Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

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

Виконала:

студентка II курсу ФІОТ

групи ІВ-91

Сайко Сабріна

Перевірив:

Регіда П.Г.

Київ – 2021

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 на лабораторну роботу:

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.

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

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

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

4. Розрахувати коефіцієнти рівняння регресії і записати його.

5. Провести 3 статистичні перевірки.

Варіант завдання:

Варіант	X1		X2		X3	
124	-3	6	-8	2	-3	4

Довірча ймовірність дорівнює 0.95, а рівень значимості q = 0.05.

Роздруківка тексту програми:

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

```python
# Варіант - 124 (-3, 6, -8, 2, -3, 4)


x_range = ((-3, 6), (-8, 2), (-3, 4))

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 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 plan_matrix5(n, m):
    print("\nРівняння регресії з урахуванням квадратичних членів:")
    print(
        "ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 +
b123*x1*x2*x3 + b11x1^2 + b22x2^2 + b33x3^2\n")
    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)

    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 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 kriteriy_cochrana(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)


# оцінки коефіцієнтів
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 kriteriy_studenta(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  # статистична оцінка дисперсії
    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

    ### табличні значення

    start_time_kohren = time.perf_counter()
    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 = kriteriy_cochrana(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)

    start_time_student = time.perf_counter()

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

    ts = kriteriy_studenta(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}')
    print(y_new)

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

    start_time_fisher = time.perf_counter()
    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\nНеобхідно збільшити кількість дослідів')

def main():
    n = 17
    m = 5
    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()
```

Результати роботи програми:

```
Рівняння регресії з урахуванням квадратичних членів:
ŷ = b0 + b1*x1 + b2*x2 + b3*x3 + b12*x1*x2 + b13*x1*x3 + b23*x2*x3 + b123*x1*x2*x3 + b11x1^2 + b22x2^2 + b33x3^2


X:
 [[  1  -3  -8  -3  24   9  24 -72   9  64   9]
 [  1   6  -8  -3 -48 -18  24 144  36  64   9]
 [  1  -3   2  -3  -6   9  -6  18   9   4   9]
 [  1   6   2  -3  12 -18  -6 -36  36   4   9]
 [  1  -3  -8   4  24 -12 -32  96   9  64  16]
 [  1   6  -8   4 -48  24 -32 -192  36  64  16]
 [  1  -3   2   4  -6 -12   8 -24   9   4  16]
 [  1   6   2   4  12  24   8  48  36   4  16]
 [  1   6  -3   1 -18   6  -3 -18  36   9   1]
 [  1  -4  -3   1  12  -4  -3  12  16   9   1]
 [  1   1   3   1   3   1   3   3   1   9   1]
 [  1   1  -9   1  -9   1  -9  -9   1  81   1]
 [  1   1  -3   5  -3   5 -15 -15   1   9  25]
 [  1   1  -3  -3  -3  -3   9   9   1   9   9]
 [  1   1  -3   1  -3   1  -3  -3   1   9   1]
 [  1   1  -3   1  -3   1  -3  -3   1   9   1]
 [  1   1  -3   1  -3   1  -3  -3   1   9   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]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

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

Коефіцієнти рівняння регресії:
[199.313, -0.26, 0.103, 0.013, -0.008, -0.007, 0.006, -0.011, 0.032, 0.025, 0.015]

Результат рівняння зі знайденими коефіцієнтами:
 [201.934 198.847 200.534 199.697 200.093 202.109 201.423 198.759 199.131
 200.591 199.601 200.321 199.505 199.097 199.061 199.061 199.061]
```

```
Результат рівняння зі знайденими коефіцієнтами:
 [201.934 198.847 200.534 199.697 200.093 202.109 201.423 198.759 199.131
 200.591 199.601 200.321 199.505 199.097 199.061 199.061 199.061]

    Перевірка рівняння:

Середнє значення y: [202.4, 199.0, 201.0, 200.0, 200.0, 202.0, 201.4, 198.6, 199.0, 200.0, 199.2, 200.0, 199.8, 197.6, 200.8, 198.6, 199.4]
Дисперсія y: [3.44, 6.0, 8.4, 7.6, 4.0, 1.6, 8.24, 8.64, 11.2, 6.0, 4.96, 2.8, 6.56, 1.44, 6.16, 4.24, 5.84]

Перевірка за критерієм Кохрена
Gp = 0.11532125205930807
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [771.181, 0.904, 0.324, 0.697, 0.545, 0.817, 0.363, 1.634, 497.681, 497.748, 497.145]

Коефіцієнти [-0.26, 0.103, 0.013, -0.008, -0.007, 0.006, -0.011] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [199.313, 0.032, 0.025, 0.015]
[199.385, 199.385, 199.385, 199.385, 199.385, 199.385, 199.385, 199.385, 199.3602392, 199.3602392, 199.349905625, 199.349905625, 199.335143375, 199.335143375, 199.313, 199.313,

Перевірка адекватності за критерієм Фішера
Fp = 2.090791685427286
F_t = 1.866946302659468
Математична модель не адекватна експериментальним даним
Необхідно збільшити кількість дослідів
```

Висновок: У ході лабораторної роботи я змоделювала трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план.