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 експерименту» на тему

«ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІО-93
Євтушок Олег
Варіант: 310

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Варіант

№варіанта	x_1		x_2	
	min	max	min	max
310	-25	-5	10	60

Лістинг програми

```
import itertools
import numpy as np
from random import *
import math
from functools import *

'''Варіант 310'''

global y_table
global naturalizedXtable
global a_coeffs

romanovskyCriteriaTable = [[None, 2,    6,    8,    10,   12,   15,   20],
                           [0.99, 1.72, 2.16, 2.43, 2.62, 2.75, 2.90, 3.08],
                           [0.98, 1.72, 2.13, 2.37, 2.54, 2.66, 2.80, 2.96],
                           [0.95, 1.71, 2.10, 2.27, 2.41, 2.52, 2.64, 2.78],
                           [0.90, 1.69, 2.00, 2.17, 2.29, 2.39, 2.49, 2.62]]

xTable = [[-1,-1], [-1,+1], [+1,-1]]

x1Min = -25; x1Max = -5; x2Min = 10; x2Max = 60

yMax = (30 - 310)*10; yMin = (20 - 310)*10

naturalizedXtable = [[x1Min, x2Min], [x1Min, x2Max], [x1Max,  x2Min]]

m = 5


def sigmaT(m):
    return math.sqrt(abs(2 * (2 * m - 2) / (m * (m - 4))))

def romanCriter(y1: np.array, y2: np.array, y3: np.array):

    def Fuv(y_u: np.array, y_v: np.array):
        dev_u = np.var(y_u)
        dev_v = np.var(y_v)
        return dev_u/dev_v if dev_u > dev_v else dev_v/dev_u

    def checkCrit(R, m):
        column = romanovskyCriteriaTable[0].index(sorted(filter(lambda el: el >= m,
romanovskyCriteriaTable[0][1:]))[0])
        trustedProbabilityRow = 1
        return R < romanovskyCriteriaTable[trustedProbabilityRow][column]

    sTheta = sigmaT(m)
    accord = True
    for combination in itertools.combinations((y1,y2,y3), 2):
```

```
        FUv = Fuv(combination[0], combination[1])
        sUV = (m - 2) / m * FUv
        R = abs(sUV - 1) / sTheta
        accord *= checkCrit(R,m)
    return accord

def experiment():
    return np.array([[randint(yMin, yMax) for _ in range(m)] for _ in range(3)])

def normalized_regression_coeffs():
    def m_i(arr: np.array):
        return np.average(arr)

    def a_i(arr: np.array):
        return sum(arr**2)/len(arr)

    def a_jj(arr1: np.array, arr2: np.array):
        return reduce(lambda res, el: res+el[0]*el[1], list(zip(arr1,arr2)),
0)/len(arr1)

    global xTable
    global y_table
    y_vals = np.array([np.average(i) for i in y_table])
    x1_vals = np.array([i[0] for i in xTable])
    x2_vals = np.array([i[1] for i in xTable])
    m_x1 = m_i(x1_vals)
    m_x2 = m_i(x2_vals)
    m_y  = m_i(y_vals)
    a1   = a_i(x1_vals)
    a2   = a_jj(x1_vals, x2_vals)
    a3   = a_i(x2_vals)
    a11  = a_jj(x1_vals, y_vals)
    a22  = a_jj(x2_vals, y_vals)
    coeffs_matrix = [[1, m_x1, m_x2],
                     [m_x1, a1, a2],
                     [m_x2, a2, a3]]
    vals_matrix = [m_y, a11, a22]
    b_coeffs = list(map(lambda num: round(num, 2), np.linalg.solve(coeffs_matrix,
vals_matrix)))
    return  b_coeffs

def assert_normalized_regression():
    global b_coeffs
    global xTable
    global y_table
    y_average_experim_vals = np.array([np.average(i) for i in y_table])
    print("\nПеревірка правильності знаходження коефіцієнтів рівняння регресії: ")
    print("Середні експериментальні значення y для матриці планування: " +
          ", ".join(map(str, y_average_experim_vals)))
    y_theoretical = [b_coeffs[0] + xTable[i][0]*b_coeffs[1] +
xTable[i][1]*b_coeffs[2] for i in range(len(xTable))]
    print("Теоретичні значення y для матриці планування: ".ljust(45) + ",
".join(map(str, y_theoretical)))
    for i in range(len(xTable)):
        try:
            assert round(y_theoretical[i], 2) == round(y_average_experim_vals[i],2)
        except:
            print("Результати пошуку коефіцієнтів рівняння регресії неправильні")
            return
    print("Результати пошуку коефіцієнтів рівняння регресії правильні")

def naturalized_regression(b_coeffs: list):
```

```
    x1 = abs(x1Max-x1Min)/2
    x2 = abs(x2Max-x2Min)/2
    x10 = (x1Max+x1Min)/2
    x20 = (x2Max+x2Min)/2
    a0 = b_coeffs[0]-b_coeffs[1]*x10/x1 - b_coeffs[2]*x20/x2
    a1 = b_coeffs[1]/x1
    a2 = b_coeffs[2]/x2
    return [a0, a1, a2]


def assert_naturalized_regression():
    # global y_table
    # global naturalized_x_table
    # global a_coeffs
    y_average_experim_vals = np.array([np.average(i) for i in y_table])
    print("\nПеревірка натуралізації коефіцієнтів рівняння регресії:")
    print("Середні експериментальні значення y для матриці планування: " +
          ", ".join(map(str, y_average_experim_vals)))
    y_theoretical = [a_coeffs[0] + naturalizedXtable[i][0]*a_coeffs[1]+
naturalizedXtable[i][1]*a_coeffs[2] for i in range(len(naturalizedXtable))]
    print("Теоретичні значення y для матриці планування: ".ljust(45) + ", ".join(
        map(str, y_theoretical)))
    for i in range(len(naturalizedXtable)):
        try:
            assert round(y_theoretical[i],2) == round(y_average_experim_vals[i],2)
        except:
            print("Результати натуралізації неправильні")
            return
    print("Результати натуралізації правильні")

y_table = experiment()

while not romanCriter(*y_table):
    m += 1
    y_table = experiment()

labels_table = ["x1", "x2"] + ["y{}".format(i+1) for i in range(m)]
rows_table = [naturalizedXtable[i] + list(y_table[i]) for i in range(3)]
rows_normalized_table = [xTable[i] + list(y_table[i]) for i in range(3)]

print("Матриця планування:")
print((" "*4).join(labels_table))
print("\n".join([" ".join(map(lambda j: "{:<+5}".format(j), rows_table[i])) for i in
range(len(rows_table))]))
print("\t")

print("Нормована матриця планування:")
print((" "*4).join(labels_table))
print("\n".join([" ".join(map(lambda j: "{:<+5}".format(j),
rows_normalized_table[i])) for i in range(len(rows_normalized_table))]))
print("\t")

b_coeffs = normalized_regression_coeffs()
print("Рівняння регресії для нормованих факторів: y = {0} {1:+}*x1
{2:+}*x2".format(*b_coeffs))
assert_normalized_regression()
a_coeffs = naturalized_regression(b_coeffs)
print("\nРівняння регресії для натуралізованих факторів: y = {0} {1:+}*x1
{2:+}*x2".format(*a_coeffs))
assert_naturalized_regression()
```

Скріншот результату виконання роботи

```
x1    x2    y1    y2    y3    y4    y5
-25   +10   -2818 -2889 -2887 -2890 -2883
-25   +60   -2855 -2885 -2856 -2830 -2887
-5    +10   -2837 -2814 -2801 -2806 -2848

Нормована матриця планування:
x1    x2    y1    y2    y3    y4    y5
-1    -1    -2818 -2889 -2887 -2890 -2883
-1    +1    -2855 -2885 -2856 -2830 -2887
+1    -1    -2837 -2814 -2801 -2806 -2848

Рівняння регресії для нормованих факторів: y = -2841.9 +26.1*x1 +5.4*x2

Перевірка правильності знаходження коефіцієнтів рівняння регресії:
Середні експериментальні значення y для матриці планування: -2873.4, -2862.6, -2821.2
Теоретичні значення y для матриці планування: -2873.4, -2862.6, -2821.2000000000003
Результати пошуку коефіцієнтів рівняння регресії правильні

Рівняння регресії для натуралізованих факторів: y = -2810.31 +2.6100000000000003*x1 +0.21600000000000003*x2

Перевірка натуралізації коефіцієнтів рівняння регресії:
Середні експериментальні значення y для матриці планування: -2873.4, -2862.6, -2821.2
Теоретичні значення y для матриці планування: -2873.4, -2862.6, -2821.2000000000003
Результати натуралізації правильні
```

Контрольні запитання

1. Що таке регресійні поліноми і де вони застосовуються?

Регресійний поліном – це рівняння регресії виду

$$\hat{y} = b_0 + \sum_{i=1}^{k} b_i x_i + \sum_{i,j=1}^{k} b_{i,j} x_i x_j + \sum_{i=1}^{k} b_{i,i} x_i^2 + \sum_{i,j,n=1}^{k} b_{i,j,k} x_i x_j x_n + \dots$$

використовується в ТПЕ для оцінки результатів вимірів.

2. Визначення однорідності дисперсії.

Однорідність дисперсій – властивість, коли дисперсії вимірювання функцій відгуку є однаковими, або близькими.

3. Що називається повним факторним експериментом?

ПФЕ – експеримент, в якому використовуються всі можливі комбінації рівнів факторів.