Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

«Методи оптимізації та планування експерименту»

Лабораторна робота №5

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

Виконав:
студент групи ІО-91
Лазарєв М.О.
Варіант: 115
Перевірив Регіда П. Г.

Київ
2021 р.

Мета роботи: 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. Варіантивибираються 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

№варіанта	x1		x2		x3	
	min	max	min	max	min	max
115	-1	2	-9	6	-5	8

Код програми

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


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


x_range = ((-1,2), (-9, 6), (-5, 8))

x_max = sum([x[1] for x in x_range]) / 3
x_min = sum([x[0] for x in x_range]) / 3

y_max = 200 + int(x_max)
y_min = 200 + int(x_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 Matrix(n, m):
    print(f'\n Матриця 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] = -1 * dx[2] + x[9][3]

    x = add_sq_nums(x)

    print('\nX:\n', x)
    print('\nX нормоване:\n')
    for i in x_norm:
        print([round(x, 2) for x in i])
    print('\nY:\n', y)

    return x, y, x_norm


def Odds(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 Cohran(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


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(x, y, y_aver, n, m):
    S_kv = S_KV(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n


    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 Fisher(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 CH(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 = Cohran(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(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(regressia([X[j][i] for i in range(len(ts)) if ts[i] in
res], final_k))

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

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

    F_p = Fisher(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 = Matrix(n, m)

    y5_aver = [round(sum(i) / len(i), 3) for i in Y5]
    B5 = Odds(X5, y5_aver)

    CH(X5_norm, Y5, B5, n, m)



main(16, 4)
```

Результат програми

```
 Матриця планування для n = 16, m = 4

X:
 [[   1   -1   -9   -5    9    5   45  -45    1   81   25]
 [   1    2   -9   -5  -18  -10   45   90    4   81   25]
 [   1   -1    6   -5   -6    5  -30   30    1   36   25]
 [   1    2    6   -5   12  -10  -30  -60    4   36   25]
 [   1   -1   -9    8    9   -8  -72   72    1   81   64]
 [   1    2   -9    8  -18   16  -72 -144    4   81   64]
 [   1   -1    6    8   -6   -8   48  -48    1   36   64]
 [   1    2    6    8   12   16   48   96    4   36   64]
 [   1    1   -1    1   -1    1   -1   -1    1    1    1]
 [   1   -1   -1    1    1   -1   -1    1    1    1    1]
 [   1    0    8    1    0    0    8    0    0   64    1]
 [   1    0  -10    1    0    0  -10    0    0  100    1]
 [   1    0   -1    8    0    0   -8    0    0    1   64]
 [   1    0   -1   -6    0    0    6    0    0    1   36]
 [   1    0   -1    1    0    0   -1    0    0    1    1]
 [   1    0   -1    1    0    0   -1    0    0    1    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]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]

Y:
 [[201. 199. 201. 199.]
 [201. 199. 202. 203.]
 [202. 196. 199. 204.]
 [204. 197. 195. 195.]
 [195. 199. 203. 203.]
 [198. 199. 202. 196.]
 [199. 198. 204. 196.]
 [202. 205. 195. 205.]
 [197. 201. 205. 197.]
 [204. 197. 200. 198.]
 [200. 198. 198. 195.]
 [196. 201. 203. 205.]
 [201. 205. 204. 198.]
 [198. 198. 201. 203.]
 [203. 204. 199. 202.]
 [195. 203. 195. 197.]]
```

```
Коефіцієнти рівняння регресії:
[199.948, 0.143, -0.102, -0.003, -0.01, 0.043, 0.009, 0.013, -0.203, -0.009, 0.016]

Результат рівняння зі знайденими коефіцієнтами:
 [200.151 201.351 199.476 197.301 200.645 198.959 199.19  201.734 200.025
 199.659 198.641 199.991 200.969 200.689 200.045 200.045]

    Перевірка рівняння:

Середнє значення y: [200.0, 201.25, 200.25, 197.75, 200.0, 198.75, 199.25, 201.75, 200.0, 199.75, 197.75, 201.25, 202.0, 200.0, 202.0, 197.5]
Дисперсія y: [1.0, 2.188, 9.188, 13.688, 11.0, 4.688, 8.688, 16.688, 11.0, 7.188, 3.188, 11.188, 7.5, 4.5, 3.5, 10.75]

Перевірка за критерієм Кохрена
Gp = 0.1325054390116085
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [570.155, 0.054, 0.58, 0.344, 0.0, 0.446, 0.98, 1.337, 390.135, 389.937, 390.727]

Коефіцієнти [0.143, -0.102, -0.003, -0.01, 0.043, 0.009, 0.013] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [199.948, -0.203, -0.009, 0.016]
[199.752, 199.752, 199.752, 199.752, 199.752, 199.752, 199.752, 199.752, 199.648326325, 199.648326325, 199.934713975, 199.934713975, 199.9716196, 199.9716196, 199.948, 199.948]

Перевірка адекватності за критерієм Фішера
Fp = 1.3901229405792048
F_t = 1.960121060357092
Математична модель адекватна експериментальним даним
```

Висновок:

Провів трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайшов рівняння регресії, яке буде адекватним для опису об'єкту.