Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 5

з дисципліни «Методи наукових досліджень»

на тему «ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ КВАДРАТИЧНИХ ЧЛЕНІВ»

ВИКОНАВ:

студент 2 курсу

групи ІВ-91

Бойко М. І.

Залікова – 9102

ПЕРЕВІРИВ:

ас. Регіда П. Г.

Київ – 2021

Лабораторна робота №5

Мета: Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Завдання:

1. Взяти рівняння з урахуванням квадратичних членів.

2. Скласти матрицю планування для ОЦКП

3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.

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

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

4. Розрахувати коефіцієнти рівняння регресії і записати його.

5. Провести 3 статистичні перевірки.

Група: ІВ-91

Номер у списку: 2

Варіант – 102

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
102	-8	9	-6	2	-1	5

Роздруківка коду програми:

```
import random
import sklearn.linear_model as lm
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *
import pandas as pd
from tabulate import tabulate
from scipy.stats import f


def add_sq(x):
    for i in range(len(x)):
        x[i][4] = x[i][1] * x[i][2]
        x[i][5] = x[i][1] * x[i][3]
        x[i][6] = x[i][2] * x[i][3]
        x[i][7] = x[i][1] * x[i][3] * x[i][2]
        x[i][8] = x[i][1] ** 2
        x[i][9] = x[i][2] ** 2
        x[i][10] = x[i][3] ** 2
    return x


def regression(x, b):
    return sum([x[i] * b[i] for i in range(len(x))])


def criteria_studenta(x, y_aver):
    S_kv = s_kv(y_aver)
    s_kv_aver = sum(S_kv) / n

    s_Bs = (s_kv_aver / n / m) ** 0.5
    Bs = bs(x, y_aver)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts


def bs(x, y_aver):
    res = [sum(y_aver) / n]

    for i in range(len(x[0])):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n
        res.append(b)

    return res


def s_kv(y_aver):
    res = []
    for i in range(n):
        s = sum([(y_aver[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(round(s, 3))
    return res


def experiment(n, m):
    # Створення матриці

    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(y_min, y_max)
    if n > 14:
```

```
        no = n - 14
    else:
        no = 1

    x_norm = ccdesign(3, center=(0, no))
    x_norm = np.insert(x_norm, 0, 1, axis=1)

    for i in range(4, 11):
        x_norm = np.insert(x_norm, i, 0, axis=1)

    l = 1.215

    for i in range(len(x_norm)):
        for j in range(len(x_norm[i])):
            if x_norm[i][j] < -1 or x_norm[i][j] > 1:
                if x_norm[i][j] < 0:
                    x_norm[i][j] = -l
                else:
                    x_norm[i][j] = l

    x_norm = add_sq(x_norm)

    x = np.ones(shape=(len(x_norm), len(x_norm[0])), dtype=np.int64)

    x_range = [[x1_min, x1_max], [x2_min, x2_max], [x3_min, x3_max]]
    for i in range(8):
        for j in range(1, 4):
            if x_norm[i][j] == -1:
                x[i][j] = x_range[j - 1][0]
            else:
                x[i][j] = x_range[j - 1][1]

    for i in range(8, len(x)):
        for j in range(1, 3):
            x[i][j] = (x_range[j - 1][0] + x_range[j - 1][1]) / 2

    dx = [x_range[i][1] - (x_range[i][0] + x_range[i][1]) / 2 for i in
range(3)]

    x[8][1] = l * dx[0] + x[9][1]
    x[9][1] = -l * dx[0] + x[9][1]
    x[10][2] = l * dx[1] + x[9][2]
    x[11][2] = -l * dx[1] + x[9][2]
    x[12][3] = l * dx[2] + x[9][3]
    x[13][3] = -l * dx[2] + x[9][3]
    x = add_sq(x)

    show_arr = pd.DataFrame(x)
    print('\nМатриця X:\n', tabulate(show_arr, headers='keys',
tablefmt='psql'))

    show_arr = pd.DataFrame(x_norm)
    print('\nНормована матриця X:\n', tabulate(show_arr.round(0),
headers='keys', tablefmt='psql'))

    show_arr = pd.DataFrame(y)
    print('\nМатриця Y:\n', tabulate(show_arr, headers='keys',
tablefmt='psql'))

    y_average = [round(sum(i) / len(i), 3) for i in y]

    # Знайдемо коефіцієнти

    skm = lm.LinearRegression(fit_intercept=False)
```

```
    skm.fit(x, y_average)
    b = skm.coef_

    print('\nКоефіцієнти рівняння регресії:')
    b = [round(i, 3) for i in b]
    print("y = {} +{}*x1 +{}*x2 +{}*x3 + {}*x1*x2 + {}*x1*x3 + {}*x2*x3 +
b{}*x1*x2*x3 + {}x1^2 + {}x2^2 + {}x3^2\n".format(*b))
    print('\nРезультат рівняння зі знайденими коефіцієнтами:')
    print(np.dot(x, b))
    print("-" * 100)

    # Проведемо перевірку

    print('\nПеревірка рівняння:')
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05
    q1 = q / f1

    student = partial(t.ppf, q=1 - q)
    t_student = student(df=f3)

    fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    G_kr = fisher_value / (fisher_value + f1 - 1)

    y_average = [round(sum(i) / len(i), 3) for i in y]
    print('\nСереднє значення y:', y_average)

    disp = []
    for i in range(n):
        s = sum([(y_average[i] - y[i][j]) ** 2 for j in range(m)]) / m
        disp.append(round(s, 3))
    print('Дисперсія y:', disp)

    Gp = max(disp) / sum(disp)
    print('\nПеревіримо за критерієм Кохрена:')

    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'З ймовірністю {1 - q}')
    else:
        print("Збільшимо кількість дослідів")
        m += 1
        experiment(n, m)

    ts = criteria_studenta(x_norm[:, 1:], y_average)
    print('\nКритерій Стьюдента:')
    print(ts)
    res = [t for t in ts if t > t_student]
    final_k = [b[i] for i in range(len(ts)) if ts[i] in res]
    print('\nКоефіцієнти, які не мають статистичного значення:')
    print([round(i, 3) for i in b if i not in final_k])

    y_new = []
    for j in range(n):
        y_new.append(round(regression([x[j][i] for i in range(len(ts)) if
ts[i] in res], final_k), 3))

    print(f'\nЗначення y з коефіцієнтами: {final_k}')
    print(y_new)

    d = len(res)
    if d >= n:
```

```
            print('\nF4 <= 0')

        f4 = n - d

        s_ad = m / (n - d) * sum([(y_new[i] - y_average[i]) ** 2 for i in
range(len(y))])
        s_kv_aver = sum(disp) / n
        f_p = s_ad / s_kv_aver
        fisher = partial(f.ppf, q=0.95)
        f_t = fisher(dfn=f4, dfd=f3)
        print("-" * 100)
        print('\nПеревірка адекватності за критерієм Фішера')
        print('Fp =', f_p)
        print('F_t =', f_t)
        if f_p < f_t:
            print('Математична модель адекватна')
        else:
            print('Математична модель неадекватна!')


n = 15
m = 6

x1_min = -8
x1_max = 9

x2_min = -6
x2_max = 2

x3_min = -1
x3_max = 5

x_average_max = (x1_max + x2_max + x3_max) / 3
x_average_min = (x1_min + x2_min + x3_min) / 3

y_max = 200 + int(x_average_max)
y_min = 200 + int(x_average_min)

y = np.zeros(shape=(n, m))

experiment(n, m)
```

Приклад роботи програми:

Матриця X:

	0	1	2	3	4	5	6	7	8	9	10
0	1	-8	-6	-1	48	8	6	-48	64	36	1
1	1	9	-6	-1	-54	-9	6	54	81	36	1
2	1	-8	2	-1	-16	8	-2	16	64	4	1
3	1	9	2	-1	18	-9	-2	-18	81	4	1
4	1	-8	-6	5	48	-40	-30	240	64	36	25
5	1	9	-6	5	-54	45	-30	-270	81	36	25
6	1	-8	2	5	-16	-40	10	-80	64	4	25
7	1	9	2	5	18	45	10	90	81	4	25
8	1	10	-2	1	-20	10	-2	-20	100	4	1
9	1	-10	-2	1	20	-10	-2	20	100	4	1
10	1	0	2	1	0	0	2	0	0	4	1
11	1	0	-6	1	0	0	-6	0	0	36	1
12	1	0	-2	4	0	0	-8	0	0	4	16
13	1	0	-2	-2	0	0	4	0	0	4	4
14	1	0	-2	1	0	0	-2	0	0	4	1

Нормована матриця X:

	0	1	2	3	4	5	6	7	8	9	10
0	1	-1	-1	-1	1	1	1	-1	1	1	1
1	1	1	-1	-1	-1	-1	1	1	1	1	1
2	1	-1	1	-1	-1	1	-1	1	1	1	1
3	1	1	1	-1	1	-1	-1	-1	1	1	1
4	1	-1	-1	1	1	-1	-1	1	1	1	1
5	1	1	-1	1	-1	1	-1	-1	1	1	1
6	1	-1	1	1	-1	-1	1	-1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1
8	1	-1	0	0	-0	-0	0	-0	1	0	0
9	1	1	0	0	0	0	0	0	1	0	0
10	1	0	-1	0	-0	0	-0	-0	0	1	0
11	1	0	1	0	0	0	0	0	0	1	0
12	1	0	0	-1	0	-0	-0	-0	0	0	1
13	1	0	0	1	0	0	0	0	0	0	1
14	1	0	0	0	0	0	0	0	0	0	0

Матриця Y:

	0	1	2	3	4	5
0	202	201	197	199	203	202
1	199	201	198	199	204	202
2	195	196	197	195	204	203
3	204	203	203	195	195	203
4	198	205	195	203	202	200
5	201	200	197	198	205	201
6	205	198	202	201	199	205
7	204	205	203	199	195	197
8	201	197	197	201	203	195
9	205	203	201	203	200	204
10	200	205	199	204	205	199
11	203	195	197	196	199	196
12	198	196	203	204	205	205
13	202	198	196	201	202	202
14	205	202	199	198	201	200

Коефіцієнти рівняння регресії:

y = 200.471 +0.001*x1 +-0.173*x2 +0.186*x3 + 0.012*x1*x2 + -0.019*x1*x3 + 0.027*x2*x3 + b-0.004*x1*x2*x3 + 0.003x1^2 + -0.052x2^2 + 0.008x3^2

```
Результат рівняння зі знайденими коефіцієнтами:
[200.421 199.18  199.461 200.396 200.517 199.786 202.389 200.57  200.709
 201.389 200.165 199.669 201.265 200.377 200.749]
---------------------------------------------------------------------------------------------------

Перевірка рівняння:

Середнє значення y: [200.667, 200.5, 198.333, 200.5, 200.5, 200.333, 201.667, 200.5, 199.0, 202.667, 202.0, 197.667, 201.833, 200.167, 200.833]
Дисперсія y: [4.222, 4.25, 13.889, 15.25, 10.917, 6.556, 7.222, 13.917, 8.0, 2.889, 7.333, 7.222, 12.472, 5.472, 5.139]

Перевіримо за критерієм Кохрена:
Gp = 0.12224448897795594
З ймовірністю 0.95

Критерій Стьюдента:
[659.497, 1.123, 1.374, 0.214, 0.293, 0.731, 0.804, 0.731, 481.591, 480.943, 481.698]

Коефіцієнти, які не мають статистичного значення:
[0.001, -0.173, 0.186, 0.012, -0.019, 0.027, -0.004]

Значення y з коефіцієнтами: [200.471, 0.003, -0.052, 0.008]
[198.799, 198.85, 200.463, 200.514, 198.991, 199.042, 200.655, 200.706, 200.571, 200.571, 200.271, 198.607, 200.391, 200.295, 200.271]
---------------------------------------------------------------------------------------------------

Перевірка адекватності за критерієм Фішера
Fp = 1.8957716742576065
F_t = 1.9187589455788492
Математична модель адекватна
Process finished with exit code 0
```