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 № 5

з дисципліни «МНД» на тему
«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)..»

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

ПЕРЕВІРИВ:
ас. Регіда П. Г.

Київ – 2021

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.

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

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

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант: 128

№ варіанта	X1		X2		X3	
128	min	max	min	max	min	max
	-2	5	-6	4	-9	8

Лістинг програми:

```
import random
import sklearn.linear_model as lm
import numpy as np
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *
from beautifultable import BeautifulTable


def plan_matrix(n, m):
    y = np.zeros(shape=(n, m))
    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

    def add_sq_nums(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

    x_norm = add_sq_nums(x_norm)
    x = np.ones(shape=(len(x_norm), len(x_norm[0])), dtype=np.int64)
    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_nums(x)
    x_table = BeautifulTable()
    for i in range(n):
        x_table.rows.append([*x[i]])
    print('X matrix:')
    print(x_table)
    x_norm_table = BeautifulTable()
    for i in range(n):
        x_norm_table.rows.append([*x_norm[i]])
    print('Normalized x matrix:')
    print(x_norm_table)
    return x, y, x_norm


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


def s_kv(y, y_aver, n, m):
    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 coef_finding(x, y, norm=False):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(x, y)
    b = skm.coef_
    if norm == 1:
        print('\nКоефіцієнти рівняння регресії з нормованим x:')
    else:
        print('\nКоефіцієнти рівняння регресії:')
    b = [round(i, 3) for i in b]
    print(b)
    print('\nРезультат рівняння з знайденими коефіцієнтами:\n{}'.format(np.dot(x, b)))
    return b


def kohren_kr(y, y_aver, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    skv = s_kv(y, y_aver, n, m)
    gp = max(skv) / sum(skv)
    print('\nПеревірка  за Кохреном')
    return gp


def kohren(f1, f2, q=0.05):
    q1 = q / f1
    fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    return fisher_value / (fisher_value + f1 - 1)
```

```
def bs(x, y_aver, n):
    res = [sum(1 * y for y in 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 student_kr(x, y, y_aver, n, m):
    skv = s_kv(y, y_aver, n, m)
    skv_aver = sum(skv) / n
    sbs_tmp = (skv_aver / n / m) ** 0.5
    bs_tmp = bs(x, y_aver, n)
    ts = [round(abs(b) / sbs_tmp, 3) for b in bs_tmp]
    return ts


def fisher_kr(y, y_aver, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i in range(len(y))])
    skv = s_kv(y, y_aver, n, m)
    skv_aver = sum(skv) / n
    return S_ad / skv_aver


def check(x, y, b, n, m):
    print('\nПеревірка рівння:')
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05
    student = partial(t.ppf, q=1 - q)
    t_student = student(df=f3)
    g_kr = kohren(f1, f2)
    y_aver = [round(sum(i) / len(i), 3) for i in y]
    print('\nСереднє значення y:', y_aver)
    disp = s_kv(y, y_aver, n, m)
    print('Дисперсія y:', disp)
    gp = kohren_kr(y, y_aver, n, m)
    print(f'gp = {gp}')
    if gp < g_kr:
        print('З ймовірністю {} дисперсії є однорідними'.format(1 - q))
    else:
        print("Необхідно збільшити кількість експериментів")
        m += 1
        main(n, m)
    ts = student_kr(x[:, 1:], y, y_aver, n, m)
    print('\nКритерій Стюдента:\n{}:'.format(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('\nКоефієнти {} є статистично незначущими, тому ми виключаємо їх із
рівняння'.format([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('Значення y з коефіцієнтами {}: '.format(final_k))
    print(y_new)
    d = len(res)
    if d >= n:
        print('\nF4 <= 0')
        print('')
        return
```

```
    f4 = n - d
    f_p = fisher_kr(y, y_aver, y_new, n, m, d)
    fisher = partial(f.ppf, q=0.95)
    f_t = fisher(dfn=f4, dfd=f3)
    print('\nПеревірка адекватності за Фішером')
    print('fp =', f_p)
    print('ft =', f_t)
    if f_p < f_t:
        print('Математична модель адекватна експериментальним даним')
    else:
        print('Математична модель неадекватна експериментальним даним')


def main(n, m):
    x, y, x_norm = plan_matrix(n, m)
    y5_aver = [round(sum(i) / len(i), 3) for i in y]
    b = coef_finding(x, y5_aver)
    check(x_norm, y, b, n, m)


x_range = ((-2, 5), (-6, 4), (-9, 8))
x_aver_max = sum([x[1] for x in x_range]) / 3
x_aver_min = sum([x[0] for x in x_range]) / 3
y_min = 200 + int(x_aver_min)
y_max = 200 + int(x_aver_max)

main(15, 3)
```

Результат виконання роботи:

```
"C:\Users\Nazar\Desktop\КПІ\Методи наукових досліджень\venv\Scripts\python.exe"
X matrix:
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | -2 | -6 | -9 | 12  | 18  | 54  | -108 | 4  | 36 | 81  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 5  | -6 | -9 | -30 | -45 | 54  | 270  | 25 | 36 | 81  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | -2 | 4  | -9 | -8  | 18  | -36 |  72  | 4  | 16 | 81  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 5  | 4  | -9 | 20  | -45 | -36 | -180 | 25 | 16 | 81  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | -2 | -6 | 8  | 12  | -16 | -48 |  96  | 4  | 36 | 64  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 5  | -6 | 8  | -30 | 40  | -48 | -240 | 25 | 36 | 64  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | -2 | 4  | 8  | -8  | -16 | 32  | -64  | 4  | 16 | 64  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 5  | 4  | 8  | 20  | 40  | 32  | 160  | 25 | 16 | 64  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 5  | -1 | 1  | -5  |  5  | -1  |  -5  | 25 | 1  |  1  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | -3 | -1 | 1  |  3  | -3  | -1  |  3   | 9  | 1  |  1  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 1  | 5  | 1  |  5  |  1  |  5  |  5   | 1  | 25 |  1  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 1  | -7 | 1  | -7  |  1  | -7  |  -7  | 1  | 49 |  1  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 1  | -1 | 11 | -1  | 11  | -11 | -11  | 1  | 1  | 121 |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 1  | -1 | -9 | -1  | -9  |  9  |  9   | 1  | 1  | 81  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
| 1 | 1  | -1 | 1  | -1  |  1  | -1  |  -1  | 1  | 1  |  1  |
+---+----+----+----+-----+-----+-----+------+----+----+-----+
```

Normalized x matrix:

1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	-1.0	1.0	1.0	1.0
1.0	1.0	-1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	1.0	1.0
1.0	-1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	-1.0	1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0
1.0	-1.0	-1.0	1.0	1.0	-1.0	-1.0	1.0	1.0	1.0	1.0
1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	-1.0	1.0	1.0	1.0
1.0	-1.0	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	-1.215	0.0	0.0	-0.0	-0.0	0.0	-0.0	1.476	0.0	0.0
1.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476	0.0	0.0
1.0	0.0	-1.215	0.0	-0.0	0.0	-0.0	-0.0	0.0	1.476	0.0
1.0	0.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476	0.0
1.0	0.0	0.0	-1.215	0.0	-0.0	-0.0	-0.0	0.0	0.0	1.476
1.0	0.0	0.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476

```
Коефіцієнти рівняння регресії:
[199.798, -0.317, 0.215, 0.024, 0.023, 0.018, 0.004, 0.0, 0.089, 0.012, -0.008]

Результат рівняння з знайденими коефіцієнтами:
[199.882 197.432 200.972 200.132 199.406 199.098 201.176 202.478 200.222
 201.374 201.114 198.498 198.794 198.354 199.374]

Перевірка рівння:

Середнє значення y: [200.667, 198.333, 201.0, 200.333, 199.333, 199.333, 200.0, 202.0, 199.333, 201.667, 202.333, 197.0, 199.667, 196.333, 201.0]
Дисперсія y: [5.556, 11.556, 2.0, 0.222, 17.556, 5.556, 12.667, 2.667, 10.889, 16.222, 9.556, 4.667, 14.222, 3.556, 4.667]

Перевірка  за Кохреном
gp = 0.14442369548943315
З ймовірністю 0.95 дисперсії є однорідними

Критерій Стюдента:
[471.028, 0.288, 0.128, 0.584, 0.576, 0.786, 0.157, 0.052, 344.508, 344.121, 343.348]:

Коефієнти [-0.317, 0.215, 0.024, 0.023, 0.018, 0.004, 0.0] є статистично незначущими, тому ми виключаємо їх із рівняння
Значення y з коефіцієнтами [199.798, 0.089, 0.012, -0.008]:
[199.891, 199.891, 199.891, 199.891, 199.891, 199.891, 199.891, 199.891, 199.929, 199.929, 199.816, 199.816, 199.786, 199.786, 199.798]

Перевірка адекватності за Фішером
fp = 1.3650090079714396
ft = 2.125558760875511
Математична модель адекватна експериментальним даним
```