Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №5

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

Виконав:
студент групи ІО-93
Варченко Є. В.
Номер у списку групи – 3

Перевірив:
ас. Регіда П. Г.

Київ – 2021

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

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.

Варіант

№варіанта	x_1		x_2		x_3	
	min	max	min	max	min	max
303	0	1	-3	1	-3	9

Код програми

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

x_values = (
    (0, 1),
    (-3, 1),
    (-3, 9)
)

x_average_max = sum([x[1] for x in x_values]) / 3
x_average_min = sum([x[0] for x in x_values]) / 3

y_max = 200 + int(x_average_max)
y_min = 200 + int(x_average_min)

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 planning_matrix_generate(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_values[j - 1][0]
            else:
                x[i][j] = x_values[j - 1][1]

    for i in range(8, len(x)):
        for j in range(1, 3):
            x[i][j] = (x_values[j - 1][0] + x_values[j - 1][1]) / 2

    dx = [x_values[i][1] - (x_values[i][0] + x_values[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_coefficient(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 kohren_criterion(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 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_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
```

```python
def student_criterion(x, y, y_average, n, m):
    S_kv = s_kv(y, y_average, n, m)
    s_kv_average = sum(S_kv) / n

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

    return ts


def fisher_criterion(y, y_average, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - y_average[i]) ** 2 for i in range(len(y))])
    S_kv = s_kv(y, y_average, n, m)
    S_kv_average = sum(S_kv) / n

    return S_ad / S_kv_average


def check(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 = kohren(f1, f2)

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

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

```
    Gp = kohren_criterion(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_criterion(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_criterion(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 = planning_matrix_generate(n, m)

    y5_average = [round(sum(i) / len(i), 3) for i in Y5]
    B5 = find_coefficient(X5, y5_average)

    check(X5_norm, Y5, B5, n, m)


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

Результати роботи програми

```
X:
 [[  1   0  -3  -3   0   0   9   0   0   9   9]
 [  1   1  -3  -3  -3  -3   9   9   1   9   9]
 [  1   0   1  -3   0   0  -3   0   0   1   9]
 [  1   1   1  -3   1  -3  -3  -3   1   1   9]
 [  1   0  -3   9   0   0 -27   0   0   9  81]
 [  1   1  -3   9  -3   9 -27 -27   1   9  81]
 [  1   0   1   9   0   0   9   0   0   1  81]
 [  1   1   1   9   1   9   9   9   1   1  81]
 [  1   0  -1   1   0   0  -1   0   0   1   1]
 [  1   0  -1   1   0   0  -1   0   0   1   1]
 [  1   0   1   1   0   0   1   0   0   1   1]
 [  1   0  -3   1   0   0  -3   0   0   9   1]
 [  1   0  -1   8   0   0  -8   0   0   1  64]
 [  1   0  -1  -6   0   0   6   0   0   1  36]
 [  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:
 [[201. 198. 198.]
 [199. 201. 201.]
 [199. 203. 199.]
 [198. 203. 202.]
 [198. 200. 201.]
 [203. 200. 200.]
 [199. 201. 201.]
 [199. 201. 203.]
 [201. 203. 199.]
 [199. 200. 203.]
 [202. 198. 198.]
 [200. 199. 201.]
 [200. 199. 202.]
 [198. 202. 203.]
 [199. 198. 200.]]

Коефіцієнти рівняння регресії:
[200.212, 0.448, -0.202, -0.052, 0.0, 0.009, -0.006, -0.008, 0.448, -0.163, 0.01]

Результат рівняння зі знайденими коефіцієнтами:
 [199.543 200.34  200.111 201.004 199.855 201.048 200.135 201.04  200.215
 200.215 199.799 199.327 200.523 200.887 200.215]

    Перевірка рівняння:

Середнє значення y: [199.0, 200.333, 200.333, 201.0, 199.667, 201.0, 200.333, 201.0, 201.0, 200.667, 199.333, 200.0, 200.333, 201.0, 199.0]
Дисперсія y: [2.0, 0.889, 3.556, 4.667, 1.556, 2.0, 0.889, 2.667, 2.667, 2.889, 3.556, 0.667, 1.556, 4.667, 0.667]

Перевірка за критерієм Кохрена
Gp = 0.13375175536640585
З ймовірністю 0.95 дисперсії однорідні.



 Критерій Стьюдента:
  [880.828, 1.054, 1.019, 0.629, 0.391, 0.0, 0.391, 0.0, 643.796, 642.785, 643.651]

 Коефіцієнти [-0.202, -0.052, 0.0, 0.009, -0.006, -0.008] статистично незначущі, тому ми виключаємо їх з рівняння.

 Значення "y" з коефіцієнтами [200.212, 0.448, -0.163, 0.01]
 [200.50699999999998, 200.50699999999998, 200.50699999999998, 200.50699999999998, 200.50699999999998, 200.506999999

 Перевірка адекватності за критерієм Фішера
 Fp = 0.7435123055385675
 F_t = 2.125558760875511
 Математична модель адекватна експериментальним даним

 Process finished with exit code 0
```

Висновки

- Ознайомилися з темою роботи.
- Були здобуті необхідні навички для виконання завдань.
- Розроблено програму, яка виконує поставлену задачу.
- Вище приведені результати свідчать про успішне виконання умов завдань.
- Основну мету лабораторної роботи було досягнуто.