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 експерименту»

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

Виконав:

студент групи ІО-82

Кузенний Павло Вадимович

Залікова книжка № IO-8209

Варіант: 208

Перевірив:

Регіда П.Г.

Київ 2020

# 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.

4. Розрахувати коефіцієнти рівняння регресії і записати його.

5. Провести 3 статистичні перевірки.

**Варіант:**



**Код програми:**

import numpy as np  
from prettytable import PrettyTable  
from scipy.stats import f, t  
from functools import partial  
import sklearn.linear\_model as lm  
  
# Variant №208  
  
x1\_min = -5  
x1\_max = 6  
x2\_min = -7  
x2\_max = 9  
x3\_min = -5  
x3\_max = 3  
  
m = 3  
N = 15  
  
x\_max\_av = (x1\_max + x2\_max + x3\_max) / 3  
x\_min\_av = (x1\_min + x2\_min + x3\_min) / 3  
  
y\_max = int(200 + x\_max\_av)  
y\_min = int(200 + x\_min\_av)  
  
x\_matrix = np.array([  
 [x1\_min, x2\_min, x3\_min],  
 [x1\_min, x2\_max, x3\_max],  
 [x1\_max, x2\_min, x3\_max],  
 [x1\_max, x2\_max, x3\_min]  
])  
  
x1\_0 = (x1\_min+x1\_max)/2  
x2\_0 = (x2\_min+x2\_max)/2  
x3\_0 = (x3\_min+x3\_max)/2  
  
while True:  
 matrix\_plan = np.random.randint(y\_min, y\_max, size=(N, m))  
  
 x0\_factor = np.array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])  
 x1\_factor = np.array([-1, -1, -1, -1, 1, 1, 1, 1, -1.215, 1.215, 0, 0, 0, 0, 0])  
 x2\_factor = np.array([-1, -1, 1, 1, -1, -1, 1, 1, 0, 0, -1.215, 1.215, 0, 0, 0])  
 x3\_factor = np.array([-1, 1, -1, 1, -1, 1, -1, 1, 0, 0, 0, 0, -1.215, 1.215, 0])  
 x1x2\_factor = x1\_factor \* x2\_factor  
 x1x3\_factor = x1\_factor \* x3\_factor  
 x2x3\_factor = x2\_factor \* x3\_factor  
 x1x2x3\_factors = x1\_factor \* x2\_factor \* x3\_factor  
 x\_1\_2\_factor = x1\_factor \* x1\_factor  
 x\_2\_2\_factor = x2\_factor \* x2\_factor  
 x\_3\_2\_factor = x3\_factor \* x3\_factor  
  
 x0 = np.array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])  
 x1 = np.array([x1\_min, x1\_min, x1\_min, x1\_min, x1\_max, x1\_max, x1\_max, x1\_max, x1\_min \* 1.215, round(x1\_max \* 1.215), x1\_0, x1\_0, x1\_0, x1\_0, x1\_0])  
 x2 = np.array([x2\_min, x2\_min, x2\_max, x2\_max, x2\_min, x2\_min, x2\_max, x2\_max, x2\_0, x2\_0, x2\_min \* 1.215, x2\_max \* 1.215, x2\_0, x2\_0, x2\_0])  
 x3 = np.array([x3\_min, x3\_max, x3\_min, x3\_max, x3\_min, x3\_max, x3\_min, x3\_max, x3\_0, x3\_0, x3\_0, x3\_0, x3\_min \* 1.215, round(x3\_max \* 1.215), x3\_0])  
 x1x2 = x1 \* x2  
 x1x3 = x1 \* x3  
 x2x3 = x2 \* x3  
 x1x2x3 = x1 \* x2 \* x3  
 x\_1\_2 = x1 \* x1  
 x\_2\_2 = x2 \* x2  
 x\_3\_2 = x3 \* x3  
  
 factor\_matrix = np.zeros((N, 11))  
 factor\_matrix[:,0] = x0\_factor  
 factor\_matrix[:,1] = x1\_factor  
 factor\_matrix[:, 2] = x2\_factor  
 factor\_matrix[:, 3] = x3\_factor  
 factor\_matrix[:, 4] = x1x2\_factor  
 factor\_matrix[:, 5] = x1x3\_factor  
 factor\_matrix[:, 6] = x2x3\_factor  
 factor\_matrix[:, 7] = x1x2x3\_factors  
 factor\_matrix[:, 8] = x1 \* x1  
 factor\_matrix[:, 9] = x2 \* x2  
 factor\_matrix[:, 10] = x3 \* x3  
  
 x\_norm = np.zeros((N, 11))  
  
 x\_norm[:, 0] = x0  
 x\_norm[:, 1] = x1  
 x\_norm[:, 2] = x2  
 x\_norm[:, 3] = x3  
 x\_norm[:, 4] = x1x2  
 x\_norm[:, 5] = x1x3  
 x\_norm[:, 6] = x2x3  
 x\_norm[:, 7] = x1x2x3  
 x\_norm[:, 8] = x1x2x3  
 x\_norm[:, 9] = x1x2x3  
 x\_norm[:, 10] = x1x2x3  
  
 y\_average = np.zeros((N, 1))  
 for i in range(N):  
 y\_average[i, 0] = round((sum(matrix\_plan[i, :] / m)), 3)  
  
 d\_list = np.zeros((N, 1))  
 np.array(d\_list)  
 for i in range(N):  
 d\_list[i][0] = (  
 round(((matrix\_plan[i][0] - y\_average[i][0]) \*\* 2 + (matrix\_plan[i][1] - y\_average[i][0]) \*\* 2 + (  
 matrix\_plan[i][2] - y\_average[i][0]) \*\* 2) / 3, 3))  
 d\_sum = sum(d\_list)  
  
 my\_table = np.hstack((x\_norm, matrix\_plan, y\_average, d\_list))  
  
 table = PrettyTable()  
 table.field\_names = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3", "X1^2", "X2^2", "X3^2", "Y1", "Y2", "Y3",  
 "Y", "S^2"]  
 for i in range(len(my\_table)):  
 table.add\_row(my\_table[i])  
  
 print(f'\nМатриця планування при N = {N}, m = {m}')  
  
 print(table)  
 y\_average = y\_average[:, 0]  
 skm = lm.LinearRegression(fit\_intercept=False) # знаходимо коефіцієнти рівняння регресії  
 skm.fit(x\_norm, y\_average)  
 B = skm.coef\_  
  
 print('\nРівняння регресії:')  
  
 print("\ny = {} + {}\*x1 + {}\*x2 + {}\*x3 + {}\*x1x2 + {}\*x1x3 + {}\*x2x3 + {}\*x1x2x3 + {}\*x1^2 + {}\*x2^2 + {}\*x3^2 \n".format(  
 round(float(B[0]), 3),  
 round(float(B[1]), 3),  
 round(float(B[2]), 3),  
 round(float(B[3]), 3),  
 round(float(B[4]), 3),  
 round(float(B[5]), 3),  
 round(float(B[6]), 3),  
 round(float(B[7]), 3),  
 round(float(B[8]), 3),  
 round(float(B[9]), 3),  
 round(float(B[10]), 3)))  
  
 print('\nКоефіцієнти рівняння регресії:')  
 B = [round(i, 3) for i in B]  
 print(B, "\n")  
 y\_list = np.zeros((15, 2))  
 y\_list[:, 0] = np.dot(x\_norm, B)  
 y\_list[:, 1] = y\_average  
 my\_table = y\_list  
 table = PrettyTable()  
 table.field\_names = ["Фактори", "Y середнє"]  
 for i in range(len(my\_table)):  
 table.add\_row(my\_table[i])  
 print(table)  
  
 print('\nПроведемо перевірку рівняння')  
  
 f1 = m - 1  
 f2 = N  
 f3 = f1 \* f2  
 q = 0.05  
  
 student = partial(t.ppf, q=1 - q)  
 t\_student = student(df=f3)  
 q1 = q / f1  
 fisher\_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) \* f2)  
 G\_kr = fisher\_value / (fisher\_value + f1 - 1)  
  
 y\_aver = [round(sum(i) / len(i), 3) for i in matrix\_plan]  
 print('\nСереднє значення y:', y\_aver)  
  
 res = []  
 for i in range(N):  
 s = sum([(y\_aver[i] - matrix\_plan[i][j]) \*\* 2 for j in range(m)]) / m  
 res.append(round(s, 3))  
 disp = res  
 print('Дисперсія y:', disp)  
  
 S\_kv = res  
 Gp = max(S\_kv) / sum(S\_kv)  
 print('\nПеревірка за критерієм Кохрена')  
  
 print(f'Gp = {Gp}')  
 if Gp < G\_kr:  
 print(f'З ймовірністю {1-q} дисперсії однорідні.')  
 break  
 else:  
 print("Необхідно збільшити кількість дослідів")  
 m += 1  
  
  
X = factor\_matrix[:, 1:]  
s\_kv\_aver = sum(S\_kv) / N  
  
  
s\_Bs = (s\_kv\_aver / N / m) \*\* 0.5  
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)  
Bs = res  
ts = [round(abs(B) / s\_Bs, 3) for B in Bs]  
  
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(  
 [i for i in B if i not in final\_k]))  
  
y\_new = []  
for j in range(N):  
 x = [x\_norm[j][i] for i in range(len(ts)) if ts[i] in res]  
 b = final\_k  
 y\_new.append(sum([x[i] \* b[i] for i in range(len(x))]))  
print(f'\nЗначення "y" з коефіцієнтами {final\_k}')  
d = len(res)  
print("Кількість значимих коефіцієнтів d =", d)  
if d >= N:  
 print('\nF4 <= 0')  
 print('')  
f4 = N - d  
  
S\_ad = m / (N - d) \* sum([(y\_new[i] - y\_aver[i]) \*\* 2 for i in range(len(matrix\_plan))])  
S\_kv\_aver = sum(S\_kv) / N  
F\_p = S\_ad / S\_kv\_aver  
  
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('Ft =', f\_t)  
if F\_p < f\_t:  
 print('Отримана математична модель адекватна експериментальним даним')  
else:  
 print('Рівняння регресії неадекватно оригіналу')

**Результати виконання програми:**

Матриця планування при N = 15, m = 3

+-----+--------+--------+--------+---------+---------+---------+---------+---------+---------+---------+-------+-------+-------+---------+--------+

| X0 | X1 | X2 | X3 | X1X2 | X1X3 | X2X3 | X1X2X3 | X1^2 | X2^2 | X3^2 | Y1 | Y2 | Y3 | Y | S^2 |

+-----+--------+--------+--------+---------+---------+---------+---------+---------+---------+---------+-------+-------+-------+---------+--------+

| 1.0 | -5.0 | -7.0 | -5.0 | 35.0 | 25.0 | 35.0 | -175.0 | -175.0 | -175.0 | -175.0 | 195.0 | 198.0 | 204.0 | 199.0 | 14.0 |

| 1.0 | -5.0 | -7.0 | 3.0 | 35.0 | -15.0 | -21.0 | 105.0 | 105.0 | 105.0 | 105.0 | 203.0 | 195.0 | 199.0 | 199.0 | 10.667 |

| 1.0 | -5.0 | 9.0 | -5.0 | -45.0 | 25.0 | -45.0 | 225.0 | 225.0 | 225.0 | 225.0 | 197.0 | 194.0 | 203.0 | 198.0 | 14.0 |

| 1.0 | -5.0 | 9.0 | 3.0 | -45.0 | -15.0 | 27.0 | -135.0 | -135.0 | -135.0 | -135.0 | 203.0 | 204.0 | 199.0 | 202.0 | 4.667 |

| 1.0 | 6.0 | -7.0 | -5.0 | -42.0 | -30.0 | 35.0 | 210.0 | 210.0 | 210.0 | 210.0 | 204.0 | 202.0 | 196.0 | 200.667 | 11.556 |

| 1.0 | 6.0 | -7.0 | 3.0 | -42.0 | 18.0 | -21.0 | -126.0 | -126.0 | -126.0 | -126.0 | 201.0 | 200.0 | 196.0 | 199.0 | 4.667 |

| 1.0 | 6.0 | 9.0 | -5.0 | 54.0 | -30.0 | -45.0 | -270.0 | -270.0 | -270.0 | -270.0 | 198.0 | 201.0 | 202.0 | 200.333 | 2.889 |

| 1.0 | 6.0 | 9.0 | 3.0 | 54.0 | 18.0 | 27.0 | 162.0 | 162.0 | 162.0 | 162.0 | 194.0 | 202.0 | 205.0 | 200.333 | 21.556 |

| 1.0 | -6.075 | 1.0 | -1.0 | -6.075 | 6.075 | -1.0 | 6.075 | 6.075 | 6.075 | 6.075 | 205.0 | 205.0 | 201.0 | 203.667 | 3.556 |

| 1.0 | 7.0 | 1.0 | -1.0 | 7.0 | -7.0 | -1.0 | -7.0 | -7.0 | -7.0 | -7.0 | 195.0 | 195.0 | 204.0 | 198.0 | 18.0 |

| 1.0 | 0.5 | -8.505 | -1.0 | -4.2525 | -0.5 | 8.505 | 4.2525 | 4.2525 | 4.2525 | 4.2525 | 205.0 | 194.0 | 196.0 | 198.333 | 22.889 |

| 1.0 | 0.5 | 10.935 | -1.0 | 5.4675 | -0.5 | -10.935 | -5.4675 | -5.4675 | -5.4675 | -5.4675 | 194.0 | 199.0 | 202.0 | 198.333 | 10.889 |

| 1.0 | 0.5 | 1.0 | -6.075 | 0.5 | -3.0375 | -6.075 | -3.0375 | -3.0375 | -3.0375 | -3.0375 | 204.0 | 197.0 | 195.0 | 198.667 | 14.889 |

| 1.0 | 0.5 | 1.0 | 4.0 | 0.5 | 2.0 | 4.0 | 2.0 | 2.0 | 2.0 | 2.0 | 196.0 | 204.0 | 200.0 | 200.0 | 10.667 |

| 1.0 | 0.5 | 1.0 | -1.0 | 0.5 | -0.5 | -1.0 | -0.5 | -0.5 | -0.5 | -0.5 | 198.0 | 197.0 | 194.0 | 196.333 | 2.889 |

+-----+--------+--------+--------+---------+---------+---------+---------+---------+---------+---------+-------+-------+-------+---------+--------+

Рівняння регресії:

y = 199.529 + -0.102\*x1 + 0.058\*x2 + 0.083\*x3 + -0.005\*x1x2 + -0.031\*x1x3 + 0.023\*x2x3 + -0.0\*x1x2x3 + -0.0\*x1^2 + -0.0\*x2^2 + -0.0\*x3^2

Коефіцієнти рівняння регресії:

[199.529, -0.102, 0.058, 0.083, -0.005, -0.031, 0.023, -0.0, -0.0, -0.0, -0.0]

+--------------------+-----------+

| Фактори | Y середнє |

+--------------------+-----------+

| 199.07299999999998 | 199.0 |

| 199.68899999999996 | 199.0 |

| 198.56099999999998 | 198.0 |

| 202.12099999999998 | 202.0 |

| 200.041 | 200.667 |

| 197.929 | 199.0 |

| 198.64899999999997 | 200.333 |

| 199.481 | 200.333 |

| 199.94269999999997 | 203.667 |

| 198.94899999999998 | 198.0 |

| 199.1340875 | 198.333 |

| 199.7658875 | 198.333 |

| 198.9837125 | 198.667 |

| 199.8955 | 200.0 |

| 199.44299999999998 | 196.333 |

+--------------------+-----------+

Проведемо перевірку рівняння

Середнє значення y: [199.0, 199.0, 198.0, 202.0, 200.667, 199.0, 200.333, 200.333, 203.667, 198.0, 198.333, 198.333, 198.667, 200.0, 196.333]

Дисперсія y: [14.0, 10.667, 14.0, 4.667, 11.556, 4.667, 2.889, 21.556, 3.556, 18.0, 22.889, 10.889, 14.889, 10.667, 2.889]

Перевірка за критерієм Кохрена

Gp = 0.13642188328833416

З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:

[400.039, 0.609, 0.401, 0.529, 0.134, 0.758, 0.758, 0.312, 8855.911, 19121.41, 5172.15]

Коефіцієнти [-0.102, 0.058, 0.083, -0.005, -0.031, 0.023] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [199.529, -0.0, -0.0, -0.0]

Кількість значимих коефіцієнтів d = 4

Перевірка адекватності за критерієм Фішера

Fp = 1.1062821909079532

Ft = 2.125558760875511

Отримана математична модель адекватна експериментальним даним

**Висновок:** Під час виконання роботи я навчився проводити повний трьохфакторний експеримент з урахуванням квадратичних членів та перевірив, чи рівняння регресії адекватне об’єкту. Закріпив знання використання статистичних перевірок за критеріями Кохрена, Стьюдента та Фішера. Отримані результати збігаються з очікуваними.