Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**Методи оптимізації та планування експерименту**

**ЛАБОРАТОРНА РОБОТА № 2**

«ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»

ВИКОНАЛА:

студент ІІ курсу ФІОТ

групи ІВ-92

Варіант №211

Копайло Ярослав

ПЕРЕВІРИВ:

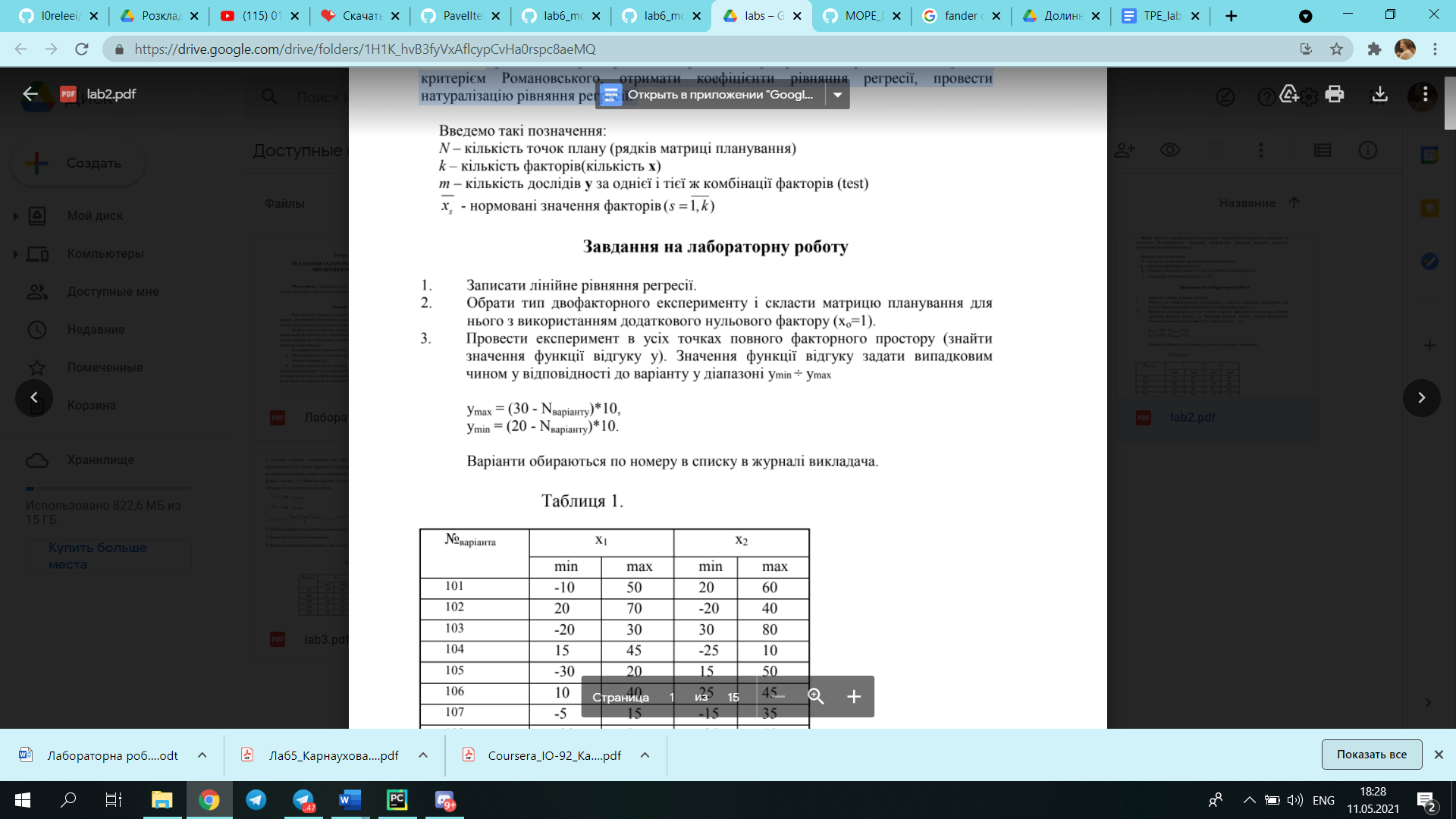
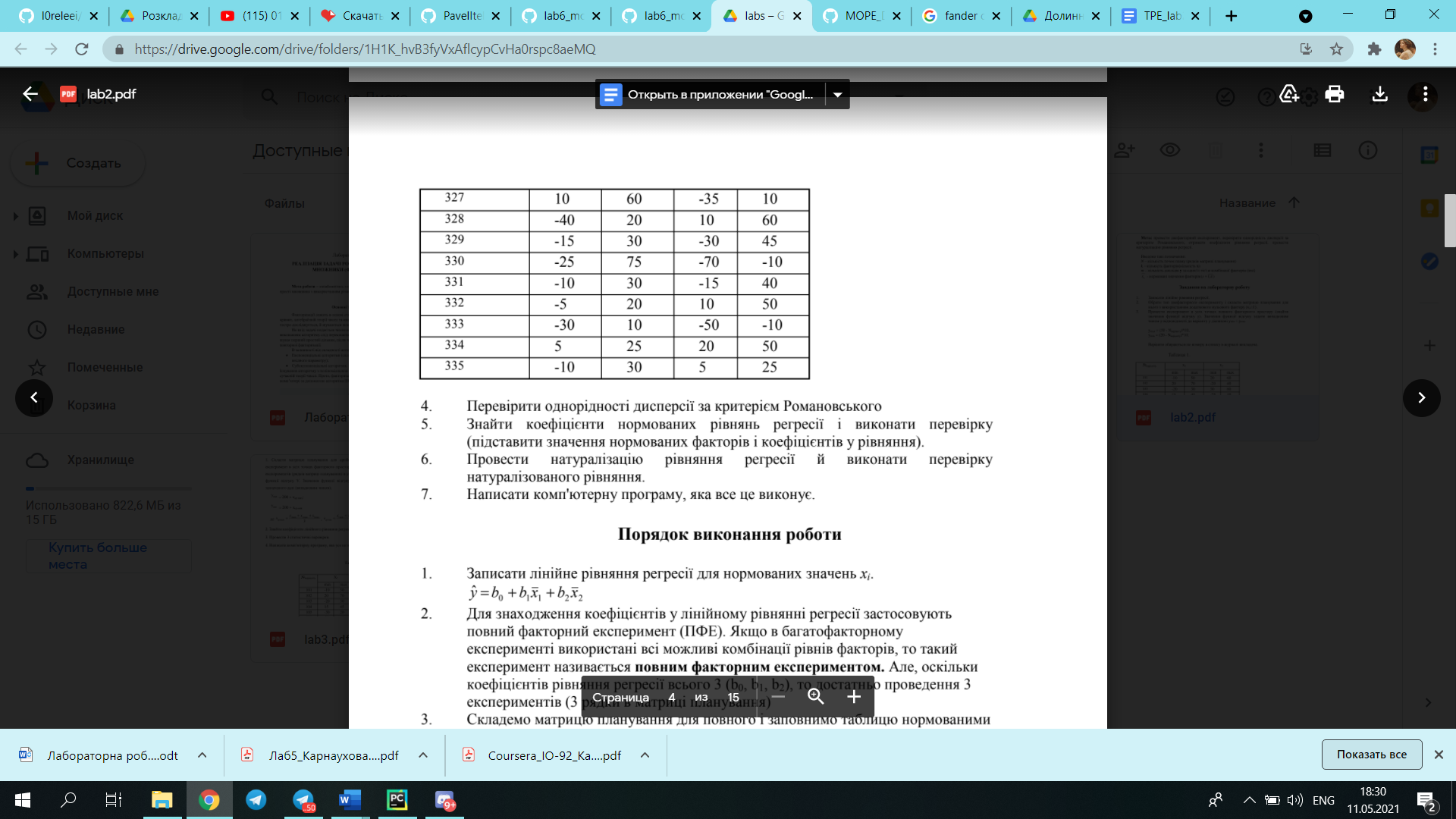
Доц. Порєв В.М.

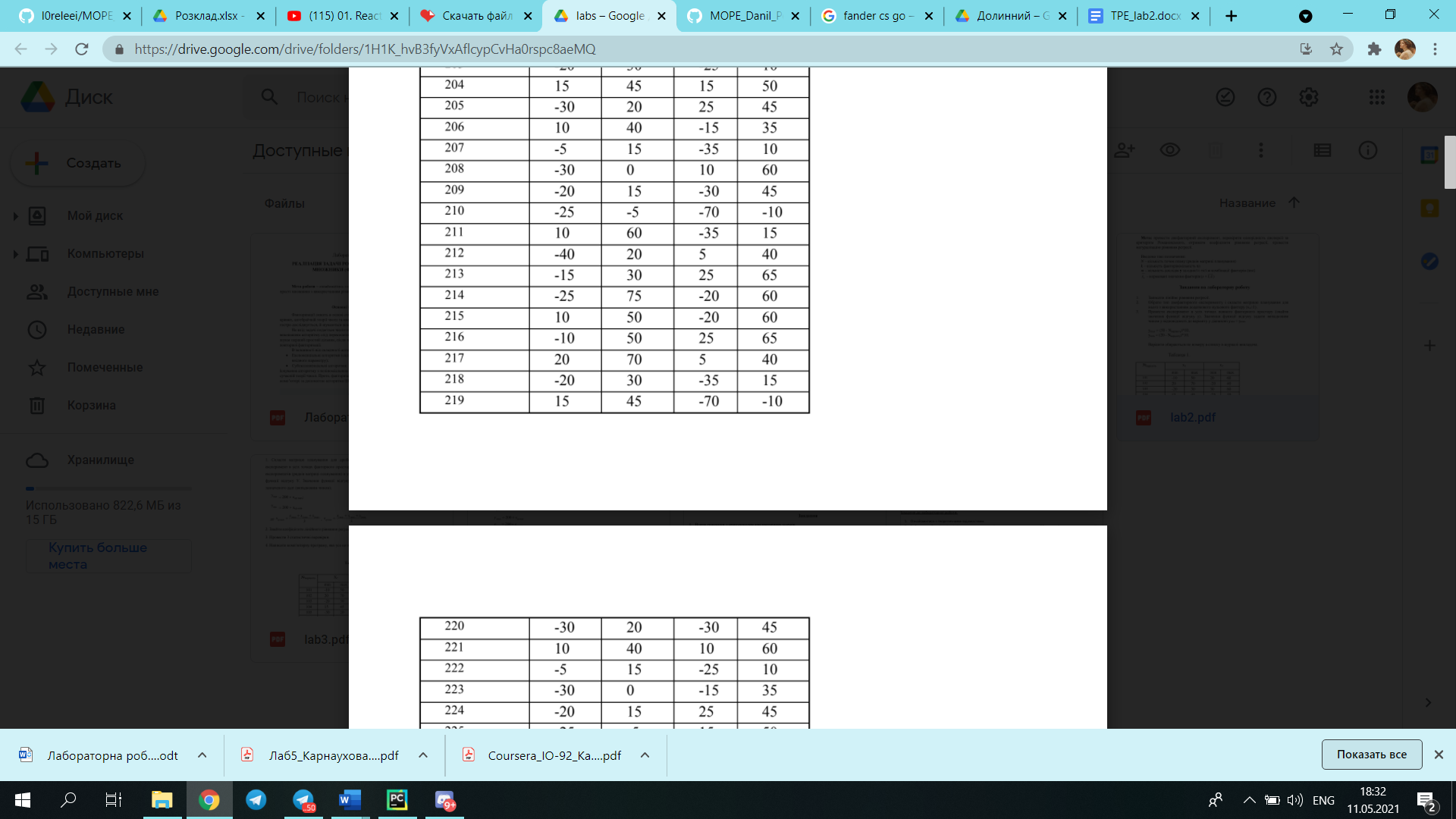
Київ – 2021

**ЛАБОРАТОРНА РОБОТА № 2**

**Тема:** ПРОВЕДЕННЯ ДВОФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

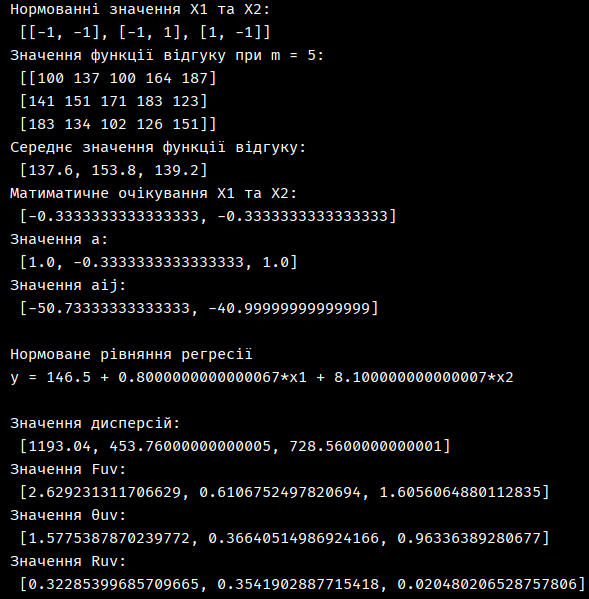
**Мета:** провести двофакторний експеримент, перевірити однорідність дисперсії за критерієм Романовського, отримати коефіцієнти рівняння регресії, провести натуралізацію рівняння регресії.

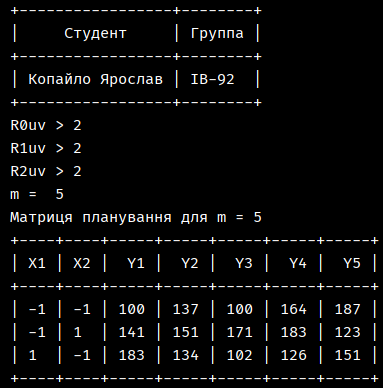
 **Завдання на лабораторну роботу:**

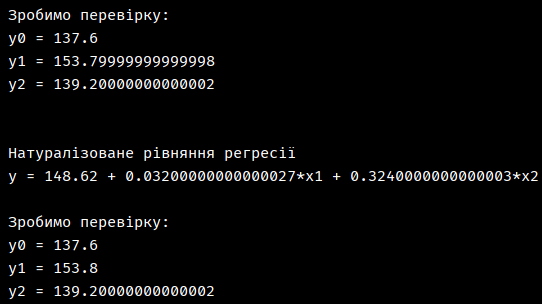
**Варіант завдання:**

**Код програми:**

import random as ran  
import math as ma  
import numpy  
from prettytable import PrettyTable  
  
table0 = PrettyTable()  
table0.field\_names = (["Студент"**,** "Группа"])  
name = "Копайло Ярослав"  
group = "ІВ-92"  
table0.add\_row([name**,** group])  
print(table0)  
  
def inf(count**,** mean):  
 array\_x1 = [**10, 60**]  
 m = **5** + count  
 array\_x2 = [-**35, 15**]  
 array\_y = [(**30** - **11**) \* **10,** (**20** - **11**) \* **10**]  
 teta0 = ma.sqrt(**2** \* (**2** \* m - **2**) / (m \* (m - **4**)))  
 matr\_x = []  
 matr\_x2 = []  
 matr\_y = [[ran.randint(array\_y[**1**]**,** array\_y[**0**]) for j in range(m)] for i in range(**3**)]  
 avey = []  
 sigma = []  
 array\_fuv = []  
 array\_a = []  
 array\_aij = []  
 for i in range(**3**):  
  
 if i == **0**:  
 matr\_x.append([-**1,** -**1**])  
 matr\_x2.append([min(array\_x1)**,** min(array\_x2)])  
 elif i == **1**:  
 matr\_x.append([-**1, 1**])  
 matr\_x2.append([min(array\_x1)**,** max(array\_x2)])  
 elif i == **2**:  
 matr\_x.append([**1,** -**1**])  
 matr\_x2.append([max(array\_x1)**,** min(array\_x2)])  
 for i in range(len(matr\_y)):  
 sumer = **0** temp = sum(matr\_y[i]) / len(matr\_y[i])  
 avey.append(temp)  
 for j in range(len(matr\_y[i])):  
 sumer += (matr\_y[i][j] - temp) \*\* **2** sigma.append(sumer / len(matr\_y[i]))  
 array\_fuv.append(sigma[**0**] / sigma[**1**])  
 array\_fuv.append(sigma[**2**] / sigma[**0**])  
 array\_fuv.append(sigma[**2**] / sigma[**1**])  
 array\_tetas = [(m - **2**) / m \* array\_fuv[i] for i in range(len(array\_fuv))]  
 array\_ruv = [ma.fabs(i - **1**) / teta0 for i in array\_tetas]  
 for i in range(len(array\_ruv)):  
 if array\_ruv[i] < **2**:  
 mean = True  
 print("R{0}uv > 2".format(i))  
 else:  
 mean = False  
 print("R{0}uv < 2".format(i))  
 if mean:  
 trans = numpy.array(matr\_x).transpose()  
 array\_mx = [sum(trans[i]) / len(trans[i]) for i in range(**2**)]  
 my = sum(avey) / len(avey)  
 for i in range(**2**):  
 temp = **0** if i == **1**:  
 for j in matr\_x:  
 temp += numpy.array(j).prod()  
 array\_a.append(temp / **3**)  
 temp = **0** for j in range(len(trans[i])):  
 temp += (trans[i][j] \*\* **2**)  
 array\_a.append(temp / **3**)  
 for i in range(**2**):  
 temp = **0** for j in range(len(trans[i])):  
 temp += trans[i][j] \* avey[j]  
 array\_aij.append(temp / **3**)  
 first = numpy.array(  
 [[**1,** array\_mx[**0**]**,** array\_mx[**1**]]**,** [array\_mx[**0**]**,** array\_a[**0**]**,** array\_a[**1**]]**,** [array\_mx[**1**]**,** array\_a[**1**]**,** array\_a[**2**]]])  
 second = numpy.array([my**,** array\_aij[**0**]**,** array\_aij[**1**]])  
 res = numpy.linalg.solve(first**,** second)  
 array\_delx = [(max(array\_x1) - min(array\_x1)) / **2,** (max(array\_x2) - min(array\_x2)) / **2**]  
 array\_zerx = [sum(array\_x1) / **2,** sum(array\_x2) / **2**]  
 a0 = res[**0**] - res[**1**] \* (array\_zerx[**0**] / array\_delx[**0**]) - res[**2**] \* (array\_zerx[**1**] / array\_delx[**1**])  
 a1 = res[**1**] / array\_delx[**0**]  
 a2 = res[**2**] / array\_delx[**1**]  
 ta = PrettyTable()  
 ta.field\_names = ["X1"**,** "X2"**,** "Y1"**,** "Y2"**,** "Y3"**,** "Y4"**,** "Y5"]  
 ta.add\_rows(  
 [  
 [matr\_x[**0**][**0**]**,** matr\_x[**0**][**1**]**,** matr\_y[**0**][**0**]**,** matr\_y[**0**][**1**]**,** matr\_y[**0**][**2**]**,** matr\_y[**0**][**3**]**,** matr\_y[**0**][**4**]]**,** [matr\_x[**1**][**0**]**,** matr\_x[**1**][**1**]**,** matr\_y[**1**][**0**]**,** matr\_y[**1**][**1**]**,** matr\_y[**1**][**2**]**,** matr\_y[**1**][**3**]**,** matr\_y[**1**][**4**]]**,** [matr\_x[**2**][**0**]**,** matr\_x[**2**][**1**]**,** matr\_y[**2**][**0**]**,** matr\_y[**2**][**1**]**,** matr\_y[**2**][**2**]**,** matr\_y[**2**][**3**]**,** matr\_y[**2**][**4**]]**,** ]  
 )  
 print("m = "**,** m)  
 print("Матриця планування для m = 5")  
 print(ta)  
 print("Нормованні значення X1 та X2:\n"**,** matr\_x)  
 print("Значення функції відгуку при m = {0}:\n".format(m)**,** numpy.array(matr\_y))  
 print("Середнє значення функції відгуку:\n"**,** avey)  
 print("Матиматичне очікування X1 та X2:\n"**,** array\_mx)  
 print("Значення а:\n"**,** array\_a)  
 print("Значення aij:\n"**,** array\_aij**,** "\n")  
 print("Нормоване рівняння регресії")  
 print("y = {0} + {1}\*x1 + {2}\*x2\n".format(res[**0**]**,** res[**1**]**,** res[**2**]))  
 print("Значення дисперсій:\n"**,** sigma)  
 print("Значення Fuv:\n"**,** array\_fuv)  
 print("Значення θuv:\n"**,** array\_tetas)  
 print("Значення Ruv:\n"**,** array\_ruv)  
  
 print("Зробимо перевірку:")  
 for i in range(len(matr\_x)):  
 check = res[**0**] + res[**1**] \* matr\_x[i][**0**] + res[**2**] \* matr\_x[i][**1**]  
 print("y{0} = {1}".format(i**,** check))  
 print("\n")  
 print("Натуралізоване рівняння регресії")  
 print("y = {0} + {1}\*x1 + {2}\*x2\n".format(a0**,** a1**,** a2))  
 print("Зробимо перевірку:")  
 for i in range(len(matr\_x2)):  
 check = a0 + a1 \* matr\_x2[i][**0**] + a2 \* matr\_x2[i][**1**]  
 print("y{0} = {1}".format(i**,** check))  
 return mean  
 else:  
 return mean  
  
  
a = **0**while True:  
 b = False  
 if inf(a**,** b):  
 n = input("Введіть \"Кінець\" щоб зупинити програму: ")  
 if n == "Кінець":  
 break  
 else:  
 n = input("Введіть \"Кінець\" щоб зупинити програму:")  
 if n == "Кінець":  
 break  
 print("Збільшуємо m на 1")  
 a += **1**

**Результат роботи програми:**





**Висновок:**

Проробивши лабораторну роботу, було проведено трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Та знайдено рівняння регресії адекватне об'єкту. У ході виконання лабораторної роботи проблем не виникло. Результати виконання лабораторної висвітлені на роздруківках.

**Контрольні запитання**

1. **Що таке регресійні поліноми і де вони застосовуються?**

В теорії планування експерименту найважливішою частиною є оцінка результатів вимірів. При цьому використовують апроксимуючі поліноми, за допомогою яких ми можемо описати нашу функцію. В ТПЕ ці поліноми отримали спеціальну назву - регресійні поліноми

1. **Визначення однорідності дисперсії.**

Для цього необхідно спочатку знайти середньоарифметичне значення дослідів y̅j (j=1,m̅̅̅̅̅) (математичне сподівання myj) в кожній точці факторного простору: y̅j=(1/m)∑ 𝐲 𝐦 𝟏 js(i=1,N̅̅̅̅̅). Оскільки теоретичні значення дисперсії σ 2 j (j=1,N̅̅̅̅̅) невідомі, то перевірка однорідності дисперсії виконується на основі аналізу статистичних оцінок дисперсії S 2 j (i=1,N̅̅̅̅̅) для усіх точок факторного простору. Статистичні оцінки дисперсії S 2 j (j=1,N̅̅̅̅̅) для кожної точки факторного простору розрахо-вують за формулою: S 2 j ={1/(m-1)}{∑ (𝐲 𝐦 𝟏 js-y̅j) 2 } (j=1,N̅̅̅̅̅). Отже, перевірка однорідності дисперсії – це перевірка гіпотези стосовно належності N значень статистичних оцінок дисперсії S 2 j (i=1,N̅̅̅̅̅) одній генеральній сукупності.

1. **Що називається повним факторним експериментом?**

ПФЕ – повний факторний експеримент,- це коли використовуються усі можливі комбінації рівнів факторів; при ПФЕ кількість комбінацій Nп=r k .