НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Дисципліна:**

«Методи наукових досліджень»

**Звіт**

Лабораторної роботи №4

**на тему:** «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії»

**Виконав:**

Студент групи: ІВ-92

Кубишка Юрій Сергійович

Варіант №13

**Перевірив:**

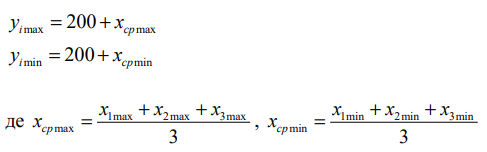
Регіда Павло Геннадійович

**Київ-2021**

**Мета:** провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об’єкту.

**Завдання на лабораторну роботу:**

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

****

1. Знайти коефіцієнти рівняння регресії і записати його.
2. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
3. Написати комп'ютерну програму, яка усе це моделює.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | x1 | | x2 | | x3 | |
|  | min | max | min | max | min | max |
| **213** | -40 | 20 | 5 | 40 | -40 | -20 |

**Код програми:**

import math  
import random  
from functools import partial  
  
import numpy as np  
from scipy.stats import f, t  
  
  
def criterion\_of\_Student(value, criterion, check):  
 if check < criterion:  
 return 0  
 else:  
 return value  
  
  
def cochren\_criterion(f1, f2, q):  
 q1 = q / f1  
 fisher\_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) \* f2)  
 return fisher\_value / (fisher\_value + f1 - 1)  
  
  
m = 3  
N = 8  
x1min, x2min, x3min = -40, 5, -40  
x1max, x2max, x3max = 20, 40, -20  
  
X\_max = [x1max, x2max, x3max]  
X\_min = [x1min, x2min, x3min]  
  
x\_av\_min = (x1min + x2min + x3min) / 3  
x\_av\_max = (x1max + x2max + x3max) / 3  
Y\_max = int(round(200 + x\_av\_max, 0))  
Y\_min = int(round(200 + x\_av\_min, 0))  
X0 = 1  
  
X\_matr = [[-1, -1, -1],  
 [-1, -1, 1],  
 [-1, 1, -1],  
 [-1, 1, 1],  
 [1, -1, -1],  
 [1, -1, 1],  
 [1, 1, -1],  
 [1, 1, 1]]  
  
x\_for\_beta = [[1, -1, -1, -1],  
 [1, -1, -1, 1],  
 [1, -1, 1, -1],  
 [1, -1, 1, 1],  
 [1, 1, -1, -1],  
 [1, 1, -1, 1],  
 [1, 1, 1, -1],  
 [1, 1, 1, 1]]  
  
x\_12\_13\_23 = [[1, 1, 1],  
 [1, -1, -1],  
 [-1, 1, -1],  
 [-1, -1, 1],  
 [-1, -1, 1],  
 [-1, 1, -1],  
 [1, -1, -1],  
 [1, 1, 1]]  
  
X\_matr\_natur = [[-40, 5, -40],  
 [-40, 5, -20],  
 [-40, 40, -40],  
 [-40, 40, -20],  
 [20, 5, -40],  
 [20, 5, -20],  
 [20, 40, -40],  
 [20, 40, -20]]  
  
x\_123 = [-1, 1, 1, -1, 1, -1, -1, 1]  
  
x\_12\_13\_23\_natur = [[X\_matr\_natur[j][0] \* X\_matr\_natur[j][1], X\_matr\_natur[j][0] \* X\_matr\_natur[j][2],  
 X\_matr\_natur[j][1] \* X\_matr\_natur[j][2]] for j in range(N)]  
x\_123\_natur = [X\_matr\_natur[j][0] \* X\_matr\_natur[j][1] \* X\_matr\_natur[j][2] for j in range(N)]  
  
flag = False  
while not flag:  
 Y\_matr = [[random.randint((Y\_min), (Y\_max)) for i in range(m)] for j in range(N)]  
 Y\_average = [sum(j) / m for j in Y\_matr]  
  
 results\_nat = [sum(Y\_average),  
 sum([Y\_average[j] \* X\_matr\_natur[j][0] for j in range(N)]),  
 sum([Y\_average[j] \* X\_matr\_natur[j][1] for j in range(N)]),  
 sum([Y\_average[j] \* X\_matr\_natur[j][2] for j in range(N)]),  
 sum([Y\_average[j] \* x\_12\_13\_23\_natur[j][0] for j in range(N)]),  
 sum([Y\_average[j] \* x\_12\_13\_23\_natur[j][1] for j in range(N)]),  
 sum([Y\_average[j] \* x\_12\_13\_23\_natur[j][2] for j in range(N)]),  
 sum([Y\_average[j] \* x\_123\_natur[j] for j in range(N)])]  
  
 mj0 = [N,  
 sum([X\_matr\_natur[j][0] for j in range(N)]),  
 sum([X\_matr\_natur[j][1] for j in range(N)]),  
 sum([X\_matr\_natur[j][2] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][0] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][1] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][2] for j in range(N)]),  
 sum([x\_123\_natur[j] for j in range(N)])]  
 mj1 = [sum([X\_matr\_natur[j][0] for j in range(N)]),  
 sum([X\_matr\_natur[j][0] \*\* 2 for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][0] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][1] for j in range(N)]),  
 sum([(X\_matr\_natur[j][0] \*\* 2) \* X\_matr\_natur[j][1] for j in range(N)]),  
 sum([(X\_matr\_natur[j][0] \*\* 2) \* X\_matr\_natur[j][2] for j in range(N)]),  
 sum([x\_123\_natur[j] for j in range(N)]),  
 sum([(X\_matr\_natur[j][0] \*\* 2) \* x\_12\_13\_23\_natur[j][2] for j in range(N)])]  
 mj2 = [sum([X\_matr\_natur[j][1] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][0] for j in range(N)]),  
 sum([X\_matr\_natur[j][1] \*\* 2 for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][2] for j in range(N)]),  
 sum([(X\_matr\_natur[j][1] \*\* 2) \* X\_matr\_natur[j][0] for j in range(N)]),  
 sum([x\_123\_natur[j] for j in range(N)]),  
 sum([(X\_matr\_natur[j][1] \*\* 2) \* X\_matr\_natur[j][2] for j in range(N)]),  
 sum([(X\_matr\_natur[j][1] \*\* 2) \* x\_12\_13\_23\_natur[j][1] for j in range(N)])]  
 mj3 = [sum([X\_matr\_natur[j][2] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][1] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][2] for j in range(N)]),  
 sum([X\_matr\_natur[j][2] \*\* 2 for j in range(N)]),  
 sum([x\_123\_natur[j] for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* X\_matr\_natur[j][0] for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* X\_matr\_natur[j][1] for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* x\_12\_13\_23\_natur[j][0] for j in range(N)])]  
 mj4 = [sum([x\_12\_13\_23\_natur[j][0] for j in range(N)]),  
 sum([(X\_matr\_natur[j][0] \*\* 2) \* X\_matr\_natur[j][1] for j in range(N)]),  
 sum([(X\_matr\_natur[j][1] \*\* 2) \* X\_matr\_natur[j][0] for j in range(N)]),  
 sum([x\_123\_natur[j] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][0] \*\* 2 for j in range(N)]),  
 sum([(X\_matr\_natur[j][0] \*\* 2) \* x\_12\_13\_23\_natur[j][2] for j in range(N)]),  
 sum([(X\_matr\_natur[j][1] \*\* 2) \* x\_12\_13\_23\_natur[j][1] for j in range(N)]),  
 sum([(x\_12\_13\_23\_natur[j][0] \*\* 2) \* X\_matr\_natur[j][2] for j in range(N)])]  
 mj5 = [sum([x\_12\_13\_23\_natur[j][1] for j in range(N)]),  
 sum([(X\_matr\_natur[j][0] \*\* 2) \* X\_matr\_natur[j][2] for j in range(N)]),  
 sum([x\_123\_natur[j] for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* X\_matr\_natur[j][0] for j in range(N)]),  
 sum([(X\_matr\_natur[j][0] \*\* 2) \* x\_12\_13\_23\_natur[j][2] for j in range(N)]),  
 sum([x\_12\_13\_23\_natur[j][1] \*\* 2 for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* x\_12\_13\_23\_natur[j][0] for j in range(N)]),  
 sum([(x\_12\_13\_23\_natur[j][1] \*\* 2) \* X\_matr\_natur[j][1] for j in range(N)])]  
 mj6 = [sum([x\_12\_13\_23\_natur[j][2] for j in range(N)]),  
 sum([x\_123\_natur[j] for j in range(N)]),  
 sum([(X\_matr\_natur[j][1] \*\* 2) \* X\_matr\_natur[j][2] for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* X\_matr\_natur[j][1] for j in range(N)]),  
 sum([(X\_matr\_natur[j][1] \*\* 2) \* x\_12\_13\_23\_natur[j][1] for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* x\_12\_13\_23\_natur[j][0] for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* X\_matr\_natur[j][1] for j in range(N)]),  
 sum([(x\_12\_13\_23\_natur[j][2] \*\* 2) \* X\_matr\_natur[j][0] for j in range(N)])]  
 mj7 = [sum([x\_123\_natur[j] for j in range(N)]),  
 sum([(X\_matr\_natur[j][0] \*\* 2) \* x\_12\_13\_23\_natur[j][2] for j in range(N)]),  
 sum([(X\_matr\_natur[j][1] \*\* 2) \* x\_12\_13\_23\_natur[j][1] for j in range(N)]),  
 sum([(X\_matr\_natur[j][2] \*\* 2) \* x\_12\_13\_23\_natur[j][0] for j in range(N)]),  
 sum([(x\_12\_13\_23\_natur[j][0] \*\* 2) \* X\_matr\_natur[j][2] for j in range(N)]),  
 sum([(x\_12\_13\_23\_natur[j][1] \*\* 2) \* X\_matr\_natur[j][1] for j in range(N)]),  
 sum([(x\_12\_13\_23\_natur[j][2] \*\* 2) \* X\_matr\_natur[j][0] for j in range(N)]),  
 sum([x\_123\_natur[j] \*\* 2 for j in range(N)])]  
  
 B\_nat1 = np.linalg.solve([mj0, mj1, mj2, mj3, mj4, mj5, mj6, mj7], results\_nat) # list of B's  
 B\_nat = list(B\_nat1)  
  
 B\_norm = [sum(Y\_average) / N,  
 sum([Y\_average[j] \* X\_matr[j][0] for j in range(N)]) / N,  
 sum([Y\_average[j] \* X\_matr[j][1] for j in range(N)]) / N,  
 sum([Y\_average[j] \* X\_matr[j][2] for j in range(N)]) / N,  
 sum([Y\_average[j] \* x\_12\_13\_23[j][0] for j in range(N)]) / N,  
 sum([Y\_average[j] \* x\_12\_13\_23[j][1] for j in range(N)]) / N,  
 sum([Y\_average[j] \* x\_12\_13\_23[j][2] for j in range(N)]) / N,  
 sum([Y\_average[j] \* x\_123[j] for j in range(N)]) / N]  
 print("Матриця планування експерименту:")  
 print("N " + "x1 " + "x2 " + "x3 " + "Y1" + " " \* 8 + "Y2" + " " \* 8 + "Y3")  
 for i in range(N):  
 print("{0:=d} {1:=4d} {2:=3d} {3:=3d} {4:=9.5f} {5:=9.5f} {6:=9.5f}".format(i + 1, X\_matr[i][0], X\_matr[i][1],  
 X\_matr[i][2], Y\_matr[i][0],  
 Y\_matr[i][1], Y\_matr[i][2]))  
 print('\_' \* 80, '\n')  
  
 y1\_nat = B\_nat[0] + B\_nat[1] \* X\_matr\_natur[0][0] + B\_nat[2] \* \  
 X\_matr\_natur[0][1] + B\_nat[3] \* X\_matr\_natur[0][2] + \  
 B\_nat[4] \* x\_12\_13\_23\_natur[0][0] + B\_nat[5] \* x\_12\_13\_23\_natur[0][1] + \  
 B\_nat[6] \* x\_12\_13\_23\_natur[0][2] + B\_nat[7] \* x\_123\_natur[0]  
  
 y1\_norm = B\_norm[0] + B\_norm[1] \* X\_matr[0][0] + B\_norm[2] \* \  
 X\_matr[0][1] + B\_norm[3] \* X\_matr[0][2] + B\_norm[4] \* \  
 x\_12\_13\_23[0][0] + B\_norm[5] \* x\_12\_13\_23[0][1] + B\_norm[6] \* \  
 x\_12\_13\_23[0][2] + B\_norm[7] \* x\_123[0]  
  
 dx = [((X\_max[i] - X\_min[i]) / 2) for i in range(3)]  
 A = [sum(Y\_average) / len(Y\_average), B\_nat[0] \* dx[0], B\_nat[1] \* dx[1], B\_nat[2] \* dx[2]]  
  
 S\_kv = [(sum([((Y\_matr[i][j] - Y\_average[i]) \*\* 2) for j in range(m)]) / m) for i in range(N)]  
  
 Gp = max(S\_kv) / sum(S\_kv)  
  
 f1 = m - 1  
 f2 = N  
 p = 0.95  
 q = 1 - p  
 # для N=8  
 Gt\_dict = {2: 5157, 3: 4377, 4: 3910, 5: 3595, 6: 3362, 7: 3185, 8: 3043, 9: 2926, 10: 2829, 16: 2462}  
  
 Gt = cochren\_criterion(f1, f2, q)  
  
 if Gp < Gt:  
 print('Дисперсії однорідні')  
 flag = False  
  
 else:  
 print('Дисперсії неоднорідні')  
 m += 1  
  
 S\_average = sum(S\_kv) / N  
 S2\_beta\_s = S\_average / (N \* m)  
 S\_beta\_s = S2\_beta\_s \*\* .5  
  
 beta = [(sum([x\_for\_beta[j][i] \* Y\_average[j] for j in range(N)]) / N) for i in range(4)]  
 ts = [(math.fabs(beta[i]) / S\_beta\_s) for i in range(4)]  
 tabl\_Stud = [12.71, 4.303, 3.182, 2.776, 2.571, 2.447, 2.365, 2.306, 2.262, 2.228, 2.201, 2.179]  
 f3 = f1 \* f2  
 student = partial(t.ppf, q=1 - 0.025)  
 criterion\_of\_St = student(df=f3)  
  
 result\_2 = [criterion\_of\_Student(B\_nat[0], criterion\_of\_St, ts[0]) +  
 criterion\_of\_Student(B\_nat[1], criterion\_of\_St, ts[1]) \* X\_matr\_natur[i][0] +  
 criterion\_of\_Student(B\_nat[2], criterion\_of\_St, ts[2]) \* X\_matr\_natur[i][1] +  
 criterion\_of\_Student(B\_nat[3], criterion\_of\_St, ts[3]) \* X\_matr\_natur[i][2] for i in range(N)]  
 znach\_koef = []  
  
 for i in ts:  
 if i > criterion\_of\_St:  
 znach\_koef.append(i)  
 else:  
 pass  
  
 d = len(znach\_koef)  
 f4 = N - d  
 f3 = (m - 1) \* N  
  
 deviation\_of\_adequacy = (m / (N - d)) \* sum([(result\_2[i] - Y\_average[i]) \*\* 2 for i in range(N)])  
  
 Fp = deviation\_of\_adequacy / S2\_beta\_s  
  
 fisher = partial(f.ppf, q=1 - 0.05)  
 Ft = fisher(dfn=f4, dfd=f3)  
  
 print("Значення після критерія Стюдента: ")  
 print("Y1 = {0:.3f};\nY2 = {1:.3f};\nY3 = {2:.3f};\nY4 = {3:.3f}.".format(result\_2[0], result\_2[1],  
 result\_2[2], result\_2[3]))  
 print("Y1a = {0:.3f};\nY2a = {1:.3f};\nY3a = {2:.3f};\nY4a = {3:.3f}.".format(Y\_average[0], Y\_average[1],  
 Y\_average[2], Y\_average[3]))  
 if Fp > Ft:  
 print(f'Fp = {round(Fp, 3)} > Ft = {Ft}')  
 print(f'Рівняння регресії неадекватно оригіналу при рівні значимості {format(round(q, 2))}')  
 else:  
 print(f'Fp = {round(Fp, 3)} < Ft = {Ft}')  
 print(f'Рівняння регресії адекватно оригіналу при рівні значимості {round(q, 2)}')  
 flag = True

**Результати роботи програми:**

