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 №6**

З дисципліни «Методи оптимізації та планування»

**Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами**

ВИКОНАВ:

Студент ІІ курсу ФІОТ

Групи ІВ-92

Залога А.С.

ПЕРЕВІРИВ:

Регіда П.Г.

Київ 2021 р.

**Мета:** Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

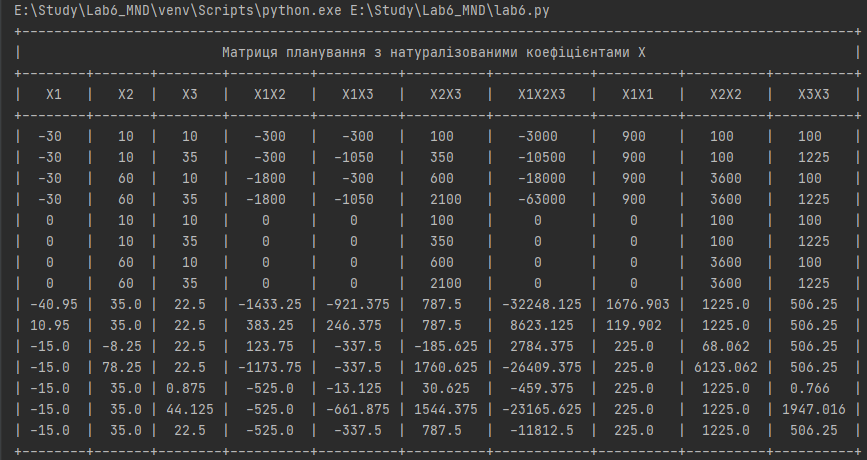
**Варіант завдання:**

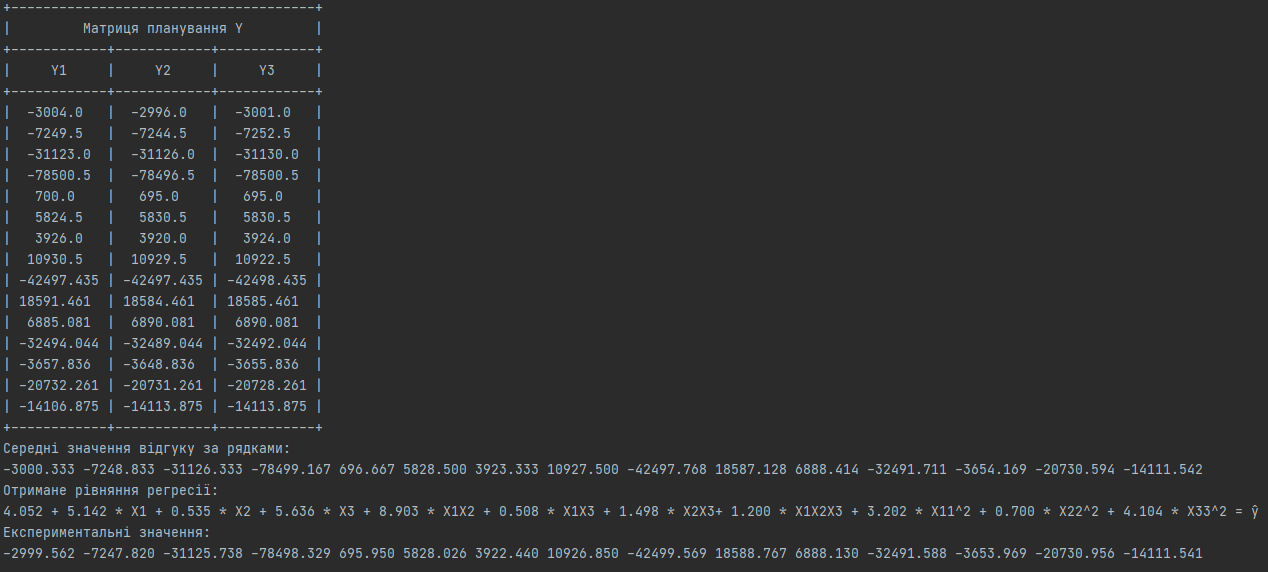
C:\Users\User\Documents\ShareX\Screenshots\2021-05\chrome_r14rkwnWx6.png

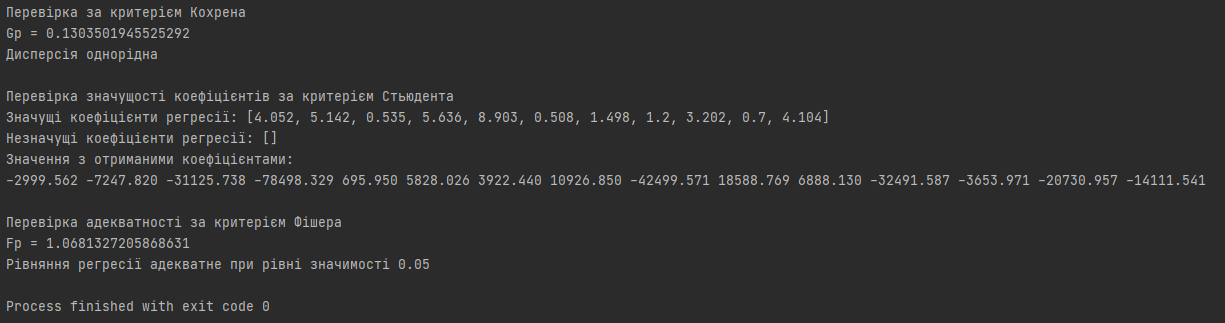
**Лістинг програми:**

from math import fabs  
from random import randrange  
import numpy as np  
from numpy.linalg import solve  
from scipy.stats import f, t  
from prettytable import PrettyTable  
  
  
def round\_matrix(matrix, n\_to\_round=3):  
 for i in range(len(matrix)):  
 matrix[i] = list(matrix[i])  
 for j in range(len(matrix[i])):  
 matrix[i][j] = round(matrix[i][j], n\_to\_round)  
 return matrix  
  
  
def function(X1, X2, X3):  
 y = 8.0 + 5.3 \* X1 + 0.5 \* X2 + 5.6 \* X3 + 3.2 \* X1 \* X1 + 0.7 \* X2 \* X2 + 4.1 \* X3 \* X3 + 8.9 \* X1 \* X2 + \  
 0.5 \* X1 \* X3 + 1.5 \* X2 \* X3 + 1.2 \* X1 \* X2 \* X3 + randrange(0, 10) - 5  
 return y  
  
  
def main(n, m):  
 x1min = -30  
 x1max = 0  
 x2min = 10  
 x2max = 60  
 x3min = 10  
 x3max = 35  
  
 x01 = (x1max + x1min) / 2  
 x02 = (x2max + x2min) / 2  
 x03 = (x3max + x3min) / 2  
 deltax1 = x1max - x01  
 deltax2 = x2max - x02  
 deltax3 = x3max - x03  
  
 xn = [[-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, +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, -1, -1, -1, +1, +1, +1],  
 [+1, +1, +1, +1, +1, +1, +1, +1, +1, +1],  
 [-1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],  
 [+1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],  
 [0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],  
 [0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],  
 [0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929],  
 [0, 0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929],  
 [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]  
  
 x1 = [x1min, x1min, x1min, x1min, x1max, x1max, x1max, x1max, -1.73 \* deltax1 + x01, 1.73 \* deltax1 + x01, x01, x01,  
 x01, x01, x01]  
 x2 = [x2min, x2min, x2max, x2max, x2min, x2min, x2max, x2max, x02, x02, -1.73 \* deltax2 + x02, 1.73 \* deltax2 + x02,  
 x02, x02, x02]  
 x3 = [x3min, x3max, x3min, x3max, x3min, x3max, x3min, x3max, x03, x03, x03, x03, -1.73 \* deltax3 + x03,  
 1.73 \* deltax3 + x03, x03]  
  
 x1x2 = [0] \* 15  
 x1x3 = [0] \* 15  
 x2x3 = [0] \* 15  
 x1x2x3 = [0] \* 15  
 x1kv = [0] \* 15  
 x2kv = [0] \* 15  
 x3kv = [0] \* 15  
  
 for i in range(15):  
 x1x2[i] = x1[i] \* x2[i]  
 x1x3[i] = x1[i] \* x3[i]  
 x2x3[i] = x2[i] \* x3[i]  
 x1x2x3[i] = x1[i] \* x2[i] \* x3[i]  
 x1kv[i] = x1[i] \*\* 2  
 x2kv[i] = x2[i] \*\* 2  
 x3kv[i] = x3[i] \*\* 2  
  
 list\_for\_a = round\_matrix(list(zip(x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3, x1kv, x2kv, x3kv)))  
  
 planning\_matrix\_with\_naturalized\_coeffs\_x = PrettyTable()  
 planning\_matrix\_with\_naturalized\_coeffs\_x.title = 'Матриця планування з натуралізованими коефіцієнтами X'  
 planning\_matrix\_with\_naturalized\_coeffs\_x.field\_names = ['X1', 'X2', 'X3', 'X1X2', 'X1X3', 'X2X3', 'X1X2X3', 'X1X1',  
 'X2X2', 'X3X3']  
 planning\_matrix\_with\_naturalized\_coeffs\_x.add\_rows(list\_for\_a)  
 print(planning\_matrix\_with\_naturalized\_coeffs\_x)  
  
 Y = round\_matrix(  
 [[function(list\_for\_a[j][0], list\_for\_a[j][1], list\_for\_a[j][2]) for i in range(m)] for j in range(15)])  
  
 planning\_matrix\_y = PrettyTable()  
 planning\_matrix\_y.title = 'Матриця планування Y'  
 planning\_matrix\_y.field\_names = ['Y1', 'Y2', 'Y3']  
 planning\_matrix\_y.add\_rows(Y)  
 print(planning\_matrix\_y)  
  
 Y\_average = []  
 for i in range(len(Y)):  
 Y\_average.append(np.mean(Y[i], axis=0))  
 print("Середні значення відгуку за рядками:")  
 for i in range(15):  
 print("{:.3f}".format(Y\_average[i]), end=" ")  
  
 dispersions = []  
 for i in range(len(Y)):  
 a = 0  
 for k in Y[i]:  
 a += (k - np.mean(Y[i], axis=0)) \*\* 2  
 dispersions.append(a / len(Y[i]))  
  
 def find\_known(num):  
 a = 0  
 for j in range(15):  
 a += Y\_average[j] \* list\_for\_a[j][num - 1] / 15  
 return a  
  
 def a(first, second):  
 a = 0  
 for j in range(15):  
 a += list\_for\_a[j][first - 1] \* list\_for\_a[j][second - 1] / 15  
 return a  
  
 my = sum(Y\_average) / 15  
 mx = []  
  
 for i in range(10):  
 number\_lst = []  
 for j in range(15):  
 number\_lst.append(list\_for\_a[j][i])  
 mx.append(sum(number\_lst) / len(number\_lst))  
  
 det1 = [  
 [1, mx[0], mx[1], mx[2], mx[3], mx[4], mx[5], mx[6], mx[7], mx[8], mx[9]],  
 [mx[0], a(1, 1), a(1, 2), a(1, 3), a(1, 4), a(1, 5), a(1, 6), a(1, 7), a(1, 8), a(1, 9), a(1, 10)],  
 [mx[1], a(2, 1), a(2, 2), a(2, 3), a(2, 4), a(2, 5), a(2, 6), a(2, 7), a(2, 8), a(2, 9), a(2, 10)],  
 [mx[2], a(3, 1), a(3, 2), a(3, 3), a(3, 4), a(3, 5), a(3, 6), a(3, 7), a(3, 8), a(3, 9), a(3, 10)],  
 [mx[3], a(4, 1), a(4, 2), a(4, 3), a(4, 4), a(4, 5), a(4, 6), a(4, 7), a(4, 8), a(4, 9), a(4, 10)],  
 [mx[4], a(5, 1), a(5, 2), a(5, 3), a(5, 4), a(5, 5), a(5, 6), a(5, 7), a(5, 8), a(5, 9), a(5, 10)],  
 [mx[5], a(6, 1), a(6, 2), a(6, 3), a(6, 4), a(6, 5), a(6, 6), a(6, 7), a(6, 8), a(6, 9), a(6, 10)],  
 [mx[6], a(7, 1), a(7, 2), a(7, 3), a(7, 4), a(7, 5), a(7, 6), a(7, 7), a(7, 8), a(7, 9), a(7, 10)],  
 [mx[7], a(8, 1), a(8, 2), a(8, 3), a(8, 4), a(8, 5), a(8, 6), a(8, 7), a(8, 8), a(8, 9), a(8, 10)],  
 [mx[8], a(9, 1), a(9, 2), a(9, 3), a(9, 4), a(9, 5), a(9, 6), a(9, 7), a(9, 8), a(9, 9), a(9, 10)],  
 [mx[9], a(10, 1), a(10, 2), a(10, 3), a(10, 4), a(10, 5), a(10, 6), a(10, 7), a(10, 8), a(10, 9), a(10, 10)]]  
  
 det2 = [my, find\_known(1), find\_known(2), find\_known(3), find\_known(4), find\_known(5), find\_known(6), find\_known(7),  
 find\_known(8), find\_known(9), find\_known(10)]  
  
 beta = solve(det1, det2)  
 print("\nОтримане рівняння регресії:")  
 print("{:.3f} + {:.3f} \* X1 + {:.3f} \* X2 + {:.3f} \* X3 + {:.3f} \* Х1X2 + {:.3f} \* Х1X3 + {:.3f} \* Х2X3"  
 "+ {:.3f} \* Х1Х2X3 + {:.3f} \* X11^2 + {:.3f} \* X22^2 + {:.3f} \* X33^2 = ŷ"  
 .format(beta[0], beta[1], beta[2], beta[3], beta[4], beta[5], beta[6], beta[7], beta[8], beta[9], beta[10]))  
 y\_i = [0] \* 15  
 print("Експериментальні значення:")  
 for k in range(15):  
 y\_i[k] = beta[0] + beta[1] \* list\_for\_a[k][0] + beta[2] \* list\_for\_a[k][1] + beta[3] \* list\_for\_a[k][2] + \  
 beta[4] \* list\_for\_a[k][3] + beta[5] \* list\_for\_a[k][4] + beta[6] \* list\_for\_a[k][5] + beta[7] \* \  
 list\_for\_a[k][6] + beta[8] \* list\_for\_a[k][7] + beta[9] \* list\_for\_a[k][8] + beta[10] \* list\_for\_a[k][  
 9]  
 for i in range(15):  
 print("{:.3f}".format(y\_i[i]), end=" ")  
 print("\n\nПеревірка за критерієм Кохрена")  
 Gp = max(dispersions) / sum(dispersions)  
 Gt = 0.3346  
 print("Gp =", Gp)  
 if Gp < Gt:  
 print("Дисперсія однорідна")  
 else:  
 print("Дисперсія неоднорідна")  
  
 print("\nПеревірка значущості коефіцієнтів за критерієм Стьюдента")  
 sb = sum(dispersions) / len(dispersions)  
 sbs = (sb / (15 \* m)) \*\* 0.5  
  
 F3 = (m - 1) \* n  
 coefs1 = []  
 coefs2 = []  
 d = 11  
 res = [0] \* 11  
 for j in range(11):  
 t\_pract = 0  
 for i in range(15):  
 if j == 0:  
 t\_pract += Y\_average[i] / 15  
 else:  
 t\_pract += Y\_average[i] \* xn[i][j - 1]  
 res[j] = beta[j]  
 if fabs(t\_pract / sbs) < t.ppf(q=0.975, df=F3):  
 coefs2.append(beta[j])  
 res[j] = 0  
 d -= 1  
 else:  
 coefs1.append(beta[j])  
 print("Значущі коефіцієнти регресії:", [round(i, 3) for i in coefs1])  
 print("Незначущі коефіцієнти регресії:", [round(i, 3) for i in coefs2])  
 y\_st = []  
 for i in range(15):  
 y\_st.append(res[0] + res[1] \* x1[i] + res[2] \* x2[i] + res[3] \* x3[i] + res[4] \* x1x2[i] + res[5] \*  
 x1x3[i] + res[6] \* x2x3[i] + res[7] \* x1x2x3[i] + res[8] \* x1kv[i] + res[9] \*  
 x2kv[i] + res[10] \* x3kv[i])  
 print("Значення з отриманими коефіцієнтами:")  
 for i in range(15):  
 print("{:.3f}".format(y\_st[i]), end=" ")  
  
 print("\n\nПеревірка адекватності за критерієм Фішера")  
 Sad = m \* sum([(y\_st[i] - Y\_average[i]) \*\* 2 for i in range(15)]) / (n - d)  
 Fp = Sad / sb  
 F4 = n - d  
 print("Fp =", Fp)  
 if Fp < f.ppf(q=0.95, dfn=F4, dfd=F3):  
 print("Рівняння регресії адекватне при рівні значимості 0.05")  
 else:  
 print("Рівняння регресії неадекватне при рівні значимості 0.05")  
  
  
main(15, 3)

**Результати роботи програми:**

****

****

****

**Висновок:**

В даній лабораторній роботі я провів трьохфакторний експеримент і отримав адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.