Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №6

“ ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З КВАДРАТИЧНИМИ ЧЛЕНАМИ”

Виконав:

студент групи ІВ-83

Кочерук Д.А.

Перевірив:

ас. Регіда П.Г.

Київ 2020 р.

**Мета:** Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

Номер у списку: 14

Варіант: 314





1. Лістинг програми

from copy import deepcopy  
from datetime import datetime  
from math import sqrt  
from random import random  
import numpy as np  
from prettytable import PrettyTable  
  
x1\_min = -25  
x1\_max = 75  
x2\_min = 5  
x2\_max = 40  
x3\_min = 15  
x3\_max = 25  
koefs = [5.5, 6.4, 0.6, 2.7, 1.9, 0.4, 0.7, 1.8, 0.2, 6.0, 4.8]  
  
x\_average\_max = (x1\_max + x2\_max + x3\_max) / 3  
x\_average\_min = (x1\_min + x2\_min + x3\_min) / 3  
y\_max = 200 + x\_average\_max  
y\_min = 200 + x\_average\_min  
  
print("-" \* 100)  
print('Значення за варіантом:')  
print('x1\_min = ',x1\_min)  
print('x1\_max = ',x1\_max)  
print('x2\_min = ',x2\_min)  
print('x2\_max = ',x2\_max)  
print('x3\_min = ',x3\_min)  
print('x3\_max = ',x3\_max)  
print('y\_min = ',y\_min)  
print('y\_max = ',y\_max)  
print('f(x1,x2,x3) = 6,8+1,7\*x1+5,7\*x2+9,9\*x3+8,9\*x1\*x1+0,5\*x2\*x2+6,1\*x3\*x3+9,9\*x1\*x2+0,7\*x1\*x3+3,2\*x2\*x3+8,4\*x1\*x2\*x3')  
print("-" \* 100)  
  
def replace\_column(list\_: list, column, list\_replace):  
 list\_ = deepcopy(list\_)  
 for i in range(len(list\_)):  
 list\_[i][column] = list\_replace[i]  
 return list\_  
  
  
def append\_to\_list\_x(x: list, variant: int):  
 if variant == 1:  
 for i in range(len(x)):  
 x[i].append(x[i][1] \* x[i][2])  
 x[i].append(x[i][1] \* x[i][3])  
 x[i].append(x[i][2] \* x[i][3])  
 x[i].append(x[i][1] \* x[i][2] \* x[i][3])  
 if variant == 2:  
 for i in range(len(x)):  
 x[i].append(x[i][1] \* x[i][2])  
 x[i].append(x[i][1] \* x[i][3])  
 x[i].append(x[i][2] \* x[i][3])  
 x[i].append(x[i][1] \* x[i][2] \* x[i][3])  
 x[i].append(x[i][1] \* x[i][1])  
 x[i].append(x[i][2] \* x[i][2])  
 x[i].append(x[i][3] \* x[i][3])  
 for i in range(len(x)):  
 for j in range(len(x[i])):  
 if round(x[i][j], 3) == 0:  
 x[i][j] = 0  
 x[i][j] = round(x[i][j], 3)  
  
  
def get\_value(table: dict, key: int):  
 value = table.get(key)  
 if value is not None:  
 return value  
 for i in table:  
 if type(i) == range and key in i:  
 return table.get(i)  
  
  
def main(m, n):  
 if n == 14:  
 const\_l = 1.73  
 print(  
 'ŷ = b0 + b1 \* x1 + b2 \* x2 + b3 \* x3 + b12 \* x1 \* x2 + b13 \* x1 \* x3 + b23 \* x2 \* x3 + b123 \* x1 \* x2 \* '  
 'x3 + b11 \* x1 \* x1 + b22 \* x2 \* x2 + b33 \* x3 \* x3')  
 norm\_x = [  
 [+1, -1, -1, -1],  
 [+1, -1, +1, +1],  
 [+1, +1, -1, +1],  
 [+1, +1, +1, -1],  
 [+1, -1, -1, +1],  
 [+1, -1, +1, -1],  
 [+1, +1, -1, -1],  
 [+1, +1, +1, +1],  
 [+1, -const\_l, 0, 0],  
 [+1, const\_l, 0, 0],  
 [+1, 0, -const\_l, 0],  
 [+1, 0, const\_l, 0],  
 [+1, 0, 0, -const\_l],  
 [+1, 0, 0, const\_l],  
 ]  
  
 delta\_x1 = (x1\_max - x1\_min) / 2  
 delta\_x2 = (x2\_max - x2\_min) / 2  
 delta\_x3 = (x2\_max - x3\_min) / 2  
 x01 = (x1\_min + x1\_max) / 2  
 x02 = (x2\_min + x2\_max) / 2  
 x03 = (x3\_min + x3\_max) / 2  
  
 x = [  
 [1, x1\_min, x2\_min, x3\_min],  
 [1, x1\_min, x2\_max, x3\_max],  
 [1, x1\_max, x2\_min, x3\_max],  
 [1, x1\_max, x2\_max, x3\_min],  
 [1, x1\_min, x2\_min, x3\_max],  
 [1, x1\_min, x2\_max, x3\_min],  
 [1, x1\_max, x2\_min, x3\_min],  
 [1, x1\_max, x2\_max, x3\_max],  
 [1, -const\_l \* delta\_x1 + x01, x02, x03],  
 [1, const\_l \* delta\_x1 + x01, x02, x03],  
 [1, x01, -const\_l \* delta\_x2 + x02, x03],  
 [1, x01, const\_l \* delta\_x2 + x02, x03],  
 [1, x01, x02, -const\_l \* delta\_x3 + x03],  
 [1, x01, x02, const\_l \* delta\_x3 + x03],  
 ]  
  
 append\_to\_list\_x(norm\_x, variant=2)  
 append\_to\_list\_x(x, variant=2)  
  
 if n == 8:  
 print(  
 'ŷ = b0 + b1 \* x1 + b2 \* x2 + b3 \* x3 + b12 \* x1 \* x2 + b13 \* x1 \* x3 + b23 \* x2 \* x3 + b123 \* x1 \* x2 \* x3'  
 )  
 norm\_x = [  
 [+1, -1, -1, -1],  
 [+1, -1, +1, +1],  
 [+1, +1, -1, +1],  
 [+1, +1, +1, -1],  
 [+1, -1, -1, +1],  
 [+1, -1, +1, -1],  
 [+1, +1, -1, -1],  
 [+1, +1, +1, +1]  
 ]  
  
 x = [  
 [1, x1\_min, x2\_min, x3\_min],  
 [1, x1\_min, x2\_max, x3\_max],  
 [1, x1\_max, x2\_min, x3\_max],  
 [1, x1\_max, x2\_max, x3\_min],  
 [1, x1\_min, x2\_min, x3\_max],  
 [1, x1\_min, x2\_max, x3\_min],  
 [1, x1\_max, x2\_min, x3\_min],  
 [1, x1\_max, x2\_max, x3\_max]  
 ]  
  
 append\_to\_list\_x(norm\_x, variant=1)  
 append\_to\_list\_x(x, variant=1)  
  
 if n == 4:  
 print('ŷ = b0 + b1 \* x1 + b2 \* x2 + b3 \* x3')  
 norm\_x = [  
 [+1, -1, -1, -1],  
 [+1, -1, +1, +1],  
 [+1, +1, -1, +1],  
 [+1, +1, +1, -1],  
 ]  
 x = [  
 [1, x1\_min, x2\_min, x3\_min],  
 [1, x1\_min, x2\_max, x3\_max],  
 [1, x1\_max, x2\_min, x3\_max],  
 [1, x1\_max, x2\_max, x3\_min],  
 ]  
 if n == 14:  
 y = [[round(sum([koefs[j] \* i[j] for j in range(len(koefs))]) + random() \* 10 - 5, 3) for k in range(m)] for i  
 in x]  
 else:  
 y = np.random.randint(y\_min, y\_max, size=(n, m))  
 # y = np.random.randint(y\_min, y\_max, size=(n, m))  
 y\_av = list(np.average(y, axis=1))  
  
 for i in range(len(y\_av)):  
 y\_av[i] = round(y\_av[i], 3)  
  
 if n == 14:  
 t1 = PrettyTable(['N', 'norm\_x\_0', 'norm\_x\_1', 'norm\_x\_2', 'norm\_x\_3', 'norm\_x\_1\_x\_2', 'norm\_x\_1\_x\_3',  
 'norm\_x\_2\_x\_3', 'norm\_x\_1\_x\_2\_x\_3', 'norm\_x\_1\_x\_1', 'norm\_x\_2\_x\_2', 'norm\_x\_3\_x\_3', ] +  
 [f'y\_{i + 1}' for i in range(m)] + ['y\_av'])  
 t2 = PrettyTable(['N', 'x\_0', 'x\_1', 'x\_2', 'x\_3', 'x\_1\_x\_2', 'x\_1\_x\_3', 'x\_2\_x\_3', 'x\_1\_x\_2\_x\_3', 'x\_1\_x\_1',  
 'x\_2\_x\_2', 'x\_3\_x\_3'] + [f'y\_{i + 1}' for i in range(m)] + ['y\_av'])  
  
 if n == 8:  
 t = PrettyTable(['N', 'norm\_x\_0', 'norm\_x\_1', 'norm\_x\_2', 'norm\_x\_3', 'norm\_x\_1\_x\_2', 'norm\_x\_1\_x\_3',  
 'norm\_x\_2\_x\_3', 'norm\_x\_1\_x\_2\_x\_3', 'x\_0', 'x\_1', 'x\_2', 'x\_3', 'x\_1\_x\_2', 'x\_1\_x\_3',  
 'x\_2\_x\_3', 'x\_1\_x\_2\_x\_3'] + [f'y\_{i + 1}' for i in range(m)] + ['y\_av'])  
 if n == 4:  
 t = PrettyTable(  
 ['N', 'norm\_x\_0', 'norm\_x\_1', 'norm\_x\_2', 'norm\_x\_3', 'x\_0', 'x\_1', 'x\_2', 'x\_3'] +  
 [f'y\_{i + 1}' for i in range(m)] + ['y\_av'])  
  
 if n == 14:  
 for i in range(n):  
 t1.add\_row([i + 1] + list(norm\_x[i]) + list(y[i]) + [y\_av[i]])  
 t2.add\_row([i + 1] + list(x[i]) + list(y[i]) + [y\_av[i]])  
 print(t1)  
 print(t2)  
 else:  
 for i in range(n):  
 t.add\_row([i + 1] + list(norm\_x[i]) + list(x[i]) + list(y[i]) + [y\_av[i]])  
 print(t)  
  
 m\_ij = []  
 for i in range(len(x[0])):  
 m\_ij.append([round(sum([x[k][i] \* x[k][j] for k in range(len(x))]) / 14, 3) for j in range(len(x[i]))])  
  
 k\_i = []  
 for i in range(len(x[0])):  
 a = sum(y\_av[j] \* x[j][i] for j in range(len(x))) / 14  
 k\_i.append(a)  
  
 det = np.linalg.det(m\_ij)  
 det\_i = [np.linalg.det(replace\_column(m\_ij, i, k\_i)) for i in range(len(k\_i))]  
  
 b\_i = [round(i / det, 3) for i in det\_i]  
 if n == 14:  
 print(  
 f"\nНормалізоване рівняння регресії: "  
 f"y = {b\_i[0]:.5f} + {b\_i[1]:.5f} \* x1 + {b\_i[2]:.5f} \* x2 + "  
 f"{b\_i[3]:.5f} \* x3 + {b\_i[4]:.5f} \* x1 \* x2 + "  
 f"{b\_i[5]:.5f} \* x1 \* x3 + {b\_i[6]:.5f} \* x2 \* x3 + {b\_i[7]:.5f} \* x1 \* x2 \* x3 + {b\_i[8]:.5f} \* x1 \* x1 + "  
 f"{b\_i[9]:.5f} \* x2 \* x2 + {b\_i[10]:.5f} \* x3 \* x3")  
 if n == 8:  
 print(  
 f"\nНормалізоване рівняння регресії: "  
 f"y = {b\_i[0]:.5f} + {b\_i[1]:.5f} \* x1 + {b\_i[2]:.5f} \* x2 + "  
 f"{b\_i[3]:.5f} \* x3 + {b\_i[4]:.5f} \* x1 \* x2 + "  
 f"{b\_i[5]:.5f} \* x1 \* x3 + {b\_i[6]:.5f} \* x2 \* x3 + {b\_i[7]:.5f} \* x1 \* x2 \* x3")  
 if n == 4:  
 print(  
 f"\nНормалізоване рівняння регресії: "  
 f"y = {b\_i[0]:.5f} + {b\_i[1]:.5f} \* x1 + {b\_i[2]:.5f} \* x2 + {b\_i[3]:.5f} \* x3\n")  
  
 check\_i = [round(sum(b\_i[j] \* i[j] for j in range(len(b\_i))), 3) for i in x]  
 for i in range(len(check\_i)):  
 print(f'ŷ{i + 1} = {check\_i[i]}, y\_av{i + 1} = {y\_av[i]}')  
  
 print("-" \* 100)  
 print("Критерій Кохрена")  
 start\_time = datetime.now()  
 f\_1 = m - 1  
 f\_2 = n  
 s\_i = [sum([(i - y\_av[j]) \*\* 2 for i in y[j]]) / m for j in range(len(y))]  
 g\_p = max(s\_i) / sum(s\_i)  
  
 table = {2: 0.75, 3: 0.6841, 4: 0.6287, 5: 0.5892, 6: 0.5598, 7: 0.5365, 8: 0.5175, 9: 0.5017, 10: 0.4884,  
 range(11, 17): 0.4366, range(17, 37): 0.3720, range(37, 2 \*\* 100): 0.3093}  
 g\_t = get\_value(table, m)  
  
 if g\_p < g\_t:  
 print(f"Тест тривав {(datetime.now() - start\_time).total\_seconds()} сек.")  
 print(f"Дисперсія однорідна: Gp = {g\_p:.5} < Gt = {g\_t}")  
 else:  
 print(f"Тест тривав {(datetime.now() - start\_time).total\_seconds()} сек.")  
 print(f"Дисперсія неоднорідна: Gp = {g\_p:.5} > Gt = {g\_t}\nПочнемо знову з m = m + 1 = {m + 1}")  
 return main(m=m + 1, n=n)  
  
 print("-" \* 100)  
 print("Критерій Стьюдента")  
 start\_time = datetime.now()  
 s2\_b = sum(s\_i) / n  
 s2\_beta\_s = s2\_b / (n \* m)  
 s\_beta\_s = sqrt(s2\_beta\_s)  
 beta\_i = [sum([norm\_x[i][j] \* y\_av[i] for i in range(len(norm\_x))]) / n for j in range(len(norm\_x[0]))]  
 beta\_i = [round(i, 3) for i in beta\_i]  
  
 t = [abs(i) / s\_beta\_s for i in beta\_i]  
 if n == 14:  
 beta\_i = b\_i  
 f\_3 = f\_1 \* f\_2  
 t\_table = {4: 2.776, 5: 2.571, 6: 2.447, 7: 2.365, 8: 2.306, 9: 2.262, 10: 2.228, 11: 2.201, 12: 2.179, 13: 2.160,  
 14: 2.145, 15: 2.131, 16: 2.120, 17: 2.110, 18: 2.101, 19: 2.093, 20: 2.086, 21: 2.08, 22: 2.074,  
 23: 2.069, 24: 2.064, range(25, 30): 2.06, range(30, 40): 2.042, range(40, 60): 2.021, range(60, 100): 2,  
 range(100, 2 \*\* 100): 1.96}  
 d = deepcopy(len(beta\_i))  
 for i in range(len(t)):  
 if get\_value(t\_table, f\_3) > t[i]:  
 beta\_i[i] = 0  
 d -= 1  
 print(f"Тест тривав {(datetime.now() - start\_time).total\_seconds()} сек.")  
 print("-" \* 100)  
 if n == d:  
 n = 8 if n == 4 else 14  
 print(f"n=d\nПочнемо знову з n = {n} and m = {m}")  
 return main(m=m, n=n)  
 if n == 14:  
 print(  
 f"Нормалізоване спрощене рівяння регресії: "  
 f"y = {beta\_i[0]:.5f} + {beta\_i[1]:.5f} \* x1 + "  
 f"{beta\_i[2]:.5f} \* x2 + {beta\_i[3]:.5f} \* x3 + {beta\_i[4]:.5f} \* x1 \* x2 + "  
 f"{beta\_i[5]:.5f} \* x1 \* x3 + {beta\_i[6]:.5f} \* x2 \* x3 + {beta\_i[7]:.5f} \* x1 \* x2 \* x3 + "  
 f"{beta\_i[8]:.5f} \* x1 \* x1 + {beta\_i[9]:.5f} \* x2 \* x2 + {beta\_i[10]:.5f} \* x3 \* x3")  
 check\_i = [round(sum(beta\_i[j] \* i[j] for j in range(len(beta\_i))), 3) for i in x]  
  
 if n == 8:  
 print(  
 f"Нормалізоване рівняння регресії: "  
 f"y = {beta\_i[0]:.5f} + {beta\_i[1]:.5f} \* x1 + {beta\_i[2]:.5f} \* x2 + "  
 f"{beta\_i[3]:.5f} \* x3 + {beta\_i[4]:.5f} \* x1 \* x2 + "  
 f"{beta\_i[5]:.5f} \* x1 \* x3 + {beta\_i[6]:.5f} \* x2 \* x3 + {beta\_i[7]:.5f} \* x1 \* x2 \* x3")  
 check\_i = [round(sum(beta\_i[j] \* i[j] for j in range(len(beta\_i))), 3) for i in norm\_x]  
  
 if n == 4:  
 print(  
 f"Нормалізоване рівняння регресії: "  
 f"y = {beta\_i[0]:.5f} + {beta\_i[1]:.5f} \* x1 + {beta\_i[2]:.5f} \* x2 + "  
 f"{beta\_i[3]:.5f} \* x3")  
 check\_i = [round(sum(beta\_i[j] \* i[j] for j in range(len(beta\_i))), 3) for i in norm\_x]  
  
 for i in range(len(check\_i)):  
 print(f'ŷ{i + 1} = {check\_i[i]}, y\_av{i + 1} = {y\_av[i]}')  
  
 print("-" \* 100)  
 print("Критерій Фішера")  
 start\_time = datetime.now()  
 f\_4 = n - d  
 s2\_ad = m / f\_4 \* sum([(check\_i[i] - y\_av[i]) \*\* 2 for i in range(len(y\_av))])  
 f\_p = s2\_ad / s2\_b  
 f\_t = {  
 1: [164.4, 199.5, 215.7, 224.6, 230.2, 234, 235.8, 237.6],  
 2: [18.5, 19.2, 19.2, 19.3, 19.3, 19.3, 19.4, 19.4],  
 3: [10.1, 9.6, 9.3, 9.1, 9, 8.9, 8.8, 8.8],  
 4: [7.7, 6.9, 6.6, 6.4, 6.3, 6.2, 6.1, 6.1],  
 5: [6.6, 5.8, 5.4, 5.2, 5.1, 5, 4.9, 4.9],  
 6: [6, 5.1, 4.8, 4.5, 4.4, 4.3, 4.2, 4.2],  
 7: [5.5, 4.7, 4.4, 4.1, 4, 3.9, 3.8, 3.8],  
 8: [5.3, 4.5, 4.1, 3.8, 3.7, 3.6, 3.5, 3.5],  
 9: [5.1, 4.3, 3.9, 3.6, 3.5, 3.4, 3.3, 3.3],  
 10: [5, 4.1, 3.7, 3.5, 3.3, 3.2, 3.1, 3.1],  
 11: [4.8, 4, 3.6, 3.4, 3.2, 3.1, 3, 3],  
 12: [4.8, 3.9, 3.5, 3.3, 3.1, 3, 2.9, 2.9],  
 13: [4.7, 3.8, 3.4, 3.2, 3, 2.9, 2.8, 2.8],  
 14: [4.6, 3.7, 3.3, 3.1, 3, 2.9, 2.8, 2.7],  
 15: [4.5, 3.7, 3.3, 3.1, 2.9, 2.8, 2.7, 2.7, 2.7, 2.7, 2.6, 2.6],  
 16: [4.5, 3.6, 3.2, 3, 2.9, 2.7, 2.6, 2.6],  
 17: [4.5, 3.6, 3.2, 3, 2.8, 2.7, 2.5, 2.3],  
 18: [4.4, 3.6, 3.2, 2.9, 2.8, 2.7, 2.5, 2.3],  
 19: [4.4, 3.5, 3.1, 2.9, 2.7, 2.7, 2.4, 2.3],  
 range(20, 22): [4.4, 3.5, 3.1, 2.8, 2.7, 2.7, 2.4, 2.3],  
 range(22, 24): [4.3, 3.4, 3.1, 2.8, 2.7, 2.6, 2.4, 2.3],  
 range(24, 26): [4.3, 3.4, 3, 2.8, 2.6, 2.5, 2.3, 2.2],  
 range(26, 28): [4.2, 3.4, 3, 2.7, 2.6, 2.5, 2.3, 2.2],  
 range(28, 30): [4.2, 3.3, 3, 2.7, 2.6, 2.4, 2.3, 2.1],  
 range(30, 40): [4.2, 3.3, 3, 2.7, 2.6, 2.4, 2.3, 2.1, 2, 2, 2, 2],  
 range(40, 60): [4.1, 3.2, 2.9, 2.6, 2.5, 2.3, 2.2, 2, 1.9, 1.9, 1.9, 1.9],  
 range(60, 120): [4, 3.2, 2.8, 2.5, 2.4, 2.3, 2.1, 1.9, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8],  
 range(120, 2 \*\* 100): [3.8, 3, 2.6, 2.4, 2.2, 2.1, 2, 2, 1.9, 1.9, 1.9, 1.8, 1.8]  
 }  
 if f\_p > get\_value(f\_t, f\_3)[f\_4]:  
 print(f"Тест тривав {(datetime.now() - start\_time).total\_seconds()} сек.")  
 n = 8 if n == 4 else 14  
 print(  
 f"fp = {f\_p} > ft = {get\_value(f\_t, f\_3)[f\_4]}.\n"  
 f"Математична модель неадекватна з цими експериментальними даними\n"  
 f"-----------------------------------------------------------------"  
 f"\nПочнемо знову з m = {m} and n = {n}")  
 return main(m=m, n=n)  
 else:  
 print(f"Тест тривав {(datetime.now() - start\_time).total\_seconds()} сек.")  
 print(  
 f"fP = {f\_p} < fT = {get\_value(f\_t, f\_3)[f\_4]}.\n"  
 f"Математична модель адекватна\n")  
  
main(m=2, n=14)

2. Результат

"C:\Python 3.7.4\python.exe" C:/mope/Lab6/main.py

y=b0+b1\*x1+b2\*x2+b3\*x3+b12\*x1\*x2+b13\*x1\*x3+b23\*x2\*x3+b123\*x1\*x2\*x3+b11\*x1^2+b22\*x2^2+b33\*x3^2

Кодованє значення X

№ X1 X2 X3 X4 X5 X6 X7 X8 X9 X10

1 -1 -1 -1 1 1 1 -1 1 1 1

2 -1 -1 1 1 -1 -1 1 1 1 1

3 -1 1 -1 -1 1 -1 1 1 1 1

4 -1 1 1 -1 -1 1 -1 1 1 1

5 1 -1 -1 -1 -1 1 1 1 1 1

6 1 -1 1 -1 1 -1 -1 1 1 1

7 1 1 -1 1 -1 -1 -1 1 1 1

8 1 1 1 1 1 1 1 1 1 1

9 -1.73 0 0 -0.0 -0.0 0 -0.0 2.993 0 0

10 1.73 0 0 0.0 0.0 0 0.0 2.993 0 0

11 0 -1.73 0 -0.0 0 -0.0 -0.0 0 2.993 0

12 0 1.73 0 0.0 0 0.0 0.0 0 2.993 0

13 0 0 -1.73 0 -0.0 -0.0 -0.0 0 0 2.993

14 0 0 1.73 0 0.0 0.0 0.0 0 0 2.993

15 0 0 0 0 0 0 0 0 0 0

Матриця для m=15

№ X0 X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 Y1 Y2 Y3

1 1 -25 5 15 -125 -375 75 -1875 625 25 225 -4492.1 -4498.1 -4491.1

2 1 -25 5 25 -125 -625 125 -3125 625 25 625 -7858.0999999999985 -7861.0999999999985 -7859.0999999999985

3 1 -25 40 15 -1000 -375 600 -15000 625 1600 225 -71350.6 -71359.6 -71356.6

4 1 -25 40 25 -1000 -625 1000 -25000 625 1600 625 -119171.59999999999 -119165.59999999999 -119167.59999999999

5 1 75 5 15 375 1125 75 5625 5625 25 225 66348.9 66340.9 66345.9

6 1 75 5 25 375 1875 125 9375 5625 25 625 89281.9 89284.9 89284.9

7 1 75 40 15 3000 1125 600 45000 5625 1600 225 274239.39999999997 274234.39999999997 274230.39999999997

8 1 75 40 25 3000 1875 1000 75000 5625 1600 625 431227.4 431221.4 431221.4

9 1 -61.5 22.5 20.0 -1383.75 -1230.0 450.0 -27675.0 3782.25 506.25 400.0 -118355.7 -118348.7 -118346.7

10 1 111.5 22.5 20.0 2508.75 2230.0 450.0 50175.0 12432.25 506.25 400.0 341816.0 341809.0 341810.0

11 1 25.0 -7.775 20.0 -194.375 500.0 -155.5 -3887.5 625.0 60.451 400.0 -12798.631999999996 -12800.631999999996 -12805.631999999996

12 1 25.0 52.775 20.0 1319.375 500.0 1055.5 26387.5 625.0 2785.201 400.0 147936.453 147940.453 147943.453

13 1 25.0 22.5 11.35 562.5 283.75 255.375 6384.375 625.0 506.25 128.822 39579.1455 39579.1455 39581.1455

14 1 25.0 22.5 28.65 562.5 716.25 644.625 16115.625 625.0 506.25 820.822 95258.3855 95262.3855 95261.3855

15 1 25.0 22.5 20.0 562.5 500.0 450.0 11250.0 625.0 506.25 400.0 66837.15 66839.15 66838.15

Середнє значення відгуку функції за рядками

y1av1=-18641.5=-4493.77

y2av2=-10090.74=-7859.43

y3av3=-96753.28=-71355.6

y4av4=-88202.53=-119168.27

Значення співпадають

Дисперсія по рядкам

d1= 9.56 d2= 1.56 d3= 14.0 d4= 6.22

Дисперсія однорідна

Критерій Стьюдента

f3 = f1\*f2, з таблиці tтабл = 2.306

Критерій Фішера

4 значимих коефіцієнтів

d1= 9.56 d2= 1.56 d3= 14.0 d4= 6.22 d5= 7.83

Fp= 14014987708.95

Ft берем із таблиці 8 рядяк 2 стовпець Ft = 4.5

Fp= 14014987708.95 >Ft 4.5 Рівняння неадекватно оригіналу

З УРАХУВАННЯМ ВЗАЄМОДІЇ

[[1, -25, 5, 15, -125, -375, 75, -1875, 625, 25, 225], [1, -25, 5, 25, -125, -625, 125, -3125, 625, 25, 625], [1, -25, 40, 15, -1000, -375, 600, -15000, 625, 1600, 225], [1, -25, 40, 25, -1000, -625, 1000, -25000, 625, 1600, 625], [1, 75, 5, 15, 375, 1125, 75, 5625, 5625, 25, 225], [1, 75, 5, 25, 375, 1875, 125, 9375, 5625, 25, 625], [1, 75, 40, 15, 3000, 1125, 600, 45000, 5625, 1600, 225], [1, 75, 40, 25, 3000, 1875, 1000, 75000, 5625, 1600, 625], [1, -61.5, 22.5, 20.0, -1383.75, -1230.0, 450.0, -27675.0, 3782.25, 506.25, 400.0], [1, 111.5, 22.5, 20.0, 2508.75, 2230.0, 450.0, 50175.0, 12432.25, 506.25, 400.0], [1, 25.0, -7.774999999999999, 20.0, -194.37499999999997, 500.0, -155.49999999999997, -3887.4999999999995, 625.0, 60.45062499999998, 400.0], [1, 25.0, 52.775, 20.0, 1319.375, 500.0, 1055.5, 26387.5, 625.0, 2785.200625, 400.0], [1, 25.0, 22.5, 11.35, 562.5, 283.75, 255.375, 6384.375, 625.0, 506.25, 128.8225], [1, 25.0, 22.5, 28.65, 562.5, 716.25, 644.625, 16115.625, 625.0, 506.25, 820.8224999999999], [1, 25.0, 22.5, 20.0, 562.5, 500.0, 450.0, 11250.0, 625.0, 506.25, 400.0]]

y1av1=234156.46=-4493.77

y2av2=139453.42=-7859.43

y3av3=-3736532.8=-71355.6

y4av4=-5906645.61=-119168.27

y5av5=1127317.78=66345.23

y6av6=1289441.86=89283.9

y7av7=12117089.07=274234.73

y8av8=17834923.39=431223.4

Значення приблизно співпадають

Дисперсія по рядкам

d1=9.56

d2=1.56

d3=14.0

d4=6.22

d5=10.89

d6=2.0

d7=13.56

d8=8.0

Дисперсія однорідна

Критерій Стьюдента

f3 = f1\*f2, з таблиці tтабл = 2.306

Критерій Фішера

8 значимих коефіцієнтів

Fp= 151493792767595.8

Ft берем із таблиці 16 рядяк 1 стовпець Ft = 2.4

Fp= 151493792767595.8 >Ft 2.4 Рівняння неадекватно оригіналу

З УРАХУВАННЯМ ВЗАЄМОДІЇ І З УРАХУВАННЯМ КВАДРАТИЧНИХ ЕЛЕМЕНТІВ

y1av1=12233918.5=-4493.77

y2av2=29334068.72=-7859.43

y3av3=226727339.34=-71355.6

y4av4=346845625.51=-119168.27

y5av5=-230866801.59=66345.23

y6av6=-285524026.05=89283.9

y7av7=-715883076.69=274234.73

y8av8=-1096553651.38=431223.4

y9av9=285005845.56=-118350.37

y10av10=-988260534.06=341811.67

y11av11=33203690.99=-12801.63

y12av12=-297244008.13=147940.12

y13av13=-94123453.38=39579.81

y14av14=-223055752.77=95260.72

y15av15=-158524842.62=66838.15

Значення приблизно співпадають

Дисперсія по рядкам

d1=9.56

d2=1.56

d3=14.0

d4=6.22

d5=10.89

d6=2.0

d7=13.56

d8=8.0

d9=14.89

d10=9.56

d11=8.67

d12=8.22

d13=0.89

d14=2.89

d15=0.67

Дисперсія однорідна

Критерій Стьюдента

f3 = f1\*f2, з таблиці tтабл = 2.042

Критерій Фішера

11 значимих коефіцієнтів

Fp= 2.8444857515961178e+17

Ft берем із таблиці 16 рядяк 4 стовпець Ft = 2.4

Fp= 2.8444857515961178e+17 >Ft 2.4 Рівняння неадекватно оригіналу

y= -1421.3340255115077 + 15703.767334785694 \*x1+ 12320.303519604446 \*x2+ 11269.149729662651 \*x3+ -15989.27619680167 \*x1\*x2+ -10467.162377899864 \*x1\*x3 -12113.53026436437 \*x2\*x3 -12258.042462000874 \*x1\*x2\*x3 -25808.07933822523 \*x1^2 28917.07277398852 \*x2^2 -865.521082698843 \*x3^2

Process finished with exit code 0