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-83

Мартинюк Н.О.

Залікова книжка № IO-8305

Варіант: 305

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 itertools import product, combinations  
from scipy.stats import t, f  
  
np.set\_printoptions(formatter={'float\_kind': lambda x: "%.2f"%(x)})  
  
  
  
def tableStudent(prob, n, m):  
 x\_vec = [i\*0.0001 for i in range(int(5/0.0001))]  
 par = 0.5 + prob/0.1\*0.05  
 f3 = (m - 1) \* n  
 for i in x\_vec:  
 if abs(t.cdf(i, f3) - par) < 0.000005:  
 return i  
  
  
def tableFisher(prob, n, m, d):  
 x\_vec = [i\*0.001 for i in range(int(10/0.001))]  
 f3 = (m - 1) \* n  
 for i in x\_vec:  
 if abs(f.cdf(i, n-d, f3)-prob) < 0.0001:  
 return i  
  
def fisherCriteria(Y\_matrix, d):  
 if d == N:  
 return False  
 Sad = m / (N - d) \* np.mean(check1 - mean\_Y)  
 mean\_dispersion = np.mean(np.mean((Y\_matrix.T - mean\_Y) \*\* 2, axis=0))  
 Fp = Sad / mean\_dispersion  
 if (m-1)\*N > 32:  
 if N-d > 6:  
 return tableFisher(0.95, N, m, d)  
 return Fp < tableFisher(0.95, N, m, d)  
 if N - d > 6:  
 return Fp < tableFisher(0.95, N, m, d)  
 return Fp < tableFisher(0.95, N, m, d)  
  
def make\_norm\_plan\_matrix(plan\_matrix, matrix\_of\_min\_and\_max\_x):  
 X0 = np.mean(matrix\_with\_min\_max\_x, axis=1)  
 interval\_of\_change = np.array([(matrix\_of\_min\_and\_max\_x[i, 1] - X0[i]) for i in range(len(plan\_matrix[0]))])  
 X\_norm = np.array(  
 [[round((plan\_matrix[i, j] - X0[j]) / interval\_of\_change[j], 3) for j in range(len(plan\_matrix[i]))]  
 for i in range(len(plan\_matrix))])  
 return X\_norm  
  
  
def cochranCheck(Y\_matrix):  
 mean\_Y = np.mean(Y\_matrix, axis=1)  
 dispersion\_Y = np.mean((Y\_matrix.T - mean\_Y) \*\* 2, axis=0)  
 Gp = np.max(dispersion\_Y) / (np.sum(dispersion\_Y))  
 fisher = tableFisher(0.95, N, m, 1)  
 Gt = fisher / (fisher + (m - 1) - 2)  
 return Gp < Gt  
  
  
def students\_t\_test(norm\_matrix, Y\_matrix):  
 mean\_Y = np.mean(Y\_matrix, axis=1)  
 dispersion\_Y = np.mean((Y\_matrix.T - mean\_Y) \*\* 2, axis=0)  
 mean\_dispersion = np.mean(dispersion\_Y)  
 sigma = np.sqrt(mean\_dispersion / (N \* m))  
 betta = np.mean(norm\_matrix.T \* mean\_Y, axis=1)  
 t = np.abs(betta) / sigma  
 if (m - 1) \* N > 32:  
 return np.where(t > tableStudent(0.95, N, m))  
 return np.where(t > tableStudent(0.95, N, m))  
  
  
matrix\_with\_min\_max\_x = np.array([[15, 45], [30, 80], [15, 45]])  
m = 6  
N = 8  
norm\_matrix = np.array(list(product("01", repeat=3)), dtype=np.int)  
norm\_matrix[norm\_matrix == 0] = -1  
norm\_matrix = np.insert(norm\_matrix, 0, 1, axis=1)  
plan\_matrix = np.empty((8, 3))  
for i in range(len(norm\_matrix)):  
 for j in range(1, len(norm\_matrix[i])):  
 if j == 1:  
 if norm\_matrix[i, j] == -1:  
 plan\_matrix[i, j-1] = 15  
 elif norm\_matrix[i, j] == 1:  
 plan\_matrix[i, j-1] = 45  
 elif j == 2:  
 if norm\_matrix[i, j] == -1:  
 plan\_matrix[i, j-1] = 30  
 elif norm\_matrix[i, j] == 1:  
 plan\_matrix[i, j-1] = 80  
 elif j == 3:  
 if norm\_matrix[i, j] == -1:  
 plan\_matrix[i, j-1] = 15  
 elif norm\_matrix[i, j] == 1:  
 plan\_matrix[i, j-1] = 45  
plan\_matr = np.insert(plan\_matrix, 0, 1, axis=1)  
Y\_matrix = np.random.randint(200 + np.mean(matrix\_with\_min\_max\_x, axis=0)[0],  
 200 + np.mean(matrix\_with\_min\_max\_x, axis=0)[1], size=(N, m))  
mean\_Y = np.mean(Y\_matrix, axis=1)  
combination = list(combinations(range(1, 4), 2))  
  
for i in combination:  
 plan\_matr = np.append(plan\_matr, np.reshape(plan\_matr[:, i[0]]\*plan\_matr[:, i[1]], (8, 1)), axis=1)  
 norm\_matrix = np.append(norm\_matrix, np.reshape(norm\_matrix[:, i[0]]\*norm\_matrix[:, i[1]], (8, 1)), axis=1)  
plan\_matr = np.append(plan\_matr, np.reshape(plan\_matr[:, 1]\*plan\_matr[:, 2]\*plan\_matr[:, 3], (8, 1)), axis=1)  
norm\_matrix = np.append(norm\_matrix, np.reshape(norm\_matrix[:, 1]\*norm\_matrix[:, 2]\*norm\_matrix[:, 3], (8, 1)), axis=1)  
  
if cochranCheck(Y\_matrix):  
 b\_natura = np.linalg.lstsq(plan\_matr, mean\_Y, rcond=None)[0]  
 b\_norm = np.linalg.lstsq(norm\_matrix, mean\_Y, rcond=None)[0]  
 check1 = np.sum(b\_natura \* plan\_matr, axis=1)  
 check2 = np.sum(b\_norm \* norm\_matrix, axis=1)  
 indexes = students\_t\_test(norm\_matrix, Y\_matrix)  
 print("Matrix Experiment Plan: \n", plan\_matr)  
 print("Normalized matrix: \n", norm\_matrix)  
 print("Matrix reviews: \n", Y\_matrix)  
 print("The average values of Y: ", mean\_Y)  
 print("Naturalized coefficients: ", b\_natura)  
 print("Normalized coefficients: ", b\_norm)  
 print("Check #1: ", check1)  
 print("Check #2: ", check2)  
 print("Indices of coefficients that satisfy the Student's criterion: ", np.array(indexes)[0])  
 print("Student test: ", np.sum(b\_natura[indexes] \* np.reshape(plan\_matr[:, indexes], (N, np.size(indexes))), axis=1))  
 if fisherCriteria(Y\_matrix, np.size(indexes)):  
 print("The regression equation is adequate to the original.")  
 else:  
 print("The regression equation is inadequate to the original.")  
else:  
 print("Dispersion is heterogeneous")

**Результати виконання:**

Matrix Experiment Plan:

[[1.00 15.00 30.00 15.00 450.00 225.00 450.00 6750.00]

[1.00 15.00 30.00 45.00 450.00 675.00 1350.00 20250.00]

[1.00 15.00 80.00 15.00 1200.00 225.00 1200.00 18000.00]

[1.00 15.00 80.00 45.00 1200.00 675.00 3600.00 54000.00]

[1.00 45.00 30.00 15.00 1350.00 675.00 450.00 20250.00]

[1.00 45.00 30.00 45.00 1350.00 2025.00 1350.00 60750.00]

[1.00 45.00 80.00 15.00 3600.00 675.00 1200.00 54000.00]

[1.00 45.00 80.00 45.00 3600.00 2025.00 3600.00 162000.00]]

Normalized matrix:

[[ 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 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]]

Matrix reviews:

[[250 231 251 253 245 221]

[221 220 246 253 233 230]

[239 248 231 249 234 240]

[248 224 247 251 238 234]

[243 232 228 238 243 225]

[252 252 236 221 226 222]

[236 249 230 231 229 227]

[247 252 222 230 243 224]]

The average values of Y: [241.83 233.83 240.17 240.33 234.83 234.83 233.67 236.33]

Naturalized coefficients: [255.76 -0.43 -0.15 -0.62 0.00 0.01 0.01 -0.00]

Normalized coefficients: [236.98 -2.06 0.65 -0.65 -0.56 1.31 1.35 -0.69]

Check #1: [241.83 233.83 240.17 240.33 234.83 234.83 233.67 236.33]

Check #2: [241.83 233.83 240.17 240.33 234.83 234.83 233.67 236.33]

Indices of coefficients that satisfy the Student's criterion: [0]

Student test: [255.76 255.76 255.76 255.76 255.76 255.76 255.76 255.76]

The regression equation is adequate to the original.

**Висновок:** Під час виконання роботи проблем не виникало. Отримані результати збігаються з очікуваними.