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 наукових досліджень»
За темою:
«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

ВИКОНАВ:
Студент ІІ курсу ФІОТ
Групи ІВ-91
Гутов В.В.
Номер у списку - 8

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.

$$y_{1max} = 200 + x_{cpmax}$$
$$y_{1min} = 200 + x_{cpmin}$$

где $x_{cpmax} = \frac{x_{1max} + x_{2max} + x_{3max}}{3}$, $x_{cpmin} = \frac{x_{1min} + x_{2min} + x_{3min}}{3}$

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

№ варіанту	x_1		x_2		x_3	
	min	max	min	max	min	max
108	-5	7	-10	3	-7	1

Програмний код

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

# Гутов Віталій
# Варіант 108:
# x1_min = -5, x1_max = 7,
# x2_min = -10, x2_max = 3,
# x3_min = -7, x3_max = 1
# y_min = 200 + xc_min
# y_max = 200 + xc_max


def plan_matrix(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] = 1

    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] = 1 * dx[0] + x[9][1]
    x[9][1] = -1 * dx[0] + x[9][1]
    x[10][2] = 1 * dx[1] + x[9][2]
    x[11][2] = -1 * dx[1] + x[9][2]
    x[12][3] = 1 * dx[2] + x[9][3]
    x[13][3] = -1 * dx[2] + x[9][3]
    x = add_sq_nums(x)
    x_table = BeautifulTable()
    for i in range(n):
        x_table.rows.append([*x[i]])
    print('x matrix:')
    print(x_table)
    x_norm_table = BeautifulTable()
    for i in range(n):
        x_norm_table.rows.append([*x_norm[i]])
    print('Normalized x matrix:')
    print(x_norm_table)
    return x, y, x_norm


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 coef_finding(x, y, norm=False):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(x, y)
```

```
    b = skm.coef_
    if norm == 1:
        print('\nCoefficients of the regression equation with normalized x:')
    else:
        print('\nCoefficients of the regression equation:')
    b = [round(i, 3) for i in b]
    print(b)
    print('\nThe result of the equation with the found
coefficients:\n{}'.format(np.dot(x, b)))
    return b


def kohren_kr(y, y_aver, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    skv = s_kv(y, y_aver, n, m)
    gp = max(skv) / sum(skv)
    print('\nKohren check')
    return gp


def kohren(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_kr(x, y, y_aver, n, m):
    skv = s_kv(y, y_aver, n, m)
    skv_aver = sum(skv) / n
    sbs_tmp = (skv_aver / n / m) ** 0.5
    bs_tmp = bs(x, y_aver, n)
    ts = [round(abs(b) / sbs_tmp, 3) for b in bs_tmp]
    return ts


def fisher_kr(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))])
    skv = s_kv(y, y_aver, n, m)
    skv_aver = sum(skv) / n
    return S_ad / skv_aver


def check(x, y, b, n, m):
    print('\nCheck the equation:')
    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 = kohren(f1, f2)
    y_aver = [round(sum(i) / len(i), 3) for i in y]
    print('\nThe average value of y:', y_aver)
```

```
    disp = s_kv(y, y_aver, n, m)
    print('The variance y:', disp)
    gp = kohren_kr(y, y_aver, n, m)
    print(f'gp = {gp}')
    if gp < g_kr:
        print('With a probability of {} dispersions are
homogeneous.'.format(1 - q))
    else:
        print("It is necessary to increase the number of experiments")
        m += 1
        main(n, m)
    ts = student_kr(x[:, 1:], y, y_aver, n, m)
    print('\nStudent criterion:\n{}:'.format(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('\nThe coefficients {} are statistically insignificant, so we
exclude them from the equation.'.format([round(i, 3) for i in b if i not in
final_k]))
    y_new = []
    for j in range(n):
        y_new.append(round(regression([x[j][i] for i in range(len(ts)) if
ts[i] in res], final_k), 3))
    print('The value of y with coefficients {}: '.format(final_k))
    print(y_new)
    d = len(res)
    if d >= n:
        print('\nF4 <= 0')
        print('')
        return
    f4 = n - d
    f_p = fisher_kr(y, y_aver, y_new, n, m, d)
    fisher = partial(f.ppf, q=0.95)
    f_t = fisher(dfn=f4, dfd=f3)
    print('\nFisher adequacy check')
    print('fp =', f_p)
    print('ft =', f_t)
    if f_p < f_t:
        print('The mathematical model is adequate to the experimental data')
    else:
        print('The mathematical model is inadequate to the experimental
data')


def main(n, m):
    x, y, x_norm = plan_matrix(n, m)
    y5_aver = [round(sum(i) / len(i), 3) for i in y]
    b = coef_finding(x, y5_aver)
    check(x_norm, y, b, n, m)


x_range = ((-5, 7), (-10, 3), (-7, 1))
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_min = 200 + int(x_aver_min)
y_max = 200 + int(x_aver_max)


main(15, 3)
```

Результати роботи програми

```
"D:\4 semestr\MND\venv\Scripts\python.exe" "D:/4 semestr/MND/lab5/main_lab5.py"
x matrix:
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | -5 | -10 | -7 | 50  | 35  | 70  | -350 | 25 | 100 | 49 |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | 7  | -10 | -7 | -70 | -49 | 70  | 490  | 49 | 100 | 49 |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | -5 |  3  | -7 | -15 | 35  | -21 | 105  | 25 |  9  | 49 |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | 7  |  3  | -7 | 21  | -49 | -21 | -147 | 49 |  9  | 49 |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | -5 | -10 | 1  | 50  | -5  | -10 |  50  | 25 | 100 | 1  |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | 7  | -10 | 1  | -70 |  7  | -10 | -70  | 49 | 100 | 1  |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | -5 |  3  | 1  | -15 | -5  |  3  | -15  | 25 |  9  | 1  |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | 7  |  3  | 1  | 21  |  7  |  3  |  21  | 49 |  9  | 1  |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | 8  | -3  | 1  | -24 |  8  | -3  | -24  | 64 |  9  | 1  |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | -6 | -3  | 1  | 18  | -6  | -3  |  18  | 36 |  9  | 1  |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | 1  |  4  | 1  |  4  |  1  |  4  |  4   | 1  | 16  | 1  |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
| 1 | 1  | -10 | 1  | -10 |  1  | -10 | -10  | 1  | 100 | 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  |
+---+----+-----+----+-----+-----+-----+------+----+-----+----+
```

Normalized x matrix:

1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	-1.0	1.0	1.0	1.0
1.0	1.0	-1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	1.0	1.0
1.0	-1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	1.0	1.0
1.0	1.0	1.0	-1.0	1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0
1.0	-1.0	-1.0	1.0	1.0	-1.0	-1.0	1.0	1.0	1.0	1.0
1.0	1.0	-1.0	1.0	-1.0	1.0	-1.0	-1.0	1.0	1.0	1.0
1.0	-1.0	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.0	-1.215	0.0	0.0	-0.0	-0.0	0.0	-0.0	1.476	0.0	0.0
1.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476	0.0	0.0
1.0	0.0	-1.215	0.0	-0.0	0.0	-0.0	-0.0	0.0	1.476	0.0
1.0	0.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476	0.0
1.0	0.0	0.0	-1.215	0.0	-0.0	-0.0	-0.0	0.0	0.0	1.476
1.0	0.0	0.0	1.215	0.0	0.0	0.0	0.0	0.0	0.0	1.476
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
Coefficients of the regression equation:
[198.857, 0.094, -0.129, -0.309, 0.022, 0.005, -0.037, 0.001, -0.013, -0.008, -0.038]

The result of the equation with the found coefficients:
[197.188 195.784 198.631 199.567 199.7   197.816 196.775 198.479 198.344
 198.288 197.896 199.226 197.257 199.433 198.953]

Check the equation:

The average value of y: [197.333, 195.667, 198.667, 199.667, 198.667, 198.0, 196.0, 199.0, 197.667, 199.667, 198.0, 200.0, 197.0, 198.667, 199.333]
The variance y: [11.556, 5.556, 5.556, 4.222, 6.222, 4.667, 4.667, 8.667, 10.889, 16.222, 14.0, 2.0, 4.667, 10.889, 10.889]

Kohren check
gp = 0.134433864538531
With a probability of 0.95 dispersions are homogeneous.

Student criterion:
[468.821, 0.646, 0.961, 0.372, 0.999, 0.473, 1.104, 0.158, 342.085, 342.24, 341.697]:

The coefficients [0.094, -0.129, -0.309, 0.022, 0.005, -0.037, 0.001] are statistically insignificant, so we exclude them from the equation.
The value of y with coefficients [198.857, -0.013, -0.008, -0.038]:
[198.798, 198.798, 198.798, 198.798, 198.798, 198.798, 198.798, 198.798, 198.838, 198.838, 198.845, 198.845, 198.801, 198.801, 198.857]

Fisher adequacy check
fp = 0.9777291373320998
ft = 2.125558760875511
The mathematical model is adequate to the experimental data


Process finished with exit code 0
```

Висновок

Виконуючи дану лабораторну роботу, я провів трьохфакторний експеримент при пикористанні рівняння регресії з урахуванням квадратичних членів(центральний ортогональний композиційний план). Склав матрицю планування та знайшов коефіцієнти рівняння регресії, провів статистичні перевірки.

Результати роботи програми наведені вище. Під час виконання роботи проблем не виникло.