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 № 5

з дисципліни «Методи наукових досліджень» на тему

«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням квадратичних членів (центральний ортогональний композиційний план)»

Виконав:

студент ІІ курсу ФІОТ

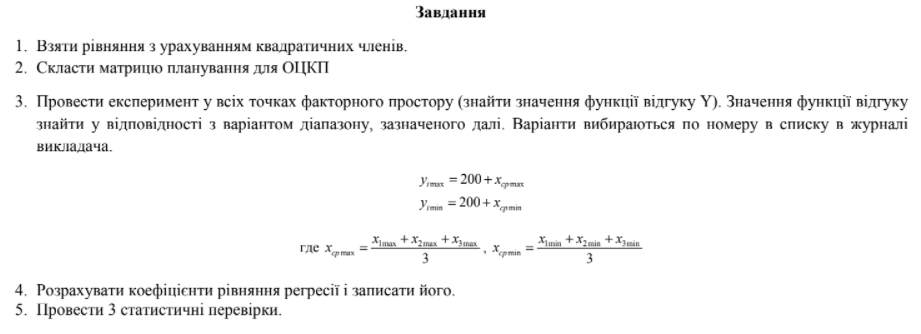
групи ІВ-93

Ільяш В. В.

ПЕРЕВІРИВ:

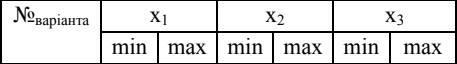
ас. Регіда П. Г.

Київ - 2021



**Варіант:**

Nваріант = 7

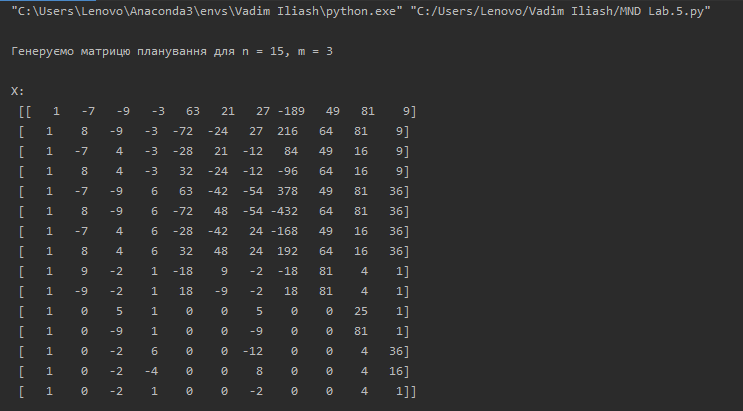


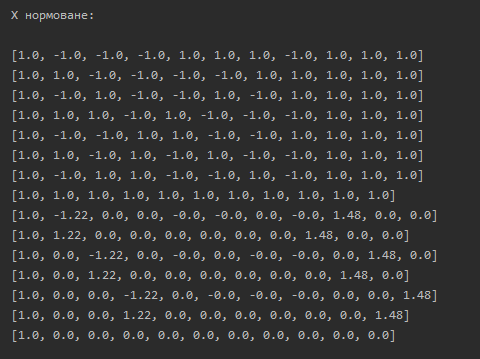


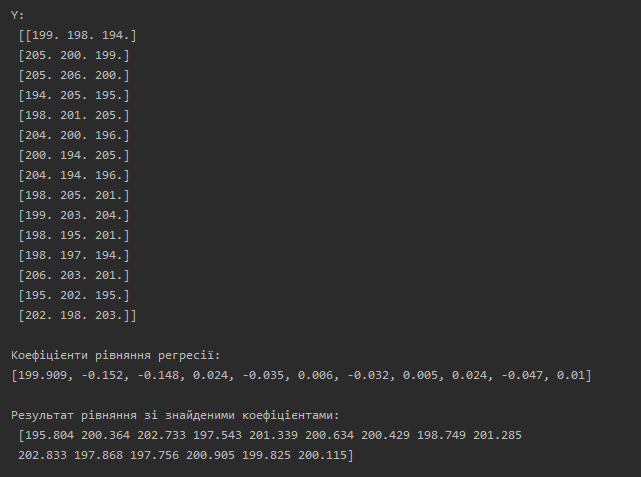
**Роздруківка коду програми:**

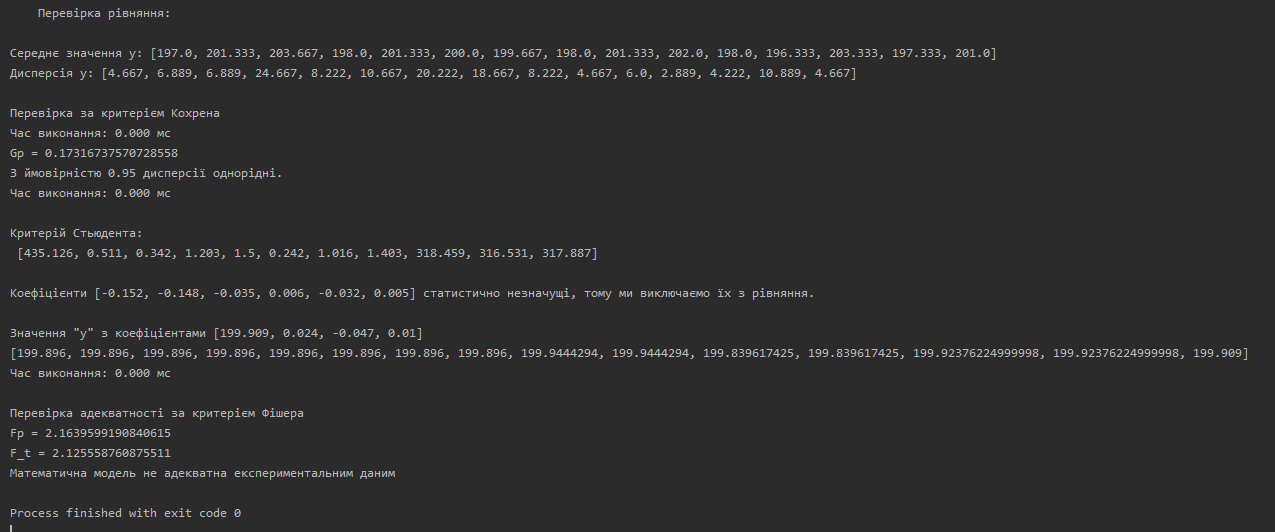
import random  
import sklearn.linear\_model as lm  
from scipy.stats import f, t  
from functools import partial  
from pyDOE2 import \*  
from time import time  
  
  
def timeit(func):  
 def wrapper(\*args, \*\*kwargs):  
 start = time()  
 result = func(\*args, \*\*kwargs)  
 end = (time() - start) \* 1000  
 print(f'Час виконання: {end:.3f} мс')  
 return result  
  
 return wrapper  
  
  
def regression(x, b):  
 y = sum([x[i] \* b[i] for i in range(len(x))])  
 return y  
  
  
x\_range = ((-7, 8), (-9, 4), (-3, 6))  
  
x\_aver\_max = sum([x[1] for x in x\_range]) / 3  
x\_aver\_min = sum([x[0] for x in x\_range]) / 3  
  
y\_max = 200 + int(x\_aver\_max)  
y\_min = 200 + int(x\_aver\_min)  
  
  
def s\_kv(y, y\_aver, n, m):  
 res = []  
 for i in range(n):  
 s = sum([(y\_aver[i] - y[i][j]) \*\* 2 for j in range(m)]) / m  
 res.append(round(s, 3))  
 return res  
  
  
def plan\_matrix5(n, m):  
 print(f'\nГенеруємо матрицю планування для n = {n}, m = {m}')  
  
 y = np.zeros(shape=(n, m))  
 for i in range(n):  
 for j in range(m):  
 y[i][j] = random.randint(y\_min, y\_max)  
  
 if n > 14:  
 no = n - 14  
 else:  
 no = 1  
 x\_norm = ccdesign(3, center=(0, no))  
 x\_norm = np.insert(x\_norm, 0, 1, axis=1)  
  
 for i in range(4, 11):  
 x\_norm = np.insert(x\_norm, i, 0, axis=1)  
  
 l = 1.215  
  
 for i in range(len(x\_norm)):  
 for j in range(len(x\_norm[i])):  
 if x\_norm[i][j] < -1 or x\_norm[i][j] > 1:  
 if x\_norm[i][j] < 0:  
 x\_norm[i][j] = -l  
 else:  
 x\_norm[i][j] = l  
  
 def add\_sq\_nums(x):  
 for i in range(len(x)):  
 x[i][4] = x[i][1] \* x[i][2]  
 x[i][5] = x[i][1] \* x[i][3]  
 x[i][6] = x[i][2] \* x[i][3]  
 x[i][7] = x[i][1] \* x[i][3] \* x[i][2]  
 x[i][8] = x[i][1] \*\* 2  
 x[i][9] = x[i][2] \*\* 2  
 x[i][10] = x[i][3] \*\* 2  
 return x  
  
 x\_norm = add\_sq\_nums(x\_norm)  
  
 x = np.ones(shape=(len(x\_norm), len(x\_norm[0])), dtype=np.int64)  
 for i in range(8):  
 for j in range(1, 4):  
 if x\_norm[i][j] == -1:  
 x[i][j] = x\_range[j - 1][0]  
 else:  
 x[i][j] = x\_range[j - 1][1]  
  
 for i in range(8, len(x)):  
 for j in range(1, 3):  
 x[i][j] = (x\_range[j - 1][0] + x\_range[j - 1][1]) / 2  
  
 dx = [x\_range[i][1] - (x\_range[i][0] + x\_range[i][1]) / 2 for i in range(3)]  
  
 x[8][1] = l \* dx[0] + x[9][1]  
 x[9][1] = -l \* dx[0] + x[9][1]  
 x[10][2] = l \* dx[1] + x[9][2]  
 x[11][2] = -l \* dx[1] + x[9][2]  
 x[12][3] = l \* dx[2] + x[9][3]  
 x[13][3] = -l \* dx[2] + x[9][3]  
  
 x = add\_sq\_nums(x)  
  
 print('\nX:\n', x)  
 print('\nX нормоване:\n')  
 for i in x\_norm:  
 print([round(x, 2) for x in i])  
 print('\nY:\n', y)  
  
 return x, y, x\_norm  
  
  
def find\_coef(X, Y, norm=False):  
 skm = lm.LinearRegression(fit\_intercept=False)  
 skm.fit(X, Y)  
 B = skm.coef\_  
  
 if norm == 1:  
 print('\nКоефіцієнти рівняння регресії з нормованими X:')  
 else:  
 print('\nКоефіцієнти рівняння регресії:')  
 B = [round(i, 3) for i in B]  
 print(B)  
 print('\nРезультат рівняння зі знайденими коефіцієнтами:\n', np.dot(X, B))  
 return B  
  
  
@timeit  
def kriteriy\_cochrana(y, y\_aver, n, m):  
 f1 = m - 1  
 f2 = n  
 q = 0.05  
 S\_kv = s\_kv(y, y\_aver, n, m)  
 Gp = max(S\_kv) / sum(S\_kv)  
 print('\nПеревірка за критерієм Кохрена')  
 return Gp  
  
  
def cohren(f1, f2, q=0.05):  
 q1 = q / f1  
 fisher\_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) \* f2)  
 return fisher\_value / (fisher\_value + f1 - 1)  
  
  
def bs(x, y\_aver, n): # метод для оцінки коефіцієнтів  
 res = [sum(1 \* y for y in y\_aver) / n]  
  
 for i in range(len(x[0])):  
 b = sum(j[0] \* j[1] for j in zip(x[:, i], y\_aver)) / n  
 res.append(b)  
 return res  
  
  
@timeit  
def kriteriy\_studenta(x, y, y\_aver, n, m):  
 S\_kv = s\_kv(y, y\_aver, n, m)  
 s\_kv\_aver = sum(S\_kv) / n  
  
 s\_Bs = (s\_kv\_aver / n / m) \*\* 0.5  
 Bs = bs(x, y\_aver, n)  
 ts = [round(abs(B) / s\_Bs, 3) for B in Bs]  
  
 return ts  
  
  
@timeit  
def kriteriy\_fishera(y, y\_aver, y\_new, n, m, d):  
 S\_ad = m / (n - d) \* sum([(y\_new[i] - y\_aver[i]) \*\* 2 for i in range(len(y))])  
 S\_kv = s\_kv(y, y\_aver, n, m)  
 S\_kv\_aver = sum(S\_kv) / n  
  
 return S\_ad / S\_kv\_aver  
  
  
def check(X, Y, B, n, m):  
 print('\n\tПеревірка рівняння:')  
 f1 = m - 1  
 f2 = n  
 f3 = f1 \* f2  
 q = 0.05  
  
 student = partial(t.ppf, q=1 - q)  
 t\_student = student(df=f3)  
  
 G\_kr = cohren(f1, f2)  
  
 y\_aver = [round(sum(i) / len(i), 3) for i in Y]  
 print('\nСереднє значення y:', y\_aver)  
  
 disp = s\_kv(Y, y\_aver, n, m)  
 print('Дисперсія y:', disp)  
  
 Gp = kriteriy\_cochrana(Y, y\_aver, n, m)  
 print(f'Gp = {Gp}')  
 if Gp < G\_kr:  
 print(f'З ймовірністю {1 - q} дисперсії однорідні.')  
 else:  
 print("Необхідно збільшити кількість дослідів")  
 m += 1  
 main(n, m)  
  
 ts = kriteriy\_studenta(X[:, 1:], Y, y\_aver, n, m)  
 print('\nКритерій Стьюдента:\n', ts)  
 res = [t for t in ts if t > t\_student]  
 final\_k = [B[i] for i in range(len(ts)) if ts[i] in res]  
 print('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з рівняння.'.format(  
 [round(i, 3) for i in B if i not in final\_k]))  
  
 y\_new = []  
 for j in range(n):  
 y\_new.append(regression([X[j][i] for i in range(len(ts)) if ts[i] in res], final\_k))  
  
 print(f'\nЗначення "y" з коефіцієнтами {final\_k}')  
 print(y\_new)  
  
 d = len(res)  
 if d >= n:  
 print('\nF4 <= 0')  
 print('')  
 return  
 f4 = n - d  
  
 F\_p = kriteriy\_fishera(Y, y\_aver, y\_new, n, m, d)  
  
 fisher = partial(f.ppf, q=0.95)  
 f\_t = fisher(dfn=f4, dfd=f3)  
 print('\nПеревірка адекватності за критерієм Фішера')  
 print('Fp =', F\_p)  
 print('F\_t =', f\_t)  
 if F\_p < f\_t:  
 print('Математична модель адекватна експериментальним даним')  
 else:  
 print('Математична модель не адекватна експериментальним даним')  
  
  
def main(n, m):  
 ŅX5, Y5, X5\_norm = plan\_matrix5(n, m)  
  
 y5\_aver = [round(sum(i) / len(i), 3) for i in Y5]  
 B5 = find\_coef(X5, y5\_aver)  
  
 check(X5\_norm, Y5, B5, n, m)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main(15, 3)

**Результат виконання коду:**









**Висновки:**

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії, рівняння регресії з  ефектом взаємодії та рівняння регресії з квадратичними членами, складено матрицю планування експерименту, було визначено коефіцієнти рівнянь регресії (натуралізовані та нормовані), для форми з квадратичними членами - натуралізовані, виконано перевірку правильності розрахунку коефіцієнтів рівнянь регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера) для кожної форми рівняння регресії . При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів, при неадекватності і такого рівняння регресії було затосовано рівняння регресії з квадратичними членами. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості q = 0.05.