МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет України
«Київський Політехнічний Інститут»
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 2-го курсу ФІОТ
групи ІО-82
Шендріков Є.О.

Перевірив:
Регіда П. Г.

Київ – 2020

ВАРІАНТ

225	-25	-5	15	50

ФРАГМЕНТ КОДУ

```
import random
import numpy

def Ruv(m, trust_p, y, mean):
    dispersion = []
    for i in range(len(y)):
        dispersion.append(0)
        for j in range(m):
            dispersion[i] += (y[i][j] - mean[i]) ** 2
            dispersion[i] /= m

    main_deviation = ((2 * (2 * m - 2)) / (m * (m - 4))) ** 0.5
    fuvs = [dispersion[i - 1] / dispersion[i] if dispersion[i - 1] >=
dispersion[i] else dispersion[i] / dispersion[i - 1] for i in range(3)]
    teta = [(m - 2) / m * fuvs[i] for i in range(3)]
    ruvs = [abs(teta[i] - 1) / main_deviation for i in range(3)]

    return True if (ruvs[0] < trust_p[m]) & (ruvs[1] < trust_p[m]) & (ruvs[2] <
trust_p[m]) else False


def get_normalized_coefficients(x, y_mean):
    mx1 = (x[0][0] + x[1][0] + x[2][0]) / 3
    mx2 = (x[0][1] + x[1][1] + x[2][1]) / 3
    my = sum(y_mean) / 3

    a1 = (x[0][0] ** 2 + x[1][0] ** 2 + x[2][0] ** 2) / 3
    a2 = (x[0][0] * x[0][1] + x[1][0] * x[1][1] + x[2][0] * x[2][1]) / 3
    a3 = (x[0][1] ** 2 + x[1][1] ** 2 + x[2][1] ** 2) / 3

    a11 = (x[0][0] * y_mean[0] + x[1][0] * y_mean[1] + x[2][0] * y_mean[2]) / 3
    a22 = (x[0][1] * y_mean[0] + x[1][1] * y_mean[1] + x[2][1] * y_mean[2]) / 3

    a = numpy.array([[1, mx1, mx2],
                     [mx1, a1, a2],
                     [mx2, a2, a3]])
    c = numpy.array([[my], [a11], [a22]])
    b = numpy.linalg.solve(a, c)
    return b


def get_naturalized_coefficients(x1_min, x1_max, x2_min, x2_max, b):
    dx1 = (x1_max - x1_min) / 2
    dx2 = (x2_max - x2_min) / 2
    x10 = (x1_max + x1_min) / 2

    x20 = (x2_max + x2_min) / 2

    a0 = b[0][0] - b[1][0] * x10 / dx1 - b[2][0] * x20 / dx2
    a1 = b[1][0] / dx1
    a2 = b[2][0] / dx2

    return a0, a1, a2


x1_min, x1_max = -25, -5
x2_min, x2_max = 15, 50
y_min, y_max = (20 - 225) * 10, (30 - 225) * 10
```

```
m = 5

xn = [[-1, -1],
      [-1, 1],
      [1, -1],
      [1, 1]]

x = [[x1_min, x2_min],
     [x1_min, x2_max],
     [x1_max, x2_min],
     [x1_max, x2_max]]

trust_p = {5: 2, 6: 2,
           7: 2.17, 8: 2.17,
           9: 2.29, 10: 2.29,
           11: 2.39, 12: 2.39,
           13: 2.39, 14: 2.49,
           15: 2.49, 16: 2.49,
           17: 2.49, 18: 2.62,
           19: 2.62, 20: 2.62}

while True:
    y = [[random.randint(y_min, y_max) for i in range(m)] for j in range(3)]
    y_mean = [sum(y[i]) / m for i in range(3)]
    if Ruv(m, trust_p, y, y_mean): break
    else: m += 1

b = get_normalized_coefficients(xn, y_mean)
a = get_naturalized_coefficients(x1_min, x1_max, x2_min, x2_max, b)

table_values = ["¬X1", "¬X2"]
for i in range(m): table_values.append("Y" + str(i + 1))
row_format = "|{:^15}" * (len(table_values))
header_separator_format = "+{0:=^15s}" * (len(table_values))
separator_format = "+{0:-^15s}" * (len(table_values))

print("\n\tЛінійне рівняння регресії: y = b0 + b1*¬x1 + b2*¬x2\n\n" +
header_separator_format.format("=") + "+\n" +
      row_format.format(*table_values) + "|\n" +
header_separator_format.format("=") + "+")

for i in range(3):
    print("|{0:^15}|{1:^15}".format(xn[i][0], xn[i][1]), end="")
    for j in range(m):
        print("|{:^15.2f}".format(y[i][j]), end="")
    print("|")

print(separator_format.format("-") + "+\n\n\tНормалізоване рівняння регресії:  "
+
      "y = {0:.2f} + {1:.2f} * ¬x1 + {2:.2f} * ¬x2".format(b[0][0], b[1][0],
b[2][0]) + "\n\tНатуралізоване рівняння регресії: " +
      "y = {0:.2f} + {1:.2f} * x1 + {2:.2f} *x2".format(*a) + "\n\n\tПеревірка
рівнянь:\n")

for i in range(3):
    print("Yc" + str(i + 1) + " = " + "{0:.2f}".format(y_mean[i]) +
          "\n\tb0 + b1 * ¬x1" + str(i + 1) + " + b2 * ¬x2" + str(i + 1) + " = "
+
          "{0:.2f}".format(b[0][0] + b[1][0] * xn[i][0] + b[2][0] * xn[i][1]) +
          "\n\ta0 + a1 * x1" + str(i + 1) + " + a2 * x2" + str(i + 1) + " = " +
          "{0:.2f}".format(a[0] + a[1] * x[i][0] + a[2] * x[i][1]))
```