Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

з дисципліни «Методи оптимізації та планування експерименту»

# «Проведення трьохфакторного експерименту

# при використанні рівняння регресії з урахуванням ефекту взаємодії»

Виконав:

студент групи ІО-82

Кузенний П.В.

Залікова книжка № IO-8209

Варіант: 208

Перевірив:

Регіда П.Г.

Київ 2020

# Мета: провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

# Завдання:

1. Скласти матрицю планування для повного трьохфакторного експерименту.

2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і

знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у

певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в

журналі викладача.

3. Знайти коефіцієнти рівняння регресії і записати його.

4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.

5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати

скореговане рівняння регресії.

6. Написати комп'ютерну програму, яка усе це моделює.

**Варіант:**



**Код програми:**

import numpy as np  
from math import \*  
from prettytable import PrettyTable  
from scipy.stats import f, t  
from functools import partial  
  
# Variant №208  
  
while True:  
 fisher\_t = partial(f.ppf, q=1 - 0.05)  
 student\_t = partial(t.ppf, q=1 - 0.025)  
  
 x1\_min = -5  
 x1\_max = 15  
 x2\_min = -35  
 x2\_max = 10  
 x3\_min = -35  
 x3\_max = -10  
  
 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)  
  
 m = 3  
 N = 8  
  
 x0\_factor = np.array([1, 1, 1, 1, 1, 1, 1, 1])  
 x1\_factor = np.array([-1, -1, -1, -1, 1, 1, 1, 1])  
 x2\_factor = np.array([-1, -1, 1, 1, -1, -1, 1, 1])  
 x3\_factor = np.array([-1, 1, -1, 1, -1, 1, -1, 1])  
 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  
  
 factor\_matrix = np.zeros((N, N))  
  
 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  
  
 # print(factor\_matrix)  
  
 matrix\_plan = np.random.randint(y\_min, y\_max, size=(N, m))  
  
 y\_average = np.zeros((N, 1))  
 for i in range(N):  
 y\_average[i, 0] = round((sum(matrix\_plan[i, :] / 3)), 3)  
  
 list\_bi = np.zeros((N, 1))  
 for i in range(N):  
 list\_bi[i, 0] = sum(factor\_matrix[i, :] \* y\_average[:, 0] / 3)  
  
 x0 = np.array([1, 1, 1, 1, 1, 1, 1, 1])  
 x1 = np.array([-5, -5, -5, -5, 15, 15, 15, 15])  
 x2 = np.array([-35, -35, 10, 10, -35, -35, 10, 10])  
 x3 = np.array([-35, -10, -35, -10, -35, -10, -35, -10])  
 x1x2 = x1 \* x2  
 x1x3 = x1 \* x3  
 x2x3 = x2 \* x3  
 x1x2x3 = x1 \* x2 \* x3  
  
 x\_matrix = np.zeros((N, N))  
  
 x\_matrix[:, 0] = x0  
 x\_matrix[:, 1] = x1  
 x\_matrix[:, 2] = x2  
 x\_matrix[:, 3] = x3  
 x\_matrix[:, 4] = x1x2  
 x\_matrix[:, 5] = x1x3  
 x\_matrix[:, 6] = x2x3  
 x\_matrix[:, 7] = x1x2x3  
  
 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\_matrix, matrix\_plan, y\_average, d\_list))  
  
 table = PrettyTable()  
 table.field\_names = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3", "Y1", "Y2", "Y3", "Y", "S^2"]  
 for i in range(len(my\_table)):  
 table.add\_row(my\_table[i])  
  
 print(table)  
 print("\ny = {} + {}\*x1 + {}\*x2 + {}\*x3 + {}\*x1x2 + {}\*x1x3 + {}\*x2x3 + {}\*x1x2x3 \n".format(round(float(list\_bi[0]), 3),  
 round(float(list\_bi[1]), 3),  
 round(float(list\_bi[2]), 3),  
 round(float(list\_bi[3]), 3),  
 round(float(list\_bi[4]), 3),  
 round(float(list\_bi[5]), 3),  
 round(float(list\_bi[6]), 3),  
 round(float(list\_bi[7]), 3)))  
  
 Gp = max(d\_list) / d\_sum  
 F1 = m - 1  
 F2 = N  
 q = 0.05  
 q1 = q / F1  
 fisher\_value = f.ppf(q=1 - q1, dfn=F2, dfd=(F1 - 1) \* F2)  
 Gt = fisher\_value / (fisher\_value + F1 - 1)  
 print("Gp = ", float(Gp), "\nGt = ", Gt)  
  
 if Gp < Gt:  
 print("Gp < Gt")  
 print("Дисперсія однорідна\n")  
 dispersion\_b = (d\_sum / N) / (m \* N)  
 s\_beta = sqrt(abs(dispersion\_b))  
  
 beta\_list = np.zeros((N, 1))  
 for i in range(N):  
 beta\_list[i, 0] = sum(factor\_matrix[i, :] \* y\_average[:, 0] / N)  
  
 t\_list = []  
 for i in range(N):  
 t\_list.append(abs(beta\_list[i, 0]) / s\_beta)  
  
 F3 = F1 \* F2  
 d = 0  
 T = student\_t(df=F3)  
 print("t табличне = ", T)  
 for i in range(len(t\_list)):  
 if t\_list[i] > T:  
 beta\_list[i, 0] = 0  
 print("Гіпотеза підтверджена, beta{} = 0".format(i))  
 else:  
 print("Гіпотеза не підтверджена beta{} = {}".format(i, beta\_list[i]))  
 d += 1  
  
 y\_for\_student = np.zeros((N, 1))  
 for i in range(N):  
 y\_for\_student[i, 0] = sum(x\_matrix[i, :] \* beta\_list[:, 0])  
  
 F4 = N - d  
 dispersion = sum(((y\_for\_student[:][0] - y\_average[:][0]) \*\* 2) \* m / (N - d))  
 Fp = dispersion / dispersion\_b  
 Ft = fisher\_t(dfn=F4, dfd=F3)  
 if Ft > Fp:  
 print("Отримана математична модель адекватна експериментальним даним")  
 break  
 else:  
 print("Рівняння регресії неадекватно оригіналу")  
 break  
  
 else:  
 print("Gp > Gt")  
 print("Дисперсія неоднорідна. Спробуйте ще раз.")  
 m += 1