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
з дисципліни « Методи оптимізації та планування » на тему
«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

Виконав:
студент II курсу ФІОТ
групи ІО – 91
Лаппо Максим
Номер залікової книжки: ІВ - 9119

Перевірив:
ас. 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. 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.

Варіант завдання:

116	-7	10	-4	6	-5	3

Код програми:

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


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


x_range = ((-7, 10), (-4, 6), (-5, 3))
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 plnMatr5(n, m):
    print(f'\nГенеруємо 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 findCoef(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 kritCochr(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 Cochr(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 kritStud(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 kritFish(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 audit(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 = Cochr(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 = kritCochr(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 = kritStud(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(regr([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 = kritFish(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 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 = plnMatr5(n, m)
    y5_aver = [round(sum(i) / len(i), 3) for i in Y5]
    B5 = findCoef(X5, y5_aver)
    audit(X5_norm, Y5, B5, n, m)


if __name__ == '__main__':
    Main(16, 4)
```

Результат роботи:

```
C:\TeoriyaUmovirnostey\venv\Scripts\python.exe C:/TeoriyaUmovirnostey/lb4.py

Генеруємо матрицю планування для n = 16, m = 4

X:
 [[   1   -7   -4   -5   28   35   20 -140   49   16   25]
 [   1   10   -4   -5  -40  -50   20  200  100   16   25]
 [   1   -7    6   -5  -42   35  -30  210   49   36   25]
 [   1   10    6   -5   60  -50  -30 -300  100   36   25]
 [   1   -7   -4    3   28  -21  -12   84   49   16    9]
 [   1   10   -4    3  -40   30  -12 -120  100   16    9]
 [   1   -7    6    3  -42  -21   18 -126   49   36    9]
 [   1   10    6    3   60   30   18  180  100   36    9]
 [   1   11    1    1   11   11    1   11  121    1    1]
 [   1   -9    1    1   -9   -9    1   -9   81    1    1]
 [   1    1    7    1    7    1    7    7    1   49    1]
 [   1    1   -5    1   -5    1   -5   -5    1   25    1]
 [   1    1    1    5    1    5    5    5    1    1   25]
 [   1    1    1   -3    1   -3   -3   -3    1    1    9]
 [   1    1    1    1    1    1    1    1    1    1    1]
 [   1    1    1    1    1    1    1    1    1    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:
 [[200. 205. 198. 204.]
 [195. 206. 204. 206.]
 [198. 196. 202. 199.]
 [198. 201. 202. 201.]
 [206. 203. 205. 201.]
 [197. 203. 206. 204.]
 [202. 202. 202. 196.]
 [201. 204. 201. 200.]
 [204. 202. 203. 202.]
 [202. 201. 197. 197.]
 [205. 198. 203. 198.]
 [202. 205. 205. 203.]
 [199. 197. 196. 204.]
 [198. 202. 206. 202.]
 [198. 200. 197. 195.]
 [204. 199. 203. 195.]]

Коефіцієнти рівняння регресії:
[200.436, 0.04, -0.369, -0.013, 0.01, -0.007, 0.005, 0.001, 0.007, 0.062, -0.046]

Результат рівняння зі знайденими коефіцієнтами:
 [201.877 203.169 198.827 200.969 202.965 202.761 199.755 201.761 201.406
 200.246 200.984 203.732 198.966 199.814 200.126 200.126]

    Перевірка рівняння:

Середнє значення y: [201.75, 202.75, 198.75, 200.5, 203.75, 202.5, 200.5, 201.5, 202.75, 199.25, 201.0, 203.75, 199.0, 202.0, 197.5, 200.25]
Дисперсія y: [8.188, 20.688, 4.688, 2.25, 3.688, 11.25, 6.75, 2.25, 0.688, 5.188, 9.5, 1.688, 9.5, 8.0, 3.25, 12.688]

Перевірка за критерієм Кохрена
Gp = 0.18763945072287624
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [612.846, 0.334, 1.173, 1.551, 0.571, 0.571, 0.19, 0.286, 420.077, 420.85, 419.795]

Коефіцієнти [0.04, -0.369, -0.013, 0.01, -0.007, 0.005, 0.001] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [200.436, 0.007, 0.062, -0.046]
[200.45900000000003, 200.45900000000003, 200.45900000000003, 200.45900000000003, 200.45900000000003, 200.45900000000003, 200.45900000000003, 200.459000

Перевірка адекватності за критерієм Фішера
Fp = 2.7308869735474524
F_t = 1.960121060357092
Математична модель не адекватна експериментальним даним

Process finished with exit code 0
```