МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет України
«Київський Політехнічний Інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни «Методи оптимізації і планування експерименту»

на тему“Проведення трьохфакторного експерименту з використанням лінійного рівняння регресії”

Виконав:
студент 2-го курсу ФІОТ
групи ІО-82
Шевчук О. Б.
Номер у списку: 24

Перевірив:
Регіда П. Г.

Київ – 2020

Варіант

224	-20	15	25	45	-20	-15

Лістинг коду програми

```
import numpy as np


def normalise(factors, def_matrx):
    X0 = np.mean(def_matrx, axis=1)
    delta = np.array([(def_matrx[i, 1] - X0[i]) for i in range(len(factors[0]))])
    X_norm = np.array(
        [[round((factors[i, j] - X0[j]) / delta[j], 3) for j in range(len(factors[i]))]
         for i in range(len(factors))])
    return X_norm


def cohran(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))
    if Gp < 0.5598:
        return True
    return False


def student(norm_factors, 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 / (4 * 6))
    beta = np.mean(norm_factors.T * mean_Y, axis=1)
    t = np.abs(beta) / sigma
    return np.where(t > 2.086)


def fisher(Y_matrix, d):
    if d == 4:
        return False
    Sad = 6 / (4 - 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 Fp > 4.4:
        return False
    return True


x1min, x1max = -20, 15
x2min, x2max = 25, 45
x3min, x3max = -20, -15
def_matrx = np.array([[x1min, x1max], [x2min, x2max], [x3min, x3max]])
factors = np.array([np.random.randint(x1min, x1max, size=4), np.random.randint(x2min,
x2max, size=4), np.random.randint(x3min, x3max, size=4)]).T
norm_factors = normalise(factors, def_matrx)
factors = np.insert(factors, 0, 1, axis=1)
norm_factors = np.insert(norm_factors, 0, 1, axis=1)
```

```
Y_matrix = np.random.randint(200 + np.mean(def_matrx, axis=0)[0], 200 +
np.mean(def_matrx, axis=0)[1], size=(4, 6))
mean_Y = np.mean(Y_matrix, axis=1)

if cohran(Y_matrix):
    b_natural = np.linalg.lstsq(factors, mean_Y, rcond=None)[0]
    b_norm = np.linalg.lstsq(norm_factors, mean_Y, rcond=None)[0]
    check1 = np.sum(b_natural * factors, axis=1)
    check2 = np.sum(b_norm * norm_factors, axis=1)
    indexes = student(norm_factors, Y_matrix)
    print("Фактори: \n", factors)
    print("Нормована матриця факторів: \n", norm_factors)
    print("Функції відгуку: \n", Y_matrix)
    print("Середні значення У: ", mean_Y)
    print("Натуралізовані коефіцієнти: ", b_natural)
    print("Нормовані коефіцієнти: ", b_norm)
    print("Перевірка 1: ", check1)
    print("Перевірка 2: ", check2)
    print("Індекси коефіцієнтів, які задовольняють критерію Стьюдента: ",
np.array(indexes)[0])
    print("Критерій Стьюдента: ", np.sum(np.sum(b_natural[indexes] * factors[:, indexes],
axis=1), axis=1))
    if fisher(Y_matrix, np.size(indexes)):
        print("Рівняння регресії є адекватним оригіналу.")
    else:
        print("Рівняння регресії не є адекватним оригіналу.")
else:
    print("Дисперсія неоднорідна!")
```