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» на тему

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІВ-92
Злочевський Нікіта Вікторович
Варіант: 209

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Хід роботи

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.

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

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.

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
209	-10	9	0	1	-3	4

Лістинг програми

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


x_range = ((-10, 9), (0, 1), (-3, 4))
avg_x_min = sum([x[0] for x in x_range]) / 3
avg_x_max = sum([x[1] for x in x_range]) / 3
y_min = 200 + int(avg_x_min)
y_max = 200 + int(avg_x_max)


def s_kv(y, y_average, n, m):
    res = []
    for i in range(n):
        s = sum([(y_average[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(round(s, 3))
    return res


def plan_matrix5(n, m):
    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 regression(x, b):
    y = sum([x[i] * b[i] for i in range(len(x))])
    return y


def find_coefficients(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 cochran_test(y, y_average, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    S_kv = s_kv(y, y_average, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp


def cohran(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_average, n):
    res = [sum(1 * y for y in y_average) / n]

    for i in range(len(x[0])):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_average)) / n
        res.append(b)
    return res


def student_test(x, y, y_average, n, m):
    S_kv = s_kv(y, y_average, n, m)
    s_kv_aver = sum(S_kv) / n

    s_Bs = (s_kv_aver / n / m) ** 0.5
    Bs = bs(x, y_average, n)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts
```

```python
def fisher_test(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('\nПеревірка 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 = cohran(f1, f2)

    y_average = [round(sum(i) / len(i), 3) for i in Y]
    print('\nСереднє значення y:', y_average)

    dispersion = s_kv(Y, y_average, n, m)
    print('Дисперсія y:', dispersion)

    Gp = cochran_test(Y, y_average, 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_test(X[:, 1:], Y, y_average, 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 = fisher_test(Y, y_average, 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 = plan_matrix5(n, m)
    y5_average = [round(sum(i) / len(i), 3) for i in y5]
    b5 = find_coefficients(x5, y5_average)
    check(x5_norm, y5, b5, n, m)


if __name__ == '__main__':
    main(15, 3)
```

Результат роботи програми

```
X:
 [[  1 -10   0  -3   0  30   0   0 100   0   9]
 [  1   9   0  -3   0 -27   0   0  81   0   9]
 [  1 -10   1  -3 -10  30  -3  30 100   1   9]
 [  1   9   1  -3   9 -27  -3 -27  81   1   9]
 [  1 -10   0   4   0 -40   0   0 100   0  16]
 [  1   9   0   4   0  36   0   0  81   0  16]
 [  1 -10   1   4 -10 -40   4 -40 100   1  16]
 [  1   9   1   4   9  36   4  36  81   1  16]
 [  1  11   0   1   0  11   0   0 121   0   1]
 [  1 -11   0   1   0 -11   0   0 121   0   1]
 [  1   0   0   1   0   0   0   0   0   0   1]
 [  1   0   0   1   0   0   0   0   0   0   1]
 [  1   0   0   5   0   0   0   0   0   0  25]
 [  1   0   0  -3   0   0   0   0   0   0   9]
 [  1   0   0   1   0   0   0   0   0   0   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:
 [[203. 197. 204.]
 [197. 198. 201.]
 [196. 201. 201.]
 [204. 197. 203.]
 [199. 196. 198.]
 [199. 200. 196.]
 [202. 196. 201.]
 [200. 197. 196.]
 [198. 201. 202.]
 [199. 196. 204.]
 [203. 204. 200.]
 [198. 198. 201.]
 [197. 204. 201.]
 [199. 201. 200.]
 [198. 201. 197.]]
```

```
Коефіцієнти рівняння регресії:
[200.238, -0.041, 0.066, -0.099, 0.049, 0.027, -0.139, -0.057, -0.006, 0.066, -0.016]

Результат рівняння зі знайденими коефіцієнтами:
 [201.011 198.807 199.36  201.336 198.316 199.703 199.682 197.668 199.243
 199.551 200.123 200.123 199.343 200.391 200.123]

Перевірка рівняння:

Середнє значення y: [201.333, 198.667, 199.333, 201.333, 197.667, 198.333, 199.667, 197.667, 200.333, 199.667, 202.333, 199.0, 200.667, 200.0, 198.667]
Дисперсія y: [9.556, 2.889, 5.556, 9.556, 1.556, 2.889, 6.889, 2.889, 2.889, 10.889, 2.889, 2.0, 8.222, 0.667, 2.889]

Перевірка за критерієм Кохрена
Gp = 0.15076497057805466
З ймовірністю 0.95 дисперсії однорідні.
```

```
Критерій Стьюдента:
 [610.331, 0.573, 0.418, 1.659, 0.408, 0.136, 0.136, 1.494, 445.212, 445.613, 445.413]

Коефіцієнти [-0.041, -0.099, 0.049, 0.027, -0.139, -0.057] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [200.238, -0.006, 0.066, -0.016]
[200.282, 200.282, 200.282, 200.282, 200.282, 200.282, 200.282, 200.282, 200.22914265, 200.22914265, 200.33543085, 200.33543085, 200.2143804, 200.2143804, 200.238]

Перевірка адекватності за критерієм Фішера
Fp = 1.8346427205024045
F_t = 2.125558760875511
Математична модель адекватна експериментальним даним
```

Висновок:

На цій лабораторній роботі ми провели трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайшли рівняння регресії, яке адекватне для опису об'єкту. Закріпили отримані знання їх практичним використанням при написанні програми у середовищі Pycharm на мові python, що реалізує завдання лабораторної роботи.