Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**ЛАБОРАТОРНА РОБОТА № 2**

з дисципліни «Методи оптимізації та планування»

Тема: ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З

ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

ВИКОНАВ:

студент ІІ курсу ФІОТ

групи ІО-83

Мельник Р.К.

Залікова – 8319

Варіант-317

ПЕРЕВІРИВ:

Регіда П.Г.

Київ – 2020

**Мета**: провести двофакторний експеримент, перевірити однорідність дисперсії за

критерієм Романовського, отримати коефіцієнти рівняння регресії, провести

натуралізацію рівняння регресії.

Введемо такі позначення:

N – кількість точок плану (рядків матриці планування)

k – кількість факторів(кількість x)

m – кількість дослідів y за однієї і тієї ж комбінації факторів (test)

s

x - нормовані значення факторів(s =1,k)

**Завдання на лабораторну роботу**

1. Записати лінійне рівняння регресії.

2. Обрати тип двофакторного експерименту і скласти матрицю планування для

нього з використанням додаткового нульового фактору (хо=1).

3. Провести експеримент в усіх точках повного факторного простору (знайти

значення функції відгуку y). Значення функції відгуку задати випадковим

чином у відповідності до варіанту у діапазоні ymin ч ymax

ymax = (30 - Nваріанту)\*10,

ymin = (20 - Nваріанту)\*10.

Варіанти обираються по номеру в списку в журналі викладача.

4. Перевірити однорідності дисперсії за критерієм Романовського

5. Знайти коефіцієнти нормованих рівнянь регресії і виконати перевірку

(підставити значення нормованих факторів і коефіцієнтів у рівняння).

6. Провести натуралізацію рівняння регресії й виконати перевірку

натуралізованого рівняння.

7. Написати комп'ютерну програму, яка все це виконує.

**Код програми:**

def generate\_matrix():  
 from random import randrange  
 matrix\_with\_y = [[randrange(y\_min, y\_max) for y in range(m)] for x in range(3)]  
 return list(matrix\_with\_y)  
  
  
def sum\_y(matrix\_with\_y):  
 sum\_y1, sum\_y2, sum\_y3 = 0, 0, 0  
 for j in range(m):  
 sum\_y1 += matrix\_with\_y[0][j]  
 sum\_y2 += matrix\_with\_y[1][j]  
 sum\_y3 += matrix\_with\_y[2][j]  
 return sum\_y1, sum\_y2, sum\_y3  
  
  
def find\_average\_y(sum\_of\_y):  
 average\_y1 = sum\_of\_y[0] / m  
 average\_y2 = sum\_of\_y[1] / m  
 average\_y3 = sum\_of\_y[2] / m  
 return average\_y1, average\_y2, average\_y3  
  
  
def find\_dispersion(lst\_y, average\_y):  
 dispersion\_y1, dispersion\_y2, dispersion\_y3 = 0, 0, 0  
 for k in range(m):  
 dispersion\_y1 += (lst\_y[0][k] - average\_y[0]) \*\* 2  
 dispersion\_y2 += (lst\_y[1][k] - average\_y[1]) \*\* 2  
 dispersion\_y3 += (lst\_y[2][k] - average\_y[2]) \*\* 2  
  
 dispersion\_y1 /= (m - 1)  
 dispersion\_y2 /= (m - 1)  
 dispersion\_y3 /= (m - 1)  
 return dispersion\_y1, dispersion\_y2, dispersion\_y3  
  
  
def find\_deviation():  
 from math import sqrt  
 deviation\_1 = sqrt((2 \* (2 \* m - 2)) / (m \* (m - 4)))  
 return deviation\_1  
  
  
def find\_dispersion\_uv(lst\_with\_f):  
 dispersion\_uv1 = (1 - 2 / m) \* lst\_with\_f[0]  
 dispersion\_uv2 = (1 - 2 / m) \* lst\_with\_f[1]  
 dispersion\_uv3 = (1 - 2 / m) \* lst\_with\_f[2]  
 return dispersion\_uv1, dispersion\_uv2, dispersion\_uv3  
  
  
def find\_Fuv(lst\_with\_dispersion):  
 Fuv1 = lst\_with\_dispersion[0] / lst\_with\_dispersion[1] if lst\_with\_dispersion[0] > lst\_with\_dispersion[1] \  
 else lst\_with\_dispersion[1] / lst\_with\_dispersion[0]  
 Fuv2 = lst\_with\_dispersion[2] / lst\_with\_dispersion[0] if lst\_with\_dispersion[2] > lst\_with\_dispersion[0] \  
 else lst\_with\_dispersion[0] / lst\_with\_dispersion[2]  
 Fuv3 = lst\_with\_dispersion[2] / lst\_with\_dispersion[1] if lst\_with\_dispersion[2] > lst\_with\_dispersion[1] \  
 else lst\_with\_dispersion[1] / lst\_with\_dispersion[2]  
 return Fuv1, Fuv2, Fuv3  
  
  
def find\_Ruv(lst\_with\_dispersion\_uv, deviation\_1):  
 from math import fabs  
 Ruv1 = fabs(lst\_with\_dispersion\_uv[0] - 1) / deviation\_1  
 Ruv2 = fabs(lst\_with\_dispersion\_uv[1] - 1) / deviation\_1  
 Ruv3 = fabs(lst\_with\_dispersion\_uv[2] - 1) / deviation\_1  
 return Ruv1, Ruv2, Ruv3  
  
  
def find\_Rkr(dictionary):  
 lst\_with\_keys = list(dictionary.keys())  
 lst\_with\_keys.append(m)  
 if lst\_with\_keys.count(m) == 1:  
 lst\_with\_keys.sort()  
 index\_m = lst\_with\_keys.index(m)  
 if m == lst\_with\_keys[-1]:  
 need\_key = lst\_with\_keys[index\_m - 1]  
 else:  
 need\_key = lst\_with\_keys[index\_m + 1]  
 else:  
 need\_key = m  
 return dictionary.get(need\_key)  
  
  
def det(a):  
 from numpy.linalg import det  
 return det(a)  
  
  
def delta\_x(min\_x, max\_x):  
 from math import fabs  
 return fabs(max\_x - min\_x) / 2  
  
  
x1\_min = -10  
x1\_max = 50  
x2\_min = 20  
x2\_max = 60  
m = 5  
p = 0.95  
  
dict\_p99 = {2: 1.73, 6: 2.16, 8: 2.43, 10: 2.62, 12: 2.75, 15: 2.9, 20: 3.08}  
dict\_p98 = {2: 1.72, 6: 2.13, 8: 2.37, 10: 2.54, 12: 2.66, 15: 2.8, 20: 2.96}  
dict\_p95 = {2: 1.71, 6: 2.10, 8: 2.27, 10: 2.41, 12: 2.52, 15: 2.64, 20: 2.78}  
dict\_p90 = {2: 1.69, 6: 2, 8: 2.17, 10: 2.29, 12: 2.39, 15: 2.49, 20: 2.62}  
  
y\_max = (30 - 17) \* 10  
y\_min = (20 - 17) \* 10  
  
matrix\_x = [[-1, -1],  
 [1, -1],  
 [-1, 1]]  
  
while True:  
 matrix = generate\_matrix()  
 sum\_y = sum\_y(matrix)  
 average = find\_average\_y(sum\_y)  
 dispersion = find\_dispersion(matrix, average)  
 deviation = find\_deviation()  
 Fuv = find\_Fuv(dispersion)  
 dispersion\_uv = find\_dispersion\_uv(Fuv)  
 Ruv = find\_Ruv(dispersion\_uv, deviation)  
 if p <= 0.925:  
 Rkr = find\_Rkr(dict\_p90)  
 elif 0.925 < p == 0.975:  
 Rkr = find\_Rkr(dict\_p95)  
 elif 0.975 < p <= 0.985:  
 Rkr = find\_Rkr(dict\_p98)  
 else:  
 Rkr = find\_Rkr(dict\_p99)  
 if Ruv[0] < Rkr and Ruv[1] < Rkr and Ruv[2] < Rkr:  
 break  
 else:  
 m += 1  
mx\_1, mx\_2, a1, a2, a3, a11, a22 = 0, 0, 0, 0, 0, 0, 0  
  
for i in range(len(matrix\_x)):  
 mx\_1 += matrix\_x[i][0]  
 mx\_2 += matrix\_x[i][1]  
 a1 += matrix\_x[i][0] \*\* 2  
 a2 += matrix\_x[i][0] \* matrix\_x[i][1]  
 a3 += matrix\_x[i][1] \*\* 2  
 a11 += matrix\_x[i][0] \* average[i]  
 a22 += matrix\_x[i][1] \* average[i]  
  
mx\_1 = mx\_1 / len(matrix\_x)  
mx\_2 = mx\_2 / len(matrix\_x)  
a1 = a1 / len(matrix\_x)  
a2 = a2 / len(matrix\_x)  
a3 = a3 / len(matrix\_x)  
a11 = a11 / len(matrix\_x)  
a22 = a22 / len(matrix\_x)  
my = sum(average) / len(average)  
  
b0\_numerator = [[my, mx\_1, mx\_2], [a11, a1, a2], [a22, a2, a3]]  
b012\_denominator = [[1, mx\_1, mx\_2], [mx\_1, a1, a2], [mx\_2, a2, a3]]  
b1\_numerator = [[1, my, mx\_2], [mx\_1, a11, a2], [mx\_2, a22, a3]]  
b2\_numerator = [[1, mx\_1, my], [mx\_1, a1, a11], [mx\_2, a2, a22]]  
  
b0 = det(b0\_numerator) / det(b012\_denominator)  
b1 = det(b1\_numerator) / det(b012\_denominator)  
b2 = det(b2\_numerator) / det(b012\_denominator)  
  
delta\_x1 = delta\_x(x1\_min, x1\_max)  
delta\_x2 = delta\_x(x2\_min, x2\_max)  
x10 = (x1\_max + x1\_min) / 2  
x20 = (x2\_max + x2\_min) / 2  
  
a0 = b0 - b1 \* (x10 / delta\_x1) - b2 \* (x20 / delta\_x2)  
a1 = b1 / delta\_x1  
a2 = b2 / delta\_x2  
  
# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_PRINTING OUT TIME\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ #  
  
print("(ಠ‿ಠ)﻿"\*8)  
print("—" \* 50)  
print("Матриця з у-ків")  
for i in range(len(matrix)):  
 print("\t|", end=" ")  
 for j in range(len(matrix[i])):  
 print(matrix[i][j], end=" ")  
 print("|")  
print("—" \* 50)  
  
print("Середні значення ȳi:")  
for i in range(3):  
 print("\tȳ"+str(i)+" = {:.3f}".format(average[i]))  
print("—" \* 50)  
  
print("Дисперсії по рядках:")  
for i in range(3):  
 print("\tσ{y"+str(i)+"}" + " = {:.3f}".format(dispersion[i]))  
print("—" \* 50)  
  
print("Ruv:")  
for i in range(3):  
 print("\tRuv"+str(i)+" = {:.3f}".format(Ruv[0]))  
print("—" \* 50)  
  
print("Нормоване рівняння регресії:\n\tŷ = {:.3f} + {:.3f}\*X1 + {:.3f}\*X2".format(b0, b1, b2))  
print("\nПеревірка:\n\t{:.3f} - {:.3f} - {:.3f} = {:.3f}".format(b0, b1, b2, b0 - b1 - b2))  
print("\t{:.3f} + {:.3f} - {:.3f} = {:.3f}".format(b0, b1, b2, b0 + b1 - b2))  
print("\t{:.3f} - {:.3f} + {:.3f} = {:.3f}".format(b0, b1, b2, b0 - b1 + b2))  
print("\nРезультат збігається середніми значеннями ȳi.")  
print("—" \* 50)  
  
print("Натуралізоване рівняння регресії:\n\tŷ = {:.3f} + {:.3f}\*X1 + {:.3f}\*X2".format(a0, a1, a2))  
print("\nЗробимо перевірку по рядкам:\n\t{:.3f} + {:.3f}\*{:.3f} + {:.3f}\*{:.3f} = "  
 "{:.3f}".format(a0, a1, x1\_min, a2, x2\_max, a0 + a1 \* x1\_min + a2 \* x2\_min))  
print("\t{:.3f} + {:.3f}\*{:.3f} + {:.3f}\*{:.3f} = {:.3f}".format(a0, a1, x1\_max, a2, x2\_min,  
 a0 + a1 \* x1\_max + a2 \* x2\_min))  
print("\t{:.3f} + {:.3f}\*{:.3f} + {:.3f}\*{:.3f} = {:.3f}".format(a0, a1, x1\_min, a2, x2\_max,  
 a0 + a1 \* x1\_min + a2 \* x2\_max))  
print("\nКоефіцієнти натуралізованого рівняння регресії вірні.")  
print("—" \* 50)  
print("(◕‿◕)♡"\*7)

**Результат:**

(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿(ಠ‿ಠ)﻿

——————————————————————————————————

Матриця з у-ків

| 119 96 125 89 105 |

| 111 127 79 35 34 |

| 100 83 46 94 98 |

——————————————————————————————————

Середні значення ȳi:

ȳ1 = 106.800

——————————————————————————————————

Дисперсії по рядках:

σ{y1} = 229.200

ȳ1 = 77.200

——————————————————————————————————

Дисперсії по рядках:

σ{y1} = 1818.200

ȳ1 = 84.200

——————————————————————————————————

Дисперсії по рядках:

σ{y1} = 499.200

σ{y1} = 229.200

σ{y2} = 1818.200

σ{y3} = 499.200

——————————————————————————————————

Ruv:

Ruv1 = 2.102

Ruv2 = 0.172

Ruv3 = 0.663

——————————————————————————————————

Нормоване рівняння регресії:

ŷ = 80.700 + -14.800\*X1 + -11.300\*X2

Перевірка:

80.700 - -14.800 - -11.300 = 106.800

80.700 + -14.800 - -11.300 = 77.200

80.700 - -14.800 + -11.300 = 84.200

Результат збігається середніми значеннями ȳi.

——————————————————————————————————

Натуралізоване рівняння регресії:

ŷ = 113.167 + -0.493\*X1 + -0.565\*X2

Зробимо перевірку по рядкам:

113.167 + -0.493\*-10.000 + -0.565\*60.000 = 106.800

113.167 + -0.493\*50.000 + -0.565\*20.000 = 77.200

113.167 + -0.493\*-10.000 + -0.565\*60.000 = 84.200

Коефіцієнти натуралізованого рівняння регресії вірні.

——————————————————————————————————

(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡(◕‿◕)♡

Process finished with exit code 0

**Висновок:** Упродовж виконання лабораторної роботи я вивчив основні поняття, визначення, принципи теорії планування експерименту, закріпив отримані знання практичним їх використанням при написанні програми, що реалізує завдання за варіантом. При виконанні роботи проблем не виникло.