a12, ai[1], a32],
                     [mx[2], a13, ai[2], aii[2]]]) / delta_b

    b3 = linalg.det([[1, mx[0], mx[1], my],
                     [mx[0], aii[0], a12, ai[0]],
                     [mx[1], a12, aii[1], ai[1]],
                     [mx[2], a13, a23, ai[2]]]) / delta_b

    check = [round(b0 + b1 * tran[0][i] + b2 * tran[1][i] + b3 * tran[2][i], 4) for i
in range(4)]
    b_list = [round(b0, 4), round(b1, 4), round(b2, 4), round(b3, 4)]
    return check, b_list


def show_matrix(matrix):
    result = ""
    for row in matrix:
        result += f"{row}\n"
    print(result)


m = 3
n = 4
d = 4
x1 = [-15, 30]
x2 = [30, 80]
x3 = [30, 35]
x_list = [x1, x2, x3]

planning_matrix = [[1, -1, -1, -1],
                   [1, -1, 1, 1],
                   [1, 1, -1, 1],
                   [1, 1, 1, -1]]

matrix_x = get_x_matrix(planning_matrix, x_list)

transposed_planning_matrix = [list(i) for i in zip(*planning_matrix)]
transposed_x_matrix = [list(i) for i in zip(*matrix_x)]

x_min_max_av = [sum(x_list[i][k] for i in range(3)) / 3 for k in range(2)]
y_min_max = [int(200 + x_min_max_av[i]) for i in range(2)]

matrix_y = [[randint(y_min_max[0], y_min_max[1]) for _ in range(m)] for _ in
range(n)]

f1 = m - 1
f2 = n
f3 = f1 * f2
f4 = n - d
```

```
matrix_disY = get_dispersion(matrix_y)
check, b_list = find_cf()

print('Матриця планування:')
show_matrix(matrix_x)

print(f"Рівняння регресії\ny = {b_list[0]} + {b_list[1]}*x1 + {b_list[2]}*x2 +
{b_list[3]}*x3")

print('\nПеревірка однорідності за Кохрена:')
if max(matrix_disY) / sum(matrix_disY) < 0.7679:
    print('Дисперсія однорідна:', max(matrix_disY) / sum(matrix_disY), "< 0.7679")
else:
    print('Дисперсія неоднорідна - ', max(matrix_disY) / sum(matrix_disY), " >
0.7679")

print('\nПеревірка значущості коефіцієнтів за критерієм Стьюдента:')
S2b = sum(matrix_disY) / n
S2bs = S2b / (m * n)
Sbs = sqrt(S2bs)
bb = [sum(get_average(matrix_y)[k] * transposed_planning_matrix[i][k] for k in
range(n)) / n for i in range(n)]
t = [abs(bb[i]) / Sbs for i in range(n)]
for i in range(n):
    print(f"t{i} = {b_list[i]}")
    if t[i] < criterion_t.ppf(q=0.975, df=f3):
        b_list[i] = 0
        d -= 1

y_regression = [b_list[0] + b_list[1] * matrix_x[i][0] + b_list[2] * matrix_x[i][1] +
b_list[3] * matrix_x[i][2] for i in range(n)]

print('Значення y:')
for i in range(n):
    print(f"{b_list[0]} + {b_list[1]}*x1 + {b_list[2]}*x2 + {b_list[3]}*x3 = "
          f"{b_list[0] + b_list[1] * matrix_x[i][0] + b_list[2] * matrix_x[i][1] +
b_list[3] * matrix_x[i][2]}")

print('\nПеревірка адекватності за Фішером:')
Sad = (m / (n - d)) * int(sum(y_regression[i] - get_average(matrix_y)[i] for i in
range(n)) ** 2)
Fp = Sad / S2b
q = 0.05
F_table = criterion_f.ppf(q=1 - q, dfn=f4, dfd=f3)
print('FP  =', Fp)

if Fp > F_table:
    print('Рівняння регресії неадекватно оригіналу при рівні значимості 0.05')
else:
    print('Рівняння регресії адекватно оригіналу при рівні значимості 0.05')
```

Контрольні запитання:

1.Що називається дробовим факторним експериментом?

Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту

2.Для чого потрібно розрахункове значення Кохрена?

Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

3. Для чого перевіряється критерій Стьюдента?

За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння

4.Чим визначається критерій Фішера і як його застосовувати?

Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваного об”єкта.

Результат виконання роботи:

```
"C:\Users\Nazar\Desktop\КПІ\Методи наукових досліджень\venv\Scripts\python.exe"
Матриця планування:
[-15, 30, 30]
[-15, 80, 35]
[30, 30, 35]
[30, 80, 30]

Рівняння регресії
y = 214.6444 + 0.1185*x1 + 0.06*x2 + 0.3333*x3

Перевірка однорідності за Кохрена:
Дисперсія однорідна: 0.40085461538461536 < 0.7679

Перевірка значущості коефіцієнтів за критерієм Стьюдента:
t0 = 214.6444
t1 = 0.1185
t2 = 0.06
t3 = 0.3333
Значення y:
214.6444 + 0*x1 + 0*x2 + 0*x3 = 214.6444
214.6444 + 0*x1 + 0*x2 + 0*x3 = 214.6444
214.6444 + 0*x1 + 0*x2 + 0*x3 = 214.6444
214.6444 + 0*x1 + 0*x2 + 0*x3 = 214.6444

Перевірка адекватності за Фішером:
FP  = 55.53846153846154
Рівняння регресії адекватно оригіналу при рівні значимості 0.05
```