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 №2 з

дисципліни

«Методи оптимізації та планування експерименту»

Виконав:

Гречаний Є.О.

Перевірив:

ас. Регіда П. Г.

Київ 2020

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

Мета: Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний
ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Завдання:

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.

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

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.

Програмний код

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


def plan_matrix5(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)

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


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):
    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 kriteriy_studenta(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 fisherr(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 start(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 = 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")

    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
    f4 = n - d

    F_p = fisherr(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')


# Варіант 203
x_range = ((-9, 9), (-9, 7), (-1, 7))
print("x діапазони за умовою варіанту", *x_range)

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)
X5, Y5, X5_norm = plan_matrix5(15, 3)

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

start(X5_norm, Y5, B5, 15, 3)
```

Результати

```
x діапазони (-9, 9) (-9, 7) (-1, 7)

X:
 [[   1   -9   -9   -1   81    9    9  -81   81   81    1]
 [   1    9   -9   -1  -81   -9    9   81   81   81    1]
 [   1   -9    7   -1  -63    9   -7   63   81   49    1]
 [   1    9    7   -1   63   -9   -7  -63   81   49    1]
 [   1   -9   -9    7   81  -63  -63  567   81   81   49]
 [   1    9   -9    7  -81   63  -63 -567   81   81   49]
 [   1   -9    7    7  -63  -63   49 -441   81   49   49]
 [   1    9    7    7   63   63   49  441   81   49   49]
 [   1   10   -1    1  -10   10   -1  -10  100    1    1]
 [   1  -10   -1    1   10  -10   -1   10  100    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    5    0    0   -5    0    0    1   25]
 [   1    0   -1   -3    0    0    3    0    0    1    9]
 [   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]
```

```
Y:
 [[205. 199. 197.]
 [194. 204. 199.]
 [203. 198. 203.]
 [204. 195. 200.]
 [200. 204. 206.]
 [198. 195. 207.]
 [204. 197. 207.]
 [198. 207. 207.]
 [200. 194. 201.]
 [197. 202. 201.]
 [197. 196. 197.]
 [200. 200. 206.]
 [196. 199. 194.]
 [196. 197. 195.]
 [205. 203. 194.]]

Коефіцієнти рівняння регресії:
[197.264, -0.081, -0.003, 0.319, 0.001, 0.006, 0.017, 0.002, 0.019, 0.026, -0.008]

Результат рівняння зі знайденими коефіцієнтами:
 [201.464 200.06  200.456 198.764 203.272 200.14  202.136 203.324 198.707
 200.267 199.351 200.035 198.603 196.315 197.587]
```

```
    Перевірка рівняння:

Середнє значення y: [200.333, 199.0, 201.333, 199.667, 203.333, 200.0, 202.667, 204.0, 198.333, 200.0, 196.667, 202.0, 196.333, 196.0, 200.667]
Дисперсія y: [11.556, 16.667, 9.556, 13.556, 4.222, 36.0, 17.556, 18.0, 9.556, 4.667, 0.222, 8.0, 4.222, 0.667, 22.889]

Перевірка за критерієм Кохрена
Gp = 0.15725552813664295
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
 [484.154, 0.481, 1.546, 1.248, 0.584, 0.135, 0.225, 0.673, 296.126, 296.192, 294.933]

Коефіцієнти [-0.081, -0.003, 0.319, 0.001, 0.006, 0.017, 0.002] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення 'y' з коефіцієнтами [197.264, 0.019, 0.026, -0.008]
[197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002, 197.30100000000002]

Перевірка адекватності за критерієм Фішера
Fp = 4.853488342861656
F_t = 2.125558740875511
Математична модель не адекватна експериментальним даним
```