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

Лабораторна робота №4

з дисципліни «Методи оптимізації і планування експерименту»

на тему“Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії.”

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

Перевірив:
Регіда П. Г.

Київ – 2020

Варіант

224	-30	0	-15	35	-30	-25

Лістинг коду програми

```
import numpy as np
from scipy.stats import t, f
from itertools import product, combinations

np.set_printoptions(formatter={"float_kind": lambda x: '%.2f'%(x)})

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):
    fish = fisher(0.95, N, m, 1)
    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))
    Gt = fish / (fish + (m - 1) - 2)
    return Gp < Gt


def student(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 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))
    beta = np.mean(norm_matrix.T * mean_Y, axis=1)
    t = np.abs(beta) / sigma
    if (m - 1) * N > 32:
        return np.where(t > student(0.95, N, m))
    return np.where(t > student(0.95, N, m))


def fisher(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 fisher_criteria(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 fisher(0.95, N, m, d)
        return Fp < fisher(0.95, N, m, d)
    if N - d > 6:
        return Fp < fisher(0.95, N, m, d)
    return Fp < fisher(0.95, N, m, d)

m = 6
N = 8
x1min, x1max = -30, 0
x2min, x2max = -15, 35
x3min, x3max = -30, -25
def_matrx = np.array([[x1min, x1max], [x2min, x2max], [x3min, x3max]])
norm_factors = np.array(list(product("01", repeat=3)), dtype=np.int)
norm_factors[norm_factors == 0] = -1
norm_factors = np.insert(norm_factors, 0, 1, axis=1)
factors = np.empty((8, 3))
for i in range(len(norm_factors)):
    for j in range(1, len(norm_factors[i])):
        if j == 1:
            if norm_factors[i, j] == -1:
                factors[i, j-1] = 10
            elif norm_factors[i, j] == 1:
                factors[i, j-1] = 60
        elif j == 2:
            if norm_factors[i, j] == -1:
                factors[i, j-1] = -35
            elif norm_factors[i, j] == 1:
                factors[i, j-1] = 15
        elif j == 3:
            if norm_factors[i, j] == -1:
                factors[i, j-1] = 10
            elif norm_factors[i, j] == 1:
                factors[i, j-1] = 15
factors = np.insert(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=(N, m))
mean_Y = np.mean(Y_matrix, axis=1)
combination = list(combinations(range(1, 4), 2))
for i in combination:
    factors = np.append(factors, np.reshape(factors[:, i[0]]*factors[:, i[1]], (8, 1)), axis=1)
    norm_factors = np.append(norm_factors, np.reshape(norm_factors[:, i[0]]*norm_factors[:, i[1]], (8, 1)), axis=1)
factors = np.append(factors, np.reshape(factors[:, 1]*factors[:, 2]*factors[:, 3], (8, 1)), axis=1)
norm_factors = np.append(norm_factors, np.reshape(norm_factors[:, 1]*norm_factors[:, 2]*norm_factors[:, 3], (8,
1)), 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 = students_t_test(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(b_natural[indexes] * np.reshape(factors[:, indexes], (N,
np.size(indexes))), axis=1))
    if fisher_criteria(Y_matrix, np.size(indexes)):
        print("Рівняння регресії є адекватним оригіналу.")
    else:
        print("Рівняння регресії не є адекватним оригіналу.")
else:
    print("Дисперсія неоднорідна!")
```