Міністерство освіти і науки України

Національний технічний університет України «КПІ» імені Ігоря Сікорського

Кафедра обчислювальної техніки ФІОТ

ЗВІТ

з лабораторної роботи №4

з навчальної дисципліни «Методи наукових досліджень»

Тема:

ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ ПРИ ВИКОРИСТАННІ РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ.

Виконав:

Студент групи ІВ-92,

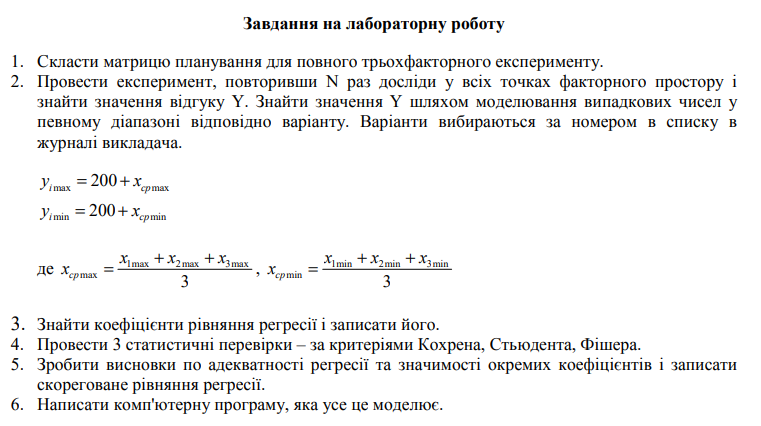
Карпека Дмитрій Юрійович

Перевірив:

Регіда П. Г.

Київ 2021



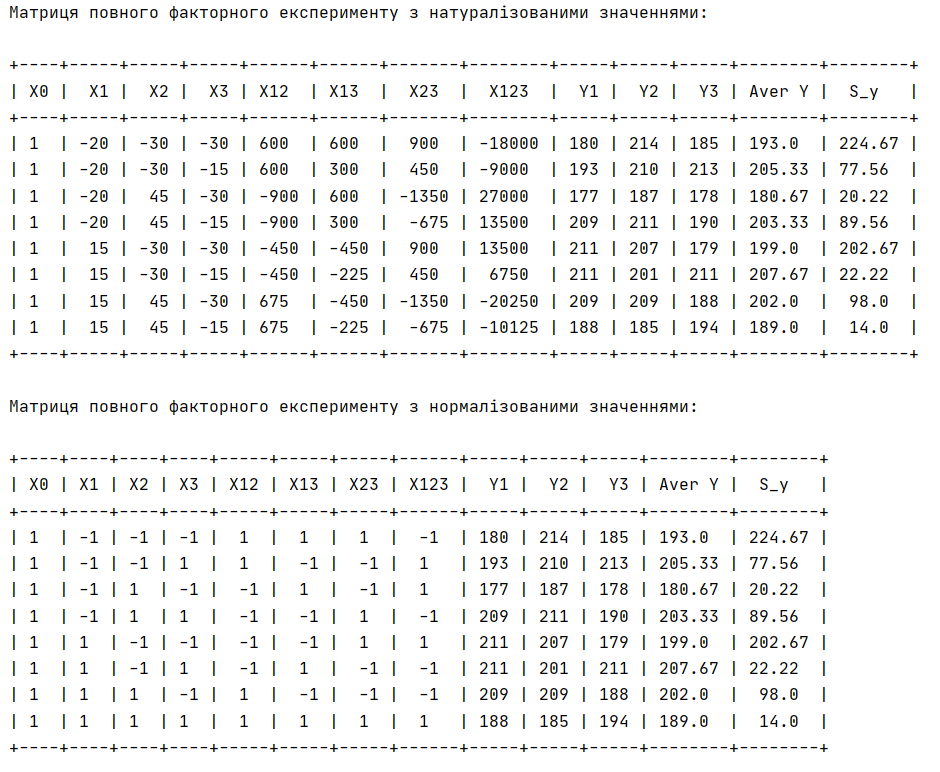


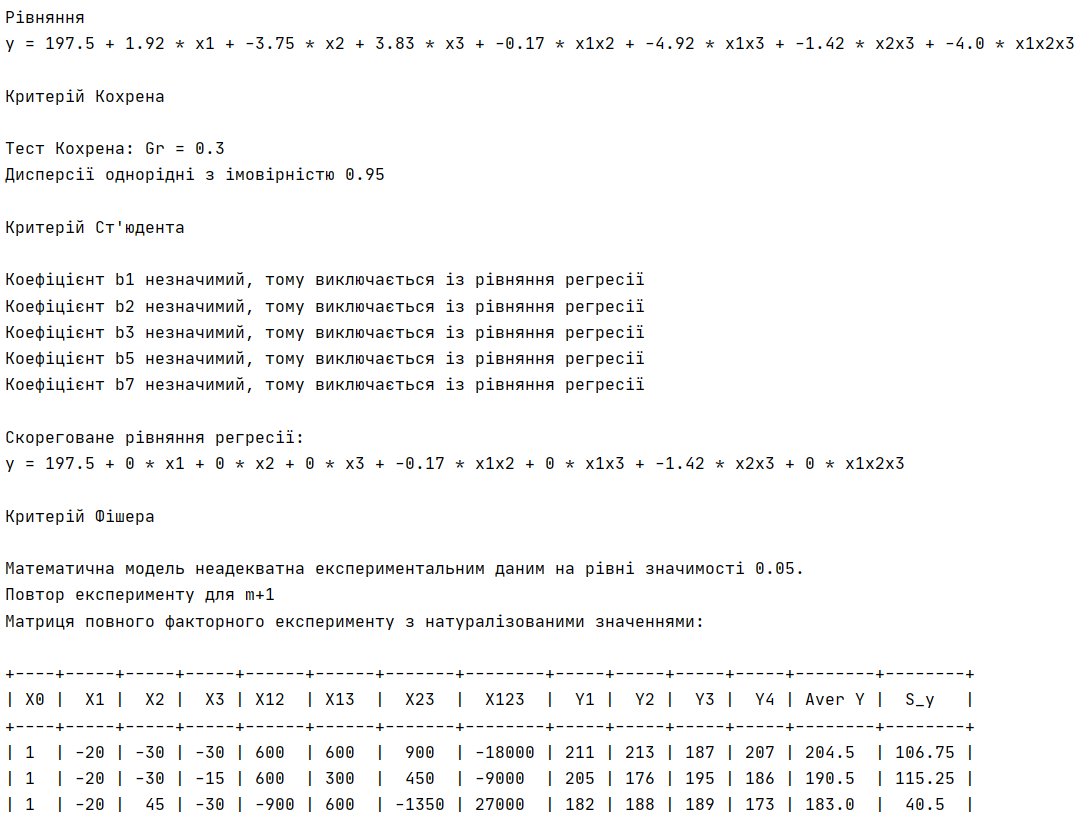
Варіант:

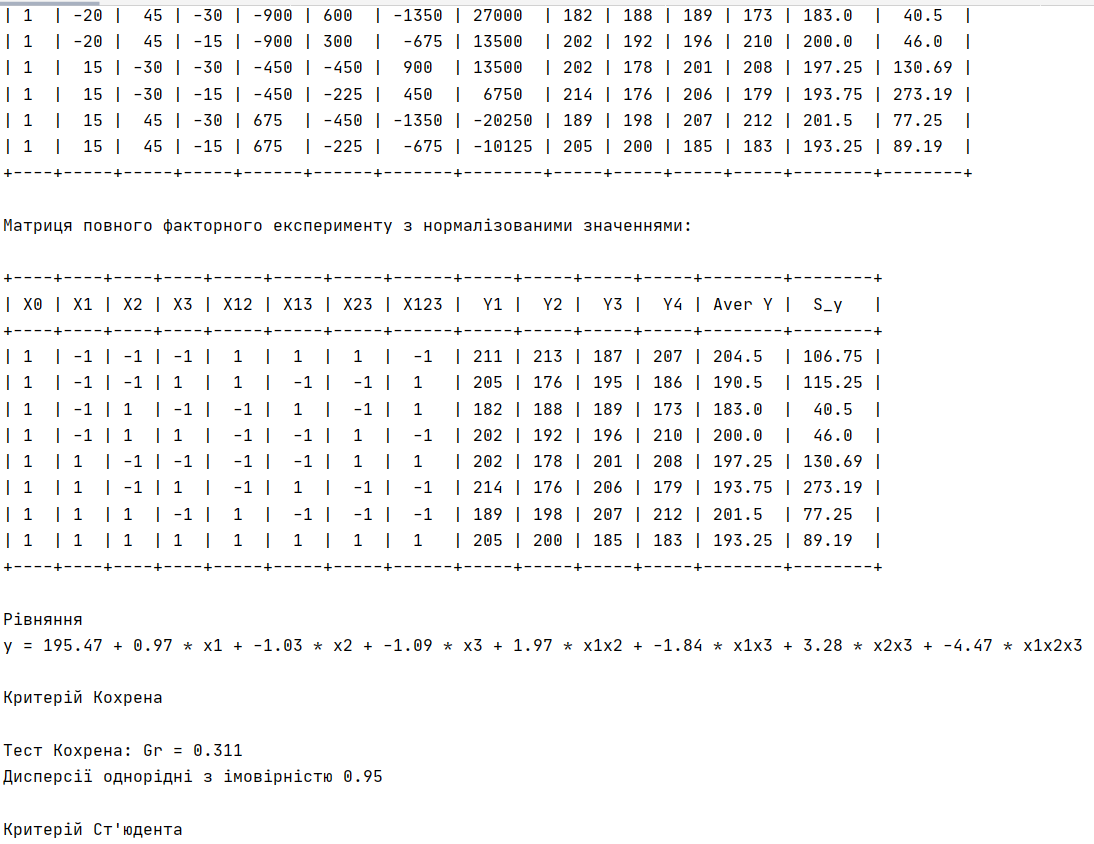


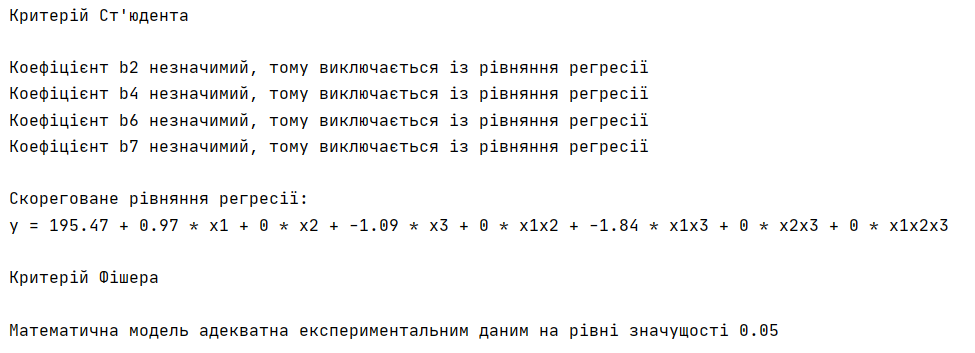
Виконання роботи:

1. Результати роботи програми:









Код програми:

from random import randint  
import prettytable  
import math  
from scipy.stats import f, t  
from functools import partial  
  
*# Лабораторна робота №4 "ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ  
# ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ З УРАХУВАННЯМ ЕФЕКТУ ВЗАЄМОДІЇ" з предмету МОПЕ  
# Варіянт №210 Карпека Дмитрій  
  
#--------------------------------------------------Початкові умови-----------------------------------------------------*variant = 210  
  
min\_x = [-20, -30, -30]  
max\_x = [15, 45, -15]  
  
x0 = [1]  
norm\_x = [[-1, -1, -1],  
 [-1, -1, 1],  
 [-1, 1, -1],  
 [-1, 1, 1],  
 [1, -1, -1],  
 [1, -1, 1],  
 [1, 1, -1],  
 [1, 1, 1]]  
natur\_x = [ [min\_x[0], min\_x[1], min\_x[2]],  
 [min\_x[0], min\_x[1], max\_x[2]],  
 [min\_x[0], max\_x[1], min\_x[2]],  
 [min\_x[0], max\_x[1], max\_x[2]],  
 [max\_x[0], min\_x[1], min\_x[2]],  
 [max\_x[0], min\_x[1], max\_x[2]],  
 [max\_x[0], max\_x[1], min\_x[2]],  
 [max\_x[0], max\_x[1], max\_x[2]] ]  
  
def experiment(m=3, n=8):  
  
 regression\_str = **'y = {} + {} \* x1 + {} \* x2 + {} \* x3 + {} \* x1x2 + {} \* x1x3 + {} \* x2x3 + {} \* x1x2x3'** def matrix\_plan(m, ymin, ymax, n):  
 return [[randint(ymin, ymax) for \_ in range(m)] for \_ in range(n)]  
  
 def multiplication(a, b):  
 return a \* b  
  
 def average(list):  
 return sum(list) / len(list)  
  
 def round\_to\_2(number):  
 return round(number, 2)  
  
 def dispersion(list\_y, aver\_list\_y):  
 return [round\_to\_2(average(list(map(lambda y: (y - aver\_list\_y[i]) \*\* 2, list\_y[i])))) for i in range(len(list\_y))]  
  
 def cochrane\_criteria(S\_y):  
 global m  
 print(**"**\n**Критерій Кохрена**\n**"**)  
 Gp = max(S\_y) / sum(S\_y)  
 q = 0.05  
 q\_ = q / f2  
 chr = f.ppf(q=1 - q\_, dfn=f1, dfd=(f2 - 1) \* f1)  
 Gt = chr / (chr + f2 - 1)  
 print(**"Тест Кохрена: Gr = "** + str(round(Gp, 3)))  
 if Gp < Gt:  
 print(**"Дисперсії однорідні з імовірністю 0.95"**)  
 pass  
 else:  
 print(**"**\n**Дисперсії неоднорідні.**\n**Повтор експерименту для m + 1**\n**"**)  
 m = m + 1  
 experiment(m)  
  
 def student\_criteria(S\_y, d):  
 print(**"**\n**Критерій Ст'юдента**\n**"**)  
 bettaList = [sum(S\_y) \* x0[0] / n,  
 average(list(map(multiplication, S\_y, x1i))),  
 average(list(map(multiplication, S\_y, x2i))),  
 average(list(map(multiplication, S\_y, x3i))),  
 average(list(map(multiplication, S\_y, norm\_x12))),  
 average(list(map(multiplication, S\_y, norm\_x13))),  
 average(list(map(multiplication, S\_y, norm\_x23))),  
 average(list(map(multiplication, S\_y, norm\_x123)))]  
 bettaList = [round\_to\_2(i) for i in bettaList]  
  
 list\_t = [bettaList[i] \* S for i in range(n)]  
  
 for i in range(n):  
 if list\_t[i] < t.ppf(q=0.975, df=f3):  
 list\_b[i] = 0  
 d -= 1  
 print(**'Коефіцієнт b'** + str(i) + **' незначимий, тому виключається із рівняння регресії'**)  
 print(**"**\n**Скореговане рівняння регресії:"**)  
 print(regression\_str.format(\*map(round\_to\_2, list\_b)))  
  
 def fisher\_criteria(d):  
 global m  
 print(**"**\n**Критерій Фішера**\n**"**)  
 f4 = n - d  
 S\_ad = (m \* sum(  
 [(list\_b[0] + list\_b[1] \* x1i[i] + list\_b[2] \* x2i[i] + list\_b[3] \* x3i[i] + list\_b[4] \* norm\_x12[i] +  
 list\_b[5] \* norm\_x13[i] + list\_b[6] \* norm\_x23[i] + list\_b[7] \* norm\_x123[i]  
 - average\_list\_y[i]) \*\* 2 for i in range(n)]) / f4)  
 Fp = S\_ad / Sb  
  
 if Fp > f.ppf(q=0.95, dfn=f4, dfd=f3): *# перевірка критерію Фішера з використанням scipy* print(**'Математична модель неадекватна експериментальним даним на рівні значимості 0.05.**\n**Повтор експерименту для m+1'**)  
 m = m + 1  
 experiment(m)  
 else:  
 print(**'Математична модель адекватна експериментальним даним на рівні значущості 0.05'**)  
  
 def printed\_matrixes():  
 pt1 = prettytable.PrettyTable()  
 pt2 = prettytable.PrettyTable()  
 pt1.field\_names = [**"X0"**, **"X1"**, **"X2"**, **"X3"**] + [**"X12"**, **"X13"**, **"X23"**, **"X123"**] + [**"Y"** + str(x) for x in range(1, m + 1)] + [**"Aver Y"**] + [**"S\_y"**]  
 pt2.field\_names = [**"X0"**, **"X1"**, **"X2"**, **"X3"**] + [**"X12"**, **"X13"**, **"X23"**, **"X123"**] + [**"Y"** + str(x) for x in range(1, m + 1)] + [**"Aver Y"**] + [**"S\_y"**]  
  
 print(**"Матриця повного факторного експерименту з натуралізованими значеннями:**\n**"**)  
 pt1.add\_rows([x0 + natur\_x[i] + natur\_x12[i] + natur\_x13[i] + natur\_x23[i] + natur\_x123[i] + matrix\_y[i] + [average\_list\_y[i]] + [S\_y[i]] for i in range(n)])  
 print(pt1)  
  
 print(**"**\n**Матриця повного факторного експерименту з нормалізованими значеннями:**\n**"**)  
 pt2.add\_rows([x0 + norm\_x[i] + [norm\_x12[i]] + [norm\_x13[i]] + [norm\_x23[i]] + [norm\_x123[i]] + matrix\_y[i] + [average\_list\_y[i]] + [S\_y[i]] for i in range(n)])  
 print(pt2)  
  
 m = m  
 n = 8  
  
 x\_average\_max = sum(max\_x) / 3  
 x\_average\_min = sum(min\_x) / 3  
  
 y\_max = round(200 + x\_average\_max)  
 y\_min = round(200 + x\_average\_min)  
  
 matrix\_y = matrix\_plan(m, y\_min, y\_max, n)  
 average\_list\_y = [round(average(matrix\_y[i]), 2) for i in range(len(matrix\_y))]  
  
 S\_y = dispersion(matrix\_y, average\_list\_y)  
  
 f1 = m - 1  
 f2 = n  
 f3 = f1 \* f2  
 d = 4  
  
 Sb = sum(S\_y) / n  
 S = math.sqrt(Sb / (n \* m))  
  
 norm\_x12 = [norm\_x[i][0] \* norm\_x[i][1] for i in range(len(norm\_x))]  
 norm\_x13 = [norm\_x[i][0] \* norm\_x[i][2] for i in range(len(norm\_x))]  
 norm\_x23 = [norm\_x[i][1] \* norm\_x[i][2] for i in range(len(norm\_x))]  
 norm\_x123 = [norm\_x[i][0] \* norm\_x[i][1] \* norm\_x[i][2] for i in range(len(norm\_x))]  
  
 natur\_x12 = [[natur\_x[i][0] \* natur\_x[i][1]] for i in range(len(natur\_x))]  
 natur\_x13 = [[natur\_x[i][0] \* natur\_x[i][2]] for i in range(len(natur\_x))]  
 natur\_x23 = [[natur\_x[i][1] \* natur\_x[i][2]] for i in range(len(natur\_x))]  
 natur\_x123 = [[natur\_x[i][0] \* natur\_x[i][1] \* natur\_x[i][2]] for i in range(len(natur\_x))]  
  
 x1i = [norm\_x[i][0] for i in range(n)]  
 x2i = [norm\_x[i][1] for i in range(n)]  
 x3i = [norm\_x[i][2] for i in range(n)]  
  
 list\_b = [0] \* n *# b0, b1, b2, b3, b12, b13, b23, b123* list\_b[0] = average(average\_list\_y)  
 list\_b[1] = average([average\_list\_y[i] \* x1i[i] for i in range(n)])  
 list\_b[2] = average([average\_list\_y[i] \* x2i[i] for i in range(n)])  
 list\_b[3] = average([average\_list\_y[i] \* x3i[i] for i in range(n)])  
 list\_b[4] = average([average\_list\_y[i] \* x1i[i] \* x2i[i] for i in range(n)])  
 list\_b[5] = average([average\_list\_y[i] \* x1i[i] \* x3i[i] for i in range(n)])  
 list\_b[6] = average([average\_list\_y[i] \* x2i[i] \* x3i[i] for i in range(n)])  
 list\_b[7] = average([average\_list\_y[i] \* x1i[i]\* x2i[i] \* x3i[i] for i in range(n)])  
  
 printed\_matrixes()  
 print(**"**\n**Рівняння**\n**"** + regression\_str.format(\*map(round\_to\_2, list\_b)))  
 cochrane\_criteria(S\_y)  
 student\_criteria(S\_y, d)  
 fisher\_criteria(d)  
*#--------------------*m = 3  
experiment(m)

**+ {} \* x1 + {} \* x2 + {} \* x3 + {} \* x1x2 + {} \* x1x3 + {} \* x2x3 + {} \* x1x2x3'** def matrix\_plan(m, ymin, ymax, n):  
 return [[randint(ymin, ymax) for \_ in range(m)] for \_ in range(n)]  
  
 def multiplication(a, b):  
 return a \* b  
  
 def average(list):  
 return sum(list) / len(list)  
  
 def round\_to\_2(number):  
 return round(number, 2)  
  
 def dispersion(list\_y, aver\_list\_y):  
 return [round\_to\_2(average(list(map(lambda y: (y - aver\_list\_y[i]) \*\* 2, list\_y[i])))) for i in range(len(list\_y))]  
  
 def cochrane\_criteria(S\_y):  
 global m  
 print(**"**\n**Критерій Кохрена**\n**"**)  
 Gp = max(S\_y) / sum(S\_y)  
 q = 0.05  
 q\_ = q / f2  
 chr = f.ppf(q=1 - q\_, dfn=f1, dfd=(f2 - 1) \* f1)  
 Gt = chr / (chr + f2 - 1)  
 print(**"Тест Кохрена: Gr = "** + str(round(Gp, 3)))  
 if Gp < Gt:  
 print(**"Дисперсії однорідні з імовірністю 0.95"**)  
 pass  
 else:  
 print(**"**\n**Дисперсії неоднорідні.**\n**Повтор експерименту для m + 1**\n**"**)  
 m = m + 1  
 experiment(m)  
  
 def student\_criteria(S\_y, d):  
 print(**"**\n**Критерій Ст'юдента**\n**"**)  
 bettaList = [sum(S\_y) \* x0[0] / n,  
 average(list(map(multiplication, S\_y, x1i))),  
 average(list(map(multiplication, S\_y, x2i))),  
 average(list(map(multiplication, S\_y, x3i))),  
 average(list(map(multiplication, S\_y, norm\_x12))),  
 average(list(map(multiplication, S\_y, norm\_x13))),  
 average(list(map(multiplication, S\_y, norm\_x23))),  
 average(list(map(multiplication, S\_y, norm\_x123)))]  
 bettaList = [round\_to\_2(i) for i in bettaList]  
  
 list\_t = [bettaList[i] \* S for i in range(n)]  
  
 for i in range(n):  
 if list\_t[i] < t.ppf(q=0.975, df=f3):  
 list\_b[i] = 0  
 d -= 1  
 print(**'Коефіцієнт b'** + str(i) + **' незначимий, тому виключається із рівняння регресії'**)  
 print(**"**\n**Скореговане рівняння регресії:"**)  
 print(regression\_str.format(\*map(round\_to\_2, list\_b)))  
  
 def fisher\_criteria(d):  
 global m  
 print(**"**\n**Критерій Фішера**\n**"**)  
 f4 = n - d  
 S\_ad = (m \* sum(  
 [(list\_b[0] + list\_b[1] \* x1i[i] + list\_b[2] \* x2i[i] + list\_b[3] \* x3i[i] + list\_b[4] \* norm\_x12[i] +  
 list\_b[5] \* norm\_x13[i] + list\_b[6] \* norm\_x23[i] + list\_b[7] \* norm\_x123[i]  
 - average\_list\_y[i]) \*\* 2 for i in range(n)]) / f4)  
 Fp = S\_ad / Sb  
  
 if Fp > f.ppf(q=0.95, dfn=f4, dfd=f3): *# перевірка критерію Фішера з використанням scipy* print(**'Математична модель неадекватна експериментальним даним на рівні значимості 0.05.**\n**Повтор експерименту для m+1'**)  
 m = m + 1  
 experiment(m)  
 else:  
 print(**'Математична модель адекватна експериментальним даним на рівні значущості 0.05'**)  
  
 def printed\_matrixes():  
 pt1 = prettytable.PrettyTable()  
 pt2 = prettytable.PrettyTable()  
 pt1.field\_names = [**"X0"**, **"X1"**, **"X2"**, **"X3"**] + [**"X12"**, **"X13"**, **"X23"**, **"X123"**] + [**"Y"** + str(x) for x in range(1, m + 1)] + [**"Aver Y"**] + [**"S\_y"**]  
 pt2.field\_names = [**"X0"**, **"X1"**, **"X2"**, **"X3"**] + [**"X12"**, **"X13"**, **"X23"**, **"X123"**] + [**"Y"** + str(x) for x in range(1, m + 1)] + [**"Aver Y"**] + [**"S\_y"**]  
  
 print(**"Матриця повного факторного експерименту з натуралізованими значеннями:**\n**"**)  
 pt1.add\_rows([x0 + natur\_x[i] + natur\_x12[i] + natur\_x13[i] + natur\_x23[i] + natur\_x123[i] + matrix\_y[i] + [average\_list\_y[i]] + [S\_y[i]] for i in range(n)])  
 print(pt1)  
  
 print(**"**\n**Матриця повного факторного експерименту з нормалізованими значеннями:**\n**"**)  
 pt2.add\_rows([x0 + norm\_x[i] + [norm\_x12[i]] + [norm\_x13[i]] + [norm\_x23[i]] + [norm\_x123[i]] + matrix\_y[i] + [average\_list\_y[i]] + [S\_y[i]] for i in range(n)])  
 print(pt2)  
  
 m = m  
 n = 8  
  
 x\_average\_max = sum(max\_x) / 3  
 x\_average\_min = sum(min\_x) / 3  
  
 y\_max = round(200 + x\_average\_max)  
 y\_min = round(200 + x\_average\_min)  
  
 matrix\_y = matrix\_plan(m, y\_min, y\_max, n)  
 average\_list\_y = [round(average(matrix\_y[i]), 2) for i in range(len(matrix\_y))]  
  
 S\_y = dispersion(matrix\_y, average\_list\_y)  
  
 f1 = m - 1  
 f2 = n  
 f3 = f1 \* f2  
 d = 4  
  
 Sb = sum(S\_y) / n  
 S = math.sqrt(Sb / (n \* m))  
  
 norm\_x12 = [norm\_x[i][0] \* norm\_x[i][1] for i in range(len(norm\_x))]  
 norm\_x13 = [norm\_x[i][0] \* norm\_x[i][2] for i in range(len(norm\_x))]  
 norm\_x23 = [norm\_x[i][1] \* norm\_x[i][2] for i in range(len(norm\_x))]  
 norm\_x123 = [norm\_x[i][0] \* norm\_x[i][1] \* norm\_x[i][2] for i in range(len(norm\_x))]  
  
 natur\_x12 = [[natur\_x[i][0] \* natur\_x[i][1]] for i in range(len(natur\_x))]  
 natur\_x13 = [[natur\_x[i][0] \* natur\_x[i][2]] for i in range(len(natur\_x))]  
 natur\_x23 = [[natur\_x[i][1] \* natur\_x[i][2]] for i in range(len(natur\_x))]  
 natur\_x123 = [[natur\_x[i][0] \* natur\_x[i][1] \* natur\_x[i][2]] for i in range(len(natur\_x))]  
  
 x1i = [norm\_x[i][0] for i in range(n)]  
 x2i = [norm\_x[i][1] for i in range(n)]  
 x3i = [norm\_x[i][2] for i in range(n)]  
  
 list\_b = [0] \* n *# b0, b1, b2, b3, b12, b13, b23, b123* list\_b[0] = average(average\_list\_y)  
 list\_b[1] = average([average\_list\_y[i] \* x1i[i] for i in range(n)])  
 list\_b[2] = average([average\_list\_y[i] \* x2i[i] for i in range(n)])  
 list\_b[3] = average([average\_list\_y[i] \* x3i[i] for i in range(n)])  
 list\_b[4] = average([average\_list\_y[i] \* x1i[i] \* x2i[i] for i in range(n)])  
 list\_b[5] = average([average\_list\_y[i] \* x1i[i] \* x3i[i] for i in range(n)])  
 list\_b[6] = average([average\_list\_y[i] \* x2i[i] \* x3i[i] for i in range(n)])  
 list\_b[7] = average([average\_list\_y[i] \* x1i[i]\* x2i[i] \* x3i[i] for i in range(n)])  
  
 printed\_matrixes()  
 print(**"**\n**Рівняння**\n**"** + regression\_str.format(\*map(round\_to\_2, list\_b)))  
 cochrane\_criteria(S\_y)  
 student\_criteria(S\_y, d)  
 fisher\_criteria(d)  
*#--------------------*m = 3  
experiment(m)