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

«ЗАГАЛЬНІ ПРИНЦИПИ ОРГАНІЗАЦІЇ ЕКСПЕРИМЕНТІВ З ДОВІЛЬНИМИ ЗНАЧЕННЯМИ ФАКТОРІВ»

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

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Тема: ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ

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

Завдання:

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).

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

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

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.

Хід роботи:

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
322	10	40	30	80	10	20

Xср max = (40+80+20)/3 = 46.6

Xср min = (10+30+10)/3 = 16.6

Ymax = 200 + 46.6 = 246.6

Ymin = 200 + 16.6 = 216.6

Лістинг 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_min = (10 + 30 + 10) / 3
        self.x_max = (40 + 80 + 20) / 3
        self.y_max = round(200 + self.x_max)
        self.y_min = round(200 + self.x_min)
        self.x_norm = [[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.x_range = [(10, 40), (30, 80), (10, 20)]
        self.y = np.zeros(shape=(self.n, self.m))
        self.y_new = []
        for i in range(self.n):
            for j in range(self.m):
                self.y[i][j] = randint(self.y_min, self.y_max)
        self.y_av = [round(sum(i) / len(i), 2) for i in self.y]
        self.x_norm = self.x_norm[:len(self.y)]
        self.x = np.ones(shape=(len(self.x_norm), len(self.x_norm[0])))
        for i in range(len(self.x_norm)):
            for j in range(1, len(self.x_norm[i])):
                if self.x_norm[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

    def regression(self, x, b):
        """Підстановка коефіцієнтів у рівняння регресії"""
        y = sum([x[i] * b[i] for i in range(len(x))])
        return y

    def count_koefs(self):
        """Розрахунок коефіцієнтів рівняння регресії"""
        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

    def dispersion(self):
        """Розрахунок дисперсії"""
        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

    def kohren(self):
        """Перевірка однорідності дисперсій за критерієм Кохрена"""
        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 знащущості коефіцієнтів 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

    def fisher(self, d):
        """Перевірка адекватності за критерієм Фішера"""
        S_ad = self.m / (self.n - d) * sum([(self.y_new[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_koefs()
        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.append(self.regression([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)

        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)  # табличне знач
        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 виконання роботи:

```
Перевірка за критерієм Кохрена
Gp = 0.2002890135538454
З ймовірністю 0.95 дисперсії однорідні.

Перевірка значущості коефіцієнтів за критерієм Стьюдента
Критерій Стьюдента:
 [202.60199005836662, 202.60199005836662, 4618.999076553124, 10454.819427784518]

Рівняння регресії
y = 229.92 + -0.04*x1 + 0.06*x2 + 0.06*x3
Коефіцієнти [-0.04] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [229.92, 229.92, 0.06, 0.06]
[462.24, 465.84, 462.84, 465.24, 462.84, 465.24, 462.24]

Перевірка адекватності за критерієм Фішера
Fp = 13398.705411466297
F_t = 2.7939488515842408
Математична модель не адекватна експериментальним даним

Process finished with exit code 0
```

Контрольні запитання:

1. Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту.

2. Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

3. За допомогою критерію Стьюдента перевіряється значущість коефіцієнтів рівняння регресії.

4. Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваному об’єкту.