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 рівняння регресії з урахуванням ефекту взаємодії.»

**Виконав:**

студент 2-го курсу ФІОТ

групи ІВ-92

**Копайло Ярослав**

Номер у списку: 11

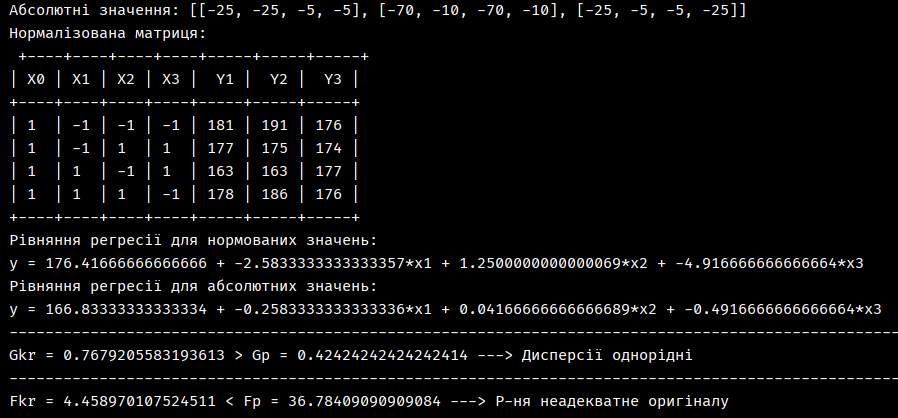
Варiант: **211**

Перевірив:  
Регіда П.Г.

Київ – 2021



#! /usr/bin/env python  
# -\*- coding: utf-8 -\*-  
  
import math  
import numpy as np  
from scipy.stats import t**,** f  
from functools import partial  
import random as r  
from functools import partial  
import prettytable as p  
from numpy.linalg import solve  
import sys  
  
m = **3**prob = **0.95**x1\_min = -**25**x1\_max = -**5**x2\_min = -**70**x2\_max = -**10**x3\_min = -**25**x3\_max = -**5**k = **3**counter = **0**x\_ranges = [[x1\_min**,** x1\_max]**,** [x2\_min**,** x2\_max]**,** [x3\_min**,** x3\_max]]  
x0\_norm = [**1, 1, 1, 1**]  
x1\_norm = [-**1,** -**1, 1, 1**]  
x2\_norm = [-**1, 1,** -**1, 1**]  
x3\_norm = [-**1, 1, 1,** -**1**]  
x1x2\_norm = [a \* b for a**,** b in zip(x1\_norm**,** x2\_norm)]  
x1x3\_norm = [a \* b for a**,** b in zip(x1\_norm**,** x3\_norm)]  
x2x3\_norm = [a \* b for a**,** b in zip(x2\_norm**,** x3\_norm)]  
x1x2x3\_norm = [a \* b \* c for a**,** b**,** c in zip(x1\_norm**,** x2\_norm**,** x3\_norm)]  
N = len(x1\_norm)  
xcp\_max = (x1\_max + x2\_max + x3\_max) / **3**xcp\_min = (x1\_min + x2\_min + x3\_min) / **3**x\_norm = [x1\_norm**,** x2\_norm**,** x3\_norm]  
Y\_min = **200** + xcp\_min  
Y\_max = **200** + xcp\_max  
  
x\_abs = []  
  
for i in range(k):  
 temp = []  
 for j in x\_norm[i]:  
 if j == **1**:  
 temp.append(x\_ranges[i][**1**])  
 else:  
 temp.append(x\_ranges[i][**0**])  
 x\_abs.append(temp)  
print("Абсолютнi значення: " + str(x\_abs))  
  
Y\_exp = []  
for i in range(N):  
 temp = []  
 for \_ in range(m):  
 temp.append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 Y\_exp.append(temp)  
  
  
def y\_perevirka\_norm(x1**,** x2**,** x3):  
 return b0 + x1 \* b1 + x2 \* b2 + x3 \* b3  
  
  
def y\_perevirka\_abs(x1**,** x2**,** x3):  
 return a0 + a1 \* x1 + a2 \* x2 + a3 \* x3  
  
  
def get\_cohren\_critical(prob**,** f1**,** f2):  
 f\_crit = f.isf((**1** - prob) / f2**,** f1**,** (f2 - **1**) \* f1)  
 return f\_crit / (f\_crit + f2 - **1**)  
  
  
fisher\_teor = partial(f.ppf**,** q=**1** - **0.05**)  
student\_teor = partial(t.ppf**,** q=**1** - **0.025**)  
  
odnorid = False  
adekvat = False  
while not adekvat:  
 while not odnorid:  
 table1 = p.PrettyTable()  
 table1.add\_column("X0"**,** x0\_norm)  
 for i in range(k):  
 table1.add\_column("X{0}".format(i + **1**)**,** x\_norm[i])  
 for i in range(m):  
 table1.add\_column("Y{0}".format(i + **1**)**,** [j[i] for j in Y\_exp])  
 print("Нормалізована матриця:\n"**,** table1)  
  
 mx\_norm\_list = [np.mean(i) for i in x\_norm]  
 y\_aver = [np.mean(i) for i in Y\_exp]  
 my = np.mean(y\_aver)  
 a1 = np.mean([x\_norm[**0**][i] \* y\_aver[i] for i in range(N)])  
 a2 = np.mean([x\_norm[**1**][i] \* y\_aver[i] for i in range(N)])  
 a3 = np.mean([x\_norm[**2**][i] \* y\_aver[i] for i in range(N)])  
 a11 = np.mean([x\_norm[**0**][i] \*\* **2** for i in range(N)])  
 a22 = np.mean([x\_norm[**1**][i] \*\* **2** for i in range(N)])  
 a33 = np.mean([x\_norm[**2**][i] \*\* **2** for i in range(N)])  
 a12 = np.mean([x\_norm[**0**][i] \* x\_norm[**1**][i] for i in range(N)])  
 a13 = np.mean([x\_norm[**0**][i] \* x\_norm[**2**][i] for i in range(N)])  
 a23 = np.mean([x\_norm[**1**][i] \* x\_norm[**2**][i] for i in range(N)])  
 a21 = a12  
 a31 = a13  
 a32 = a23  
  
 znam = np.array([[**1,** mx\_norm\_list[**0**]**,** mx\_norm\_list[**1**]**,** mx\_norm\_list[**2**]]**,** [mx\_norm\_list[**0**]**,** a11**,** a12**,** a13]**,** [mx\_norm\_list[**1**]**,** a12**,** a22**,** a32]**,** [mx\_norm\_list[**2**]**,** a13**,** a23**,** a33]])  
  
 b0\_matr = np.array([[my**,** mx\_norm\_list[**0**]**,** mx\_norm\_list[**1**]**,** mx\_norm\_list[**2**]]**,** [a1**,** a11**,** a12**,** a13]**,** [a2**,** a12**,** a22**,** a32]**,** [a3**,** a13**,** a23**,** a33]])  
  
 b1\_matr = np.array([[**1,** my**,** mx\_norm\_list[**1**]**,** mx\_norm\_list[**2**]]**,** [mx\_norm\_list[**0**]**,** a1**,** a12**,** a13]**,** [mx\_norm\_list[**1**]**,** a2**,** a22**,** a32]**,** [mx\_norm\_list[**2**]**,** a3**,** a23**,** a33]])  
  
 b2\_matr = np.array([[**1,** mx\_norm\_list[**0**]**,** my**,** mx\_norm\_list[**2**]]**,** [mx\_norm\_list[**0**]**,** a11**,** a1**,** a13]**,** [mx\_norm\_list[**1**]**,** a12**,** a2**,** a32]**,** [mx\_norm\_list[**2**]**,** a13**,** a3**,** a33]])  
  
 b3\_matr = np.array([[**1,** mx\_norm\_list[**0**]**,** mx\_norm\_list[**1**]**,** my]**,** [mx\_norm\_list[**0**]**,** a11**,** a12**,** a1]**,** [mx\_norm\_list[**1**]**,** a12**,** a22**,** a2]**,** [mx\_norm\_list[**2**]**,** a13**,** a23**,** a3]])  
  
 znam\_value = np.linalg.det(znam)  
 b0 = np.linalg.det(b0\_matr) / znam\_value  
 b1 = np.linalg.det(b1\_matr) / znam\_value  
 b2 = np.linalg.det(b2\_matr) / znam\_value  
 b3 = np.linalg.det(b3\_matr) / znam\_value  
 print("Рівняння регресії для нормованих значень:\ny = {0} + {1}\*x1 + {2}\*x2 + {3}\*x3".format(b0**,** b1**,** b2**,** b3))  
  
 delt\_x1 = (x1\_max - x1\_min) / **2** delt\_x2 = (x2\_max - x2\_min) / **2** delt\_x3 = (x3\_max - x3\_min) / **2** x10 = (x1\_max + x1\_min) / **2** x20 = (x2\_max + x2\_min) / **2** x30 = (x3\_max + x3\_min) / **2** a0 = b0 - b1 \* (x10 / delt\_x1) - b2 \* (x20 / delt\_x2) - b3 \* (x30 / delt\_x3)  
 a1 = b1 / delt\_x1  
 a2 = b2 / delt\_x2  
 a3 = b3 / delt\_x3  
  
 print("Рівняння регресії для абсолютних значень:\ny = {0} + {1}\*x1 + {2}\*x2 + {3}\*x3".format(a0**,** a1**,** a2**,** a3))  
 # Кохрен  
 y\_var = [np.var(Y\_exp[i]) for i in range(N)]  
 flag = False  
 f1 = m - **1** f2 = N  
 f3 = f2 \* f1  
 Gp = max(y\_var) / sum(y\_var)  
 Gkr = get\_cohren\_critical(prob**,** f1**,** f2)  
 print('-' \* **100**)  
 if (Gkr > Gp):  
 print("Gkr = {0} > Gp = {1} ---> Дисперсії однорідні".format(Gkr**,** Gp))  
 odnorid = True  
 else:  
 print("Gkr = {0} < Gp = {1} ---> Дисперсії неоднорідні, збільшимо m і проведемо розрахунки".format(Gkr**,** Gp))  
 Y\_exp[**0**].append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 Y\_exp[**1**].append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 Y\_exp[**2**].append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 Y\_exp[**3**].append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 m += **1** # Стьюдент  
 m = **3** S2B = sum(y\_var) / N  
 S2b = S2B / (N \* m)  
 Sb = math.sqrt(S2b)  
 beta0 = sum([y\_aver[i] \* x0\_norm[i] for i in range(N)]) / N  
 beta1 = sum([y\_aver[i] \* x1\_norm[i] for i in range(N)]) / N  
 beta2 = sum([y\_aver[i] \* x2\_norm[i] for i in range(N)]) / N  
 beta3 = sum([y\_aver[i] \* x3\_norm[i] for i in range(N)]) / N  
 t0 = abs(beta0) / Sb  
 t1 = abs(beta1) / Sb  
 t2 = abs(beta2) / Sb  
 t3 = abs(beta3) / Sb  
 tkr = student\_teor(df=f3)  
  
 d = sum([**1** if tkr < i else **0** for i in [t0**,** t1**,** t2**,** t3]])  
  
 a0 = a0 if tkr < t0 else **0** a1 = a1 if tkr < t1 else **0** a2 = a2 if tkr < t2 else **0** a3 = a3 if tkr < t3 else **0** y\_new = [y\_perevirka\_abs(x\_abs[**0**][i]**,** x\_abs[**1**][i]**,** x\_abs[**2**][i]) for i in range(N)]  
  
 print("-" \* **100**)  
 f4 = N - d  
 S2ad = (m / (N - d)) \* sum([(y\_new[i] - y\_aver[i]) \*\* **2** for i in range(N)])  
 Fp = S2ad / S2b  
 Fkr = fisher\_teor(dfn=f4**,** dfd=f3)  
 if (Fkr > Fp):  
 print("Fkr = {0} > Fp = {1} ---> Р-ня адекватне оригіналу".format(Fkr**,** Fp))  
 adekvat = True  
 else:  
 N = **8** print("Fkr = {0} < Fp = {1} ---> Р-ня неадекватне оригіналу".format(Fkr**,** Fp))  
 x0\_factor = [**1, 1, 1, 1, 1, 1, 1, 1**]  
 x1\_factor = [-**1,** -**1,** -**1,** -**1, 1, 1, 1, 1**]  
 x2\_factor = [-**1,** -**1, 1, 1,** -**1,** -**1, 1, 1**]  
 x3\_factor = [-**1, 1,** -**1, 1,** -**1, 1,** -**1, 1**]  
 x1x2\_factor = [a \* b for a**,** b in zip(x1\_factor**,** x2\_factor)]  
 x1x3\_factor = [a \* b for a**,** b in zip(x1\_factor**,** x3\_factor)]  
 x2x3\_factor = [a \* b for a**,** b in zip(x2\_factor**,** x3\_factor)]  
 x1x2x3\_factor = [a \* b \* c for a**,** b**,** c in zip(x1\_factor**,** x2\_factor**,** x3\_factor)]  
 x0 = [**1, 1, 1, 1, 1, 1, 1, 1**]  
 x1 = [-**25,** -**25,** -**25,** -**25,** -**5,** -**5,** -**5,** -**5**]  
 x2 = [-**70,** -**70,** -**10,** -**10,** -**70,** -**70,** -**10,** -**10**]  
 x3 = [-**25,** -**5,** -**25,** -**5,** -**25,** -**5,** -**25,** -**5**]  
 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)]  
 m = **3** odnorid2 = False  
 while not odnorid2:  
 y1**,** y2**,** y3 = []**,** []**,** []  
 for i in range(N):  
 y1.append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 y2.append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 y3.append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 Y\_row\_arr = [  
 [y1[**0**]**,** y2[**0**]**,** y3[**0**]]**,** [y1[**1**]**,** y2[**1**]**,** y3[**1**]]**,** [y1[**2**]**,** y2[**2**]**,** y3[**2**]]**,** [y1[**3**]**,** y2[**3**]**,** y3[**3**]]**,** [y1[**4**]**,** y2[**4**]**,** y3[**4**]]**,** [y1[**5**]**,** y2[**5**]**,** y3[**5**]]**,** [y1[**6**]**,** y2[**6**]**,** y3[**6**]]**,** [y1[**7**]**,** y2[**7**]**,** y3[**7**]]  
 ]  
 Y\_row\_av\_arr = list(map(lambda x: np.average(x)**,** Y\_row\_arr))  
 Y\_row\_av\_arr = list(map(lambda x: round(x**, 3**)**,** Y\_row\_av\_arr))  
  
 column\_names1 = ["X0"**,** "X1"**,** "X2"**,** "X3"**,** "X1X2"**,** "X1X3"**,** "X2X3"**,** "X1X2X3"**,** "Y1"**,** "Y2"**,** "Y3"]  
 pt2 = p.PrettyTable() # Table  
 pt2.add\_column(column\_names1[**0**]**,** x0\_factor)  
 pt2.add\_column(column\_names1[**1**]**,** x1\_factor)  
 pt2.add\_column(column\_names1[**2**]**,** x2\_factor)  
 pt2.add\_column(column\_names1[**3**]**,** x3\_factor)  
 pt2.add\_column(column\_names1[**4**]**,** x1x2\_factor)  
 pt2.add\_column(column\_names1[**5**]**,** x1x3\_factor)  
 pt2.add\_column(column\_names1[**6**]**,** x2x3\_factor)  
 pt2.add\_column(column\_names1[**7**]**,** x1x2x3\_factor)  
 pt2.add\_column(column\_names1[**8**]**,** y1)  
 pt2.add\_column(column\_names1[**9**]**,** y2)  
 pt2.add\_column(column\_names1[**10**]**,** y3)  
 print(pt2)  
  
 list\_for\_solve\_b = [x0\_factor**,** x1\_factor**,** x2\_factor**,** x3\_factor**,** x1x2\_factor**,** x1x3\_factor**,** x2x3\_factor**,** x1x2x3\_factor]  
 list\_for\_solve\_a = list(zip(x0**,** x1**,** x2**,** x3**,** x1x2**,** x1x3**,** x2x3**,** x1x2x3))  
  
 list\_bi = []  
 for k in range(N):  
 S = **0** for i in range(N):  
 S += (list\_for\_solve\_b[k][i] \* Y\_row\_av\_arr[i]) / N  
 list\_bi.append(round(S**, 5**))  
  
 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**]))  
  
 pt2 = p.PrettyTable() # Table  
 pt2.add\_column(column\_names1[**0**]**,** x0)  
 pt2.add\_column(column\_names1[**1**]**,** x1)  
 pt2.add\_column(column\_names1[**2**]**,** x2)  
 pt2.add\_column(column\_names1[**3**]**,** x3)  
 pt2.add\_column(column\_names1[**4**]**,** x1x2)  
 pt2.add\_column(column\_names1[**5**]**,** x1x3)  
 pt2.add\_column(column\_names1[**6**]**,** x2x3)  
 pt2.add\_column(column\_names1[**7**]**,** x1x2x3)  
 pt2.add\_column(column\_names1[**8**]**,** y1)  
 pt2.add\_column(column\_names1[**9**]**,** y2)  
 pt2.add\_column(column\_names1[**10**]**,** y3)  
 print(pt2)  
  
 list\_ai = [round(i**, 5**) for i in solve(list\_for\_solve\_a**,** Y\_row\_av\_arr)]  
 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**]))  
  
 list\_for\_solve\_b = [x0\_factor**,** x1\_factor**,** x2\_factor**,** x3\_factor**,** x1x2\_factor**,** x1x3\_factor**,** x2x3\_factor**,** x1x2x3\_factor]  
 disp = []  
 for k in range(N):  
 disp.append(np.var(Y\_row\_arr[k]))  
 # Кохрен  
 y\_var = [np.var(Y\_row\_arr[i]) for i in range(N)]  
 f1 = m - **1** f2 = N  
 f3 = f2 \* f1  
 Gp = max(y\_var) / sum(y\_var)  
 Gkr = get\_cohren\_critical(prob**,** f1**,** f2)  
 print('-' \* **100**)  
 if (Gkr > Gp):  
 print("Gkr = {0} > Gp = {1} ---> Дисперсії однорідні".format(Gkr**,** Gp))  
 odnorid2 = True  
  
 else:  
 print("Gkr = {0} < Gp = {1} ---> Дисперсії неоднорідні, збільшимо m і проведемо розрахунки".format(Gkr**,** Gp))  
 Y\_exp[**0**].append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 Y\_exp[**1**].append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 Y\_exp[**2**].append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 Y\_exp[**3**].append(r.randint(math.floor(Y\_min)**,** math.floor(Y\_max)))  
 m += **1** # Стьюдент  
 beta = []  
 Dispersion\_B = sum(y\_var) / N  
 Dispersion\_beta = Dispersion\_B / (m \* N)  
 S\_beta = math.sqrt(abs(Dispersion\_beta))  
 x\_factor\_for\_solve = [x0\_factor**,** x1\_factor**,** x2\_factor**,** x3\_factor**,** x1x2\_factor**,** x1x3\_factor**,** x2x3\_factor**,** x1x2x3\_factor]  
 for j in range(N):  
 beta.append(sum([Y\_row\_av\_arr[i] \* x\_factor\_for\_solve[j][i] for i in range(N)]) / N)  
  
 t\_list = []  
 for i in range(N):  
 t\_list.append(abs(beta[i]) / S\_beta)  
  
 f3 = f1 \* f2  
 d = **0** T = student\_teor(df=f3)  
  
  
 def student(t\_teor**,** t\_pr):  
 return t\_pr < t\_teor  
  
  
 print("t табличне = "**,** T)  
 for i in range(**1,** len(t\_list)):  
 if student(t\_list[i]**,** T):  
 list\_ai[i] = **0** print("Гіпотеза підтверджена, beta{} = 0".format(i))  
 else:  
 print("Гіпотеза не підтверджена.\nbeta{} = {}".format(i**,** list\_ai[i]))  
 d += **1** Y\_counted\_for\_Student = [**0, 0, 0, 0, 0, 0, 0, 0**]  
 x\_arr\_for\_solve = [x1**,** x2**,** x3**,** x1x2**,** x1x3**,** x2x3**,** x1x2x3]  
 for i in range(len(x\_arr\_for\_solve) + **1**):  
 Y\_counted\_for\_Student[i] += list\_ai[**0**]  
 for j in range(len(list\_ai) - **1**):  
 Y\_counted\_for\_Student[i] += list\_ai[j + **1**] \* x\_arr\_for\_solve[j][i]  
  
 f4 = N - d  
 try:  
 S2ad = (m / (N - d)) \* sum([(Y\_counted\_for\_Student[i] - Y\_row\_av\_arr[i]) \*\* **2** for i in range(N)])  
 except:  
 S2ad = **0** Fp = S2ad / Dispersion\_beta  
 Ft = fisher\_teor(dfn=f4**,** dfd=f3)  
 print(Fp**,** Ft)  
 if Ft > Fp:  
 print("Рівняння регресії адекватне")  
 adekvat = True  
 else:  
 print("Рівняння регресії неадекватне")  
 print("Новий експеремент")  
 print("#" \* **100**)  
 N = **4** k = **3** odnorid2 = False  
 odnorid = False  
 counter += **1**

**Результат роботи програми:**

