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 «Методи оптимізації та планування»

Тема: Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІО-92
Рожко М.М.

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

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 завдання:

N	X1		X2		X3	
	min	max	min	max	min	max
217	-1	6	-3	4	-3	7

Розруківка коду програми:

```
import random
import sklearn.linear_model as lm
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *
import plotly.figure_factory as ff


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

def make_table(labelData, data):
    info = [[labelData] + ["" for i in range(len(data[0])-1)]]
    for i in data: info.append(i)
    fig = ff.create_table(info)
    fig.show()

x_range = ((-1,6), (-3, 4), (-3, 7))

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_max = 200 + int(x_aver_max)
y_min = 200 + int(x_aver_min)

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 plan_matrix5(n, m):
    print(f'\nГереруємо матрицю планування для n = {n}, m = {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)
    make_table("X", x)
    make_table("X нормоване", [[round(x, 2) for x in i] for i in x_norm])
    make_table("Y", y)
    return x, y, x_norm


def find_coef(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 нормованими X:')
    else:
        print('\nКоефіцієнти рівняння регресії:')
    B = [round(i, 3) for i in B]
    print(B)
    print('\nРезультат рівняння зі знайденими коефіцієнтами:\n', np.dot(X,
B))
    return B


def cochran criterion(y, y aver, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp
```

```python
def cohren(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_criterion(x, y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n

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

    return ts


def kriteriy_fishera(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))])
    S_kv = s_kv(y, y_aver, n, m)
    S_kv_aver = sum(S_kv) / n

    return S_ad / S_kv_aver


def check(X, Y, B, n, m):
    print('\n\tПеревірка рівняння:')
    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 = cohren(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 = cochran criterion(Y, y aver, n, m)
    print(f'Gp = {Gp}')
    if Gp < G_kr:
        print(f'З ймовірністю {1 - q} дисперсії однорідні.')
    else:
        print("Необхідно збільшити кількість дослідів")
        m += 1
        main(n, m)
```

```
    ts = student_criterion(X[:, 1:], Y, y_aver, n, m)
    print('\nКритерій Стьюдента:\n', 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 {} статистично незначущі, тому ми виключаємо їх з
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(regression([X[j][i] for i in range(len(ts)) if ts[i] in
res], final_k))

    print(f'\nЗначення "y" з коефіцієнтами {final_k}')
    for i in y_new:
        print(" "*4, i)

    d = len(res)
    if d >= n:
        print('\nF4 <= 0')
        print('')
        return
    f4 = n - d

    F_p = kriteriy_fishera(Y, y_aver, y_new, n, m, d)
    fisher = partial(f.ppf, q=0.95)
    f_t = fisher(dfn=f4, dfd=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')


def main(n, m):
    X5, Y5, X5_norm = plan_matrix5(n, m)
    y5_aver = [round(sum(i) / len(i), 3) for i in Y5]
    B5 = find_coef(X5, y5_aver)
    check(X5_norm, Y5, B5, n, m)


if __name__ == '__main__':
    main(15, 5)
```

Результати роботи програми:

X										
1	-1	-3	-3	3	3	9	-9	1	9	9
1	6	-3	-3	-18	-18	9	54	36	9	9
1	-1	4	-3	-4	3	-12	12	1	16	9
1	6	4	-3	24	-18	-12	-72	36	16	9
1	-1	-3	7	3	-7	-21	21	1	9	49
1	6	-3	7	-18	42	-21	-126	36	9	49
1	-1	4	7	-4	-7	28	-28	1	16	49
1	6	4	7	24	42	28	168	36	16	49
1	6	0	1	0	6	0	0	36	0	1
1	-2	0	1	0	-2	0	0	4	0	1
1	2	4	1	8	2	4	8	4	16	1
1	2	-4	1	-8	2	-4	-8	4	16	1
1	2	0	7	0	14	0	0	4	0	49
1	2	0	-5	0	-10	0	0	4	0	25
1	2	0	1	0	2	0	0	4	0	1

X нормоване										
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	-1.22	0.0	0.0	-0.0	-0.0	0.0	-0.0	1.48	0.0	0.0
1.0	1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48	0.0	0.0
1.0	0.0	-1.22	0.0	-0.0	0.0	-0.0	-0.0	0.0	1.48	0.0
1.0	0.0	1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48	0.0
1.0	0.0	0.0	-1.22	0.0	-0.0	-0.0	-0.0	0.0	0.0	1.48
1.0	0.0	0.0	1.22	0.0	0.0	0.0	0.0	0.0	0.0	1.48
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Y				
203.0	205.0	202.0	201.0	200.0
204.0	199.0	200.0	203.0	199.0
200.0	204.0	201.0	198.0	205.0
202.0	205.0	198.0	205.0	199.0
202.0	198.0	202.0	200.0	205.0
200.0	198.0	201.0	199.0	205.0
199.0	203.0	204.0	203.0	202.0
199.0	203.0	202.0	201.0	201.0
201.0	205.0	201.0	201.0	203.0
200.0	204.0	205.0	205.0	204.0
201.0	201.0	205.0	202.0	203.0
200.0	205.0	204.0	203.0	203.0
205.0	198.0	199.0	203.0	203.0
205.0	200.0	203.0	200.0	205.0
199.0	200.0	199.0	205.0	204.0

```
Гереруємо матрицю планування для n = 15, m = 5

Коефіцієнти рівняння регресії:
[200.886, 0.22, -0.254, -0.089, 0.022, 0.003, 0.042, -0.006, -0.021, 0.079, 0.001]

Результат рівняння зі знайденими коефіцієнтами:
 [202.901 202.803 200.514 202.376 200.581 201.953 201.554 201.946 201.38
 200.268 201.704 203.144 200.71  201.682 201.16 ]

    Перевірка рівняння:

Середнє значення y: [203.0, 203.0, 201.0, 202.8, 199.8, 201.4, 201.2, 201.8, 201.4, 200.6, 201.2, 203.8, 202.6, 201.0, 200.0]
Дисперсія y: [1.2, 2.8, 4.4, 6.96, 3.36, 5.04, 4.56, 5.76, 9.84, 2.64, 3.76, 0.56, 1.44, 4.0, 2.0]

Перевірка за критерієм Кохрена
Gp = 0.16872427983539096
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [885.613, 0.887, 0.808, 2.209, 0.234, 0.117, 1.171, 0.82, 646.347, 647.644, 647.038]

Коефіцієнти [0.22, -0.254, 0.022, 0.003, 0.042, -0.006] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [200.886, -0.089, -0.021, 0.079, 0.001]
     201.03400000000002
     201.03400000000002
     201.03400000000002
     201.03400000000002
     200.85600000000002
     200.85600000000002
     200.85600000000002
     200.85600000000002
     200.85499927499998
     200.85499927499998
     201.002621775
     201.002621775
     200.995611225
     200.779341225
     200.886
```

```
Перевірка адекватності за критерієм Фішера
Fp = 3.2027843562551057
F_t = 1.9925919966294197
Математична модель не адекватна експериментальним даним

Process finished with exit code 0
```

Висновок:

Успішно проведено трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайдено рівняння регресії, яке буде адекватним для опису об'єкту. Написана відповідна програма. Результати успішного виконання надані у звіті.