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 № 4

з дисципліни «Методи оптимізації та планування експерименту»

на тему
«ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІВ-81
Федорусов Іван
Варіант: 127

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2020

Мета:

Провести дробовий трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту

Варіант:

№варианта	X_1		X_2		X_3	
	min	max	min	max	min	max
127	10	60	15	50	15	20

Лістинг програми:

```
import math
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from functools import partial
from random import randint
from prettytable import PrettyTable

def cochrane(g_prac, g_teor):
    return g_prac < g_teor
def student(t_teor, t_pr):
    return t_pr < t_teor
def fischer(f_teor, f_prac):
    return f_teor > f_prac
def cochrane_teor(f1, f2, q=0.05):
    q1 = q / f1
    fischer_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    return fischer_value / (fischer_value + f1 - 1)

fischer_teor = partial(f.ppf, q=1 - 0.05)
student_teor = partial(t.ppf, q=1 - 0.025)
# min, max
x_1 = [10, 60]
x_2 = [15, 50]
x_3 = [15, 20]
x_av = [(x_1[0] + x_2[0] + x_3[0]) / 3, (x_1[1] + x_2[1] + x_3[1]) / 3]
y = [round(200 + x_av[0]), round(200 + x_av[1])]

while True:
    x0_f = [1, 1, 1, 1, 1, 1, 1, 1]
    x1_f = [-1, -1, 1, 1, -1, -1, 1, 1]
    x2_f = [-1, 1, -1, 1, -1, 1, -1, 1]
    x3_f = [-1, 1, 1, -1, 1, -1, -1, 1]
    x1x2_f = [a * b for a, b in zip(x1_f, x2_f)]
    x1x3_f = [a * b for a, b in zip(x1_f, x3_f)]
    x2x3_f = [a * b for a, b in zip(x2_f, x3_f)]
    x1x2x3_f = [a * b * c for a, b, c in zip(x1_f, x2_f, x3_f)]
    factors = [x0_f, x1_f, x2_f, x3_f, x1x2_f, x1x3_f, x2x3_f, x1x2x3_f]
    m = 3
    y1, y2, y3 = [randint(y[0], y[1]) for i in range(8)], [randint(y[0], y[1]) for i in
range(8)], [randint(y[0], y[1]) for i in range(8)]
    y_dict = {}
    for x in range(1, 9):
        y_dict["y_row{0}".format(x)] = [y1[x - 1], y2[x - 1], y3[x - 1]]
    y_avg_dict = {}
    for x in range(1, 9):
```

```
        y_avg_dict["y_avg{0}".format(x)] = np.average(y_dict[f'y_row{x}'])
    Y_average = [round(val, 3) for val in y_avg_dict.values()]
    x0 = [1, 1, 1, 1, 1, 1, 1, 1]
    x1 = [x_1[0], x_1[0], x_1[1], x_1[1], x_1[0], x_1[0], x_1[1], x_1[1]]
    x2 = [x_2[0], x_2[1], x_2[0], x_2[1], x_2[0], x_2[1], x_2[0], x_2[1]]
    x3 = [x_3[0], x_3[1], x_3[1], x_3[0], x_3[1], x_3[0], x_3[0], x_3[1]]
    x1x2 = [a * b for a, b in zip(x1, x2)]
    x1x3 = [a * b for a, b in zip(x1, x3)]
    x2x3 = [a * b for a, b in zip(x2, x3)]
    x1x2x3 = [a * b * c for a, b, c in zip(x1, x2, x3)]
    x_arr = [x0, x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3]
    b_list = factors
    a_list = list(zip(x0, x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3))
    N = 8
    list_bi = []
    for k in range(N):
        S = 0
        for i in range(N):
            S += (b_list[k][i] * Y_average[i]) / N
        list_bi.append(round(S, 5))
    disp_dict = {}
    for x in range(1, 9):
        disp_dict["disp{0}".format(x)] = 0
    for i in range(m):
        ctr = 1
        for key, value in disp_dict.items():
            row = y_dict[f'y_row{ctr}']
            disp_dict[key] += ((row[i] - np.average(row)) ** 2) / m
            ctr += 1
    disp_sum = sum(disp_dict.values())
    disp_list = [round(disp, 3) for disp in disp_dict.values()]
    column_names = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3", "Y1", "Y2", "Y3",
"Y", "S^2"]
    pt = PrettyTable()
    factors.extend([y1, y2, y3, Y_average, disp_list])
    for k in range(len(factors)):
        pt.add_column(column_names[k], factors[k])
    print(pt, "\n")
    print("y = {} + {} * X1 + {} * X2 + {} * X3 + {} * X1X2 + {} * X1X3 + {} * X2X3 + {} *
X1X2X3 \n".format(list_bi[0], list_bi[1], list_bi[2], list_bi[3], list_bi[4], list_bi[5],
list_bi[6], list_bi[7]))
    pt = PrettyTable()
    x_arr.extend([y1, y2, y3, Y_average, disp_list])
    for k in range(len(factors)):
        pt.add_column(column_names[k], x_arr[k])
    print(pt, "\n")
    list_ai = [round(i, 5) for i in solve(a_list, Y_average)]
    print("y = {} + {} * X1 + {} * X2 + {} * X3 + {} * X1X2 + {} * X1X3 + {} * X2X3 + {} *
X1X2X3".format(list_ai[0], list_ai[1], list_ai[2], list_ai[3], list_ai[4], list_ai[5],
list_ai[6], list_ai[7]))
    Gp = max(disp_dict.values()) / disp_sum
    F1 = m - 1
    N = len(y1)
    F2 = N
    Gt = cochrane_teor(F1, F2)
    print("\nGp = ", Gp, " Gt = ", Gt)
    if cochrane(Gp, Gt):
        print("Дисперсія однорідна!\n")
        Dispersion_B = disp_sum / N
        Dispersion_beta = Dispersion_B / (m * N)
        S_beta = math.sqrt(abs(Dispersion_beta))
        beta_dict = {}
        for x in range(8):
            beta_dict["beta{0}".format(x)] = 0
        for i in range(len(x0_f)):
```

```
            ctr = 0
            for key, value in beta_dict.items():
                beta_dict[key] += (Y_average[i] * factors[ctr][i]) / N
                ctr += 1
        beta_list = list(beta_dict.values())
        t_list = [abs(beta) / S_beta for beta in beta_list]
        F3 = F1 * F2
        d = 0
        T = student_teor(df=F3)
        print("t table = ", T)
        for i in range(len(t_list)):
            if student(t_list[i], T):
                beta_list[i] = 0
                print("Гіпотеза підтверджена, бета {} = 0".format(i))
            else:
                print("Гіпотеза НЕ підтверджена, бета {} = {}".format(i, beta_list[i]))
                d += 1
        factors[0] = None
        y_student = [sum([a * b[x_idx] if b else a for a, b in zip(beta_list, x_arr)]) for
x_idx in range(8)]
        F4 = N - d
        Dispersion_ad = 0
        for i in range(len(y_student)): Dispersion_ad += ((y_student[i] - Y_average[i]) ** 2) *
m / (N - d)
        Fp = Dispersion_ad / Dispersion_beta
        Ft = fischer_teor(dfn=F4, dfd=F3)
        print("Рівняння регресії є адекватним.") if fischer(Ft, Fp) else print("Рівняння
регресії неадекватне.")
        break
    else:
        print("Дисперсія не є однорідною.")
        m += 1
```