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 **«Методи оптимізації та планування експерименту »** на тему

«Проведення трьохфакторного експерименту з використанням лінійного рівняння регресії»

Виконала:
студентка II курсу ФІОТ
групи ІО-93
Дяченко Віта
У списку групи №9

Перевірив:
Регіда П. Г.

Київ – 2021

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

Завдання на лабораторну роботу:

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).
2.Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.
3. Провести 3 статистичні перевірки.

Варіант:

Варіант:	min	max	min	max	min	max
309	-20	15	-35	10	10	20

Код програми:

```
from random import *
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial


class FractionalExperiment:
# Проведення дробового трьохфакторного експерименту
    def __init__(self, n, m):
        self.n = n
        self.m = m
        self.x_range = [[-20, 15], [-35, 10], [10, 20]]
        # Xcp max та Xcp мін
        self.x_min = (- 20 - 35 + 10) / 3
        self.x_max = (15 + 10 + 20) / 3
        # y макс та y мін
        self.y_max = round(200 + self.x_max)
        self.y_min = round(200 + self.x_min)
        # матриця планування ПФЕ
        self.x_n = [[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]]

        self.y = np.zeros(shape=(self.n, self.m))
        self.y_new_values = []
        for i in range(self.n):
            for j in range(self.m):
                self.y[i][j] = randint(self.y_min, self.y_max)
        # середнє значення y
        self.y_av = [round(sum(i) / len(i), 2) for i in self.y]

        self.x_n = self.x_n[:len(self.y)]
        self.x = np.ones(shape=(len(self.x_n), len(self.x_n[0])))

        for i in range(len(self.x_n)):
            for j in range(1, len(self.x_n[i])):
                if self.x_n[i][j] == -1:
                    self.x[i][j] = self.x_range[j - 1][0]
                else:
                    self.x[i][j] = self.x_range[j - 1][1]
        self.f1 = m - 1
        self.f2 = n
        self.f3 = self.f1 * self.f2
        self.q = 0.05

# підстановка у регресію
    def regr(self, x, b):
        y = sum([x[i] * b[i] for i in range(len(x))])
        return y

# Розрахунок коефіцієнтів рівняння регресії
    def count_koefitients(self):
        mx1 = sum(self.x[:, 1]) / self.n
        mx2 = sum(self.x[:, 2]) / self.n
        mx3 = sum(self.x[:, 3]) / self.n
        my = sum(self.y_av) / self.n
        a12 = sum([self.x[i][1] * self.x[i][2] for i in range(len(self.x))]) /
self.n
        a13 = sum([self.x[i][1] * self.x[i][3] for i in range(len(self.x))]) /
self.n
        a23 = sum([self.x[i][2] * self.x[i][3] for i in range(len(self.x))]) /
self.n
        a11 = sum([i ** 2 for i in self.x[:, 1]]) / self.n
        a22 = sum([i ** 2 for i in self.x[:, 2]]) / self.n
        a33 = sum([i ** 2 for i in self.x[:, 3]]) / self.n
        a1 = sum([self.y_av[i] * self.x[i][1] for i in range(len(self.x))]) /
self.n
        a2 = sum([self.y_av[i] * self.x[i][2] for i in range(len(self.x))]) /
self.n
        a3 = sum([self.y_av[i] * self.x[i][3] for i in range(len(self.x))]) /
self.n
```

```
        X = [[1, mx1, mx2, mx3], [mx1, a11, a12, a13], [mx2, a12, a22, a23],
[mx3, a13, a23, a33]]
        Y = [my, a1, a2, a3]
        B = [round(i, 2) for i in solve(X, Y)]
        print('\nРівняння регресії')
        print(f'y = {B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

        return B
# Розрахунок дисперсії
    def dispersion(self):
        res = []
        for i in range(self.n):
            s = sum([(self.y_av[i] - self.y[i][j]) ** 2 for j in
range(self.m)]) / self.m
            res.append(s)
        return res
# Перевірка за критерієм Кохрена
    def kohren(self):
        q1 = self.q / self.f1
        fisher_value = f.ppf(q=1 - q1, dfn=self.f2, dfd=(self.f1 - 1) *
self.f2)
        G_cr = fisher_value / (fisher_value + self.f1 - 1)
        s = self.dispersion()
        Gp = max(s) / sum(s)
        return Gp, G_cr

    def student(self):
        # Перевірка за критерієм Стьюдента
        def bs():
            res = [sum(1 * y for y in self.y_av) / self.n]
            for i in range(3):  # 4 - ксть факторів
                b = sum(j[0] * j[1] for j in zip(self.x[:, i], self.y_av)) /
self.n
                res.append(b)
            return res

        S_kv = self.dispersion()
        s_kv_aver = sum(S_kv) / self.n

        # статистична оцінка дисперсії
        s_Bs = (s_kv_aver / self.n / self.m) ** 0.5
        Bs = bs()
        ts = [abs(B) / s_Bs for B in Bs]
        return ts
# Перевірка адекватності за критерієм Фішера
    def fisher(self, d):
        S_ad = self.m / (self.n - d) * sum([(self.y_new_values[i] -
self.y_av[i]) ** 2 for i in range(len(self.y))])
        S_kv = self.dispersion()
        S_kv_aver = sum(S_kv) / self.n
        F_p = S_ad / S_kv_aver
        return F_p
```

```
    def check(self):
        # Проведення статистичних перевірок
        student = partial(t.ppf, q=1 - 0.025)
        t_student = student(df=self.f3)

        print('\nПеревірка за критерієм Кохрена')
        Gp, G_kr = self.kohren()
        print(f'Gp = {Gp}')
        if Gp < G_kr:
            print(f'З ймовірністю {1-self.q} дисперсії однорідні.')
        else:
            print("Необхідно збільшити кількість дослідів")
            self.m += 1
            FractionalExperiment(self.n, self.m)

        ts = self.student()
        print('\nПеревірка значущості коефіцієнтів за критерієм Стьюдента')
        print('Критерій Стьюдента:\n', ts)
        res = [t for t in ts if t > t_student]
        B = self.count_koefitients()
        final_k = [B[ts.index(i)] for i in ts if i in res]
        print('Коефіцієнти {} статистично незначущі, тому ми виключаємо їх з
рівняння.'.format(
            [i for i in B if i not in final_k]))

        for j in range(self.n):
            self.y_new_values.append(self.regr([self.x[j][ts.index(i)] for i
in ts if i in res], final_k))

        print(f'\nЗначення "y" з коефіцієнтами {final_k}')
        print(self.y_new_values)

        d = len(res)
        f4 = self.n - d
        F_p = self.fisher(d)

        fisher = partial(f.ppf, q=1 - 0.05)
        f_t = fisher(dfn=f4, dfd=self.f3)
        print('\nПеревірка адекватності за критерієм Фішера')
        print('Fp =', F_p)
        print('F_t =', f_t)
        if F_p < f_t:
            print('Математична модель адекватна експериментальним даним')
        else:
            print('Математична модель не адекватна експериментальним даним')


experiment = FractionalExperiment(7, 8)
experiment.check()
```

Результат роботи:

```
Перевірка за критерієм Кохрена
Gp = 0.1920724433968133
З ймовірністю 0.95 дисперсії однорідні.

Перевірка значущості коефіцієнтів за критерієм Стьюдента
Критерій Стьюдента:
 [175.347613921568, 175.347613921568, 870.9232401598462, 2761.265771864493]

Рівняння регресії
y = 198.69 + 0.03*x1 + -0.0*x2 + 0.18*x3
Коефіцієнти [0.03] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [198.69, 198.69, -0.0, 0.18]
[399.18, 400.98, 400.98, 399.18, 400.98, 399.18, 399.18]

Перевірка адекватності за критерієм Фішера
Fp = 10005.381783453144
F_t = 2.7939488515842408
Математична модель не адекватна експериментальним даним

Process finished with exit code 0
```

Висновок: В процесі виконання роботи було проведено дробовий трьохфакторний експеримент. Склали матрицю планування, знайшли коефіцієнти рівняння регресії та провели перевірку однорідність дисперсії за критерієм Кохрена, нуль-гіпотезу за критерієм Стьюдента та адекватність моделі за критерієм Фішера.

Контрольні запитання:

1. *Що називається дробовим факторним експериментом?*

Дробовий факторний експеримент – це частина ПФЕ, який мінімізує число дослідів, за рахунок тієї інформації, яка не дуже істотна для побудови лінійної моделі.

2. *Для чого потрібно розрахункове значення Кохрена?*

Критерій Кохрена — використовують для порівняння трьох і більше вибірок однакового обсягу n.
Якщо вибіркові дисперсії отримані за вибірками однакових обсягів, для їх порівняння використовують більш зручний і точний критерій Кохрена.
Кохрена досліджував розподіл максимальної вибіркової дисперсії до суми всіх дисперсій.

3. *Для чого перевіряється критерій Стьюдента?*

Критерій Стьюдента - загальна назва для статистичних тестів, в яких статистика критерію має розподіл Стьюдента. Найбільш часто t-критерії застосовуються для перевірки рівності середніх значень у двох вибірках. Тому перед застосуванням критерію Стьюдента рекомендується виконати перевірку нормальності.

4. *Чим визначається критерій Фішера і як його застосовувати?*

Критерій Фішера застосовується для перевірки рівності дисперсій двох вибірок. Його відносять до критеріїв розсіювання. Критерій Фішера заснований на додаткових припущеннях про незалежність і нормальності вибірок даних. Перед його застосуванням рекомендується виконати перевірку нормальності.