Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №5

з дисципліни *« Методи оптимізації та планування »* на тему

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

Виконала:

студентка II курсу ФІОТ

групи ІО – 91

Копернак Єлизавета

Номер залікової книжки: 9118

Перевірив:

ас. Регіда П.Г.

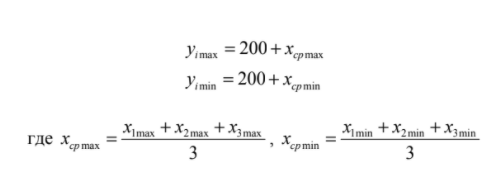
Київ – 2021

**Мета**

Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

**Завдання на лабораторну роботу**

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.



4.Розрахувати коефіцієнти рівняння регресії і записати його.

5. Провести 3 статистичні перевірки.**Варіант завдання**

****

Xcp max = 17/3 ≈ 6

Xcp min = -17/3 ≈ -6

Ymax = 206

Ymin = 194

**Роздруківка тексту програми:**

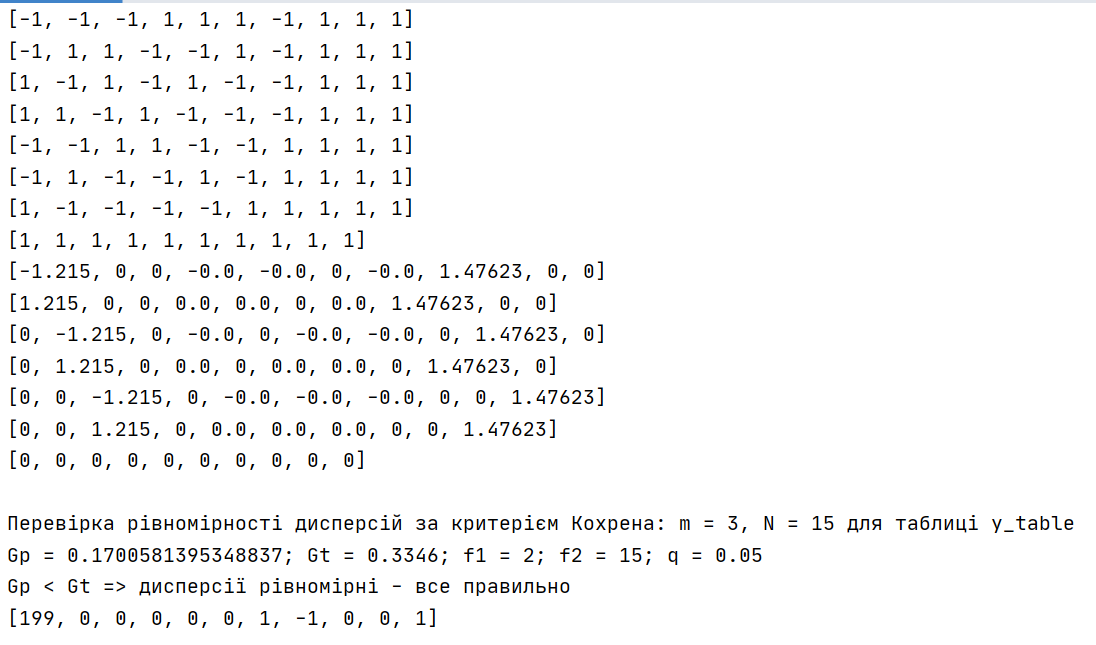
*Module Lab5\_1.py*

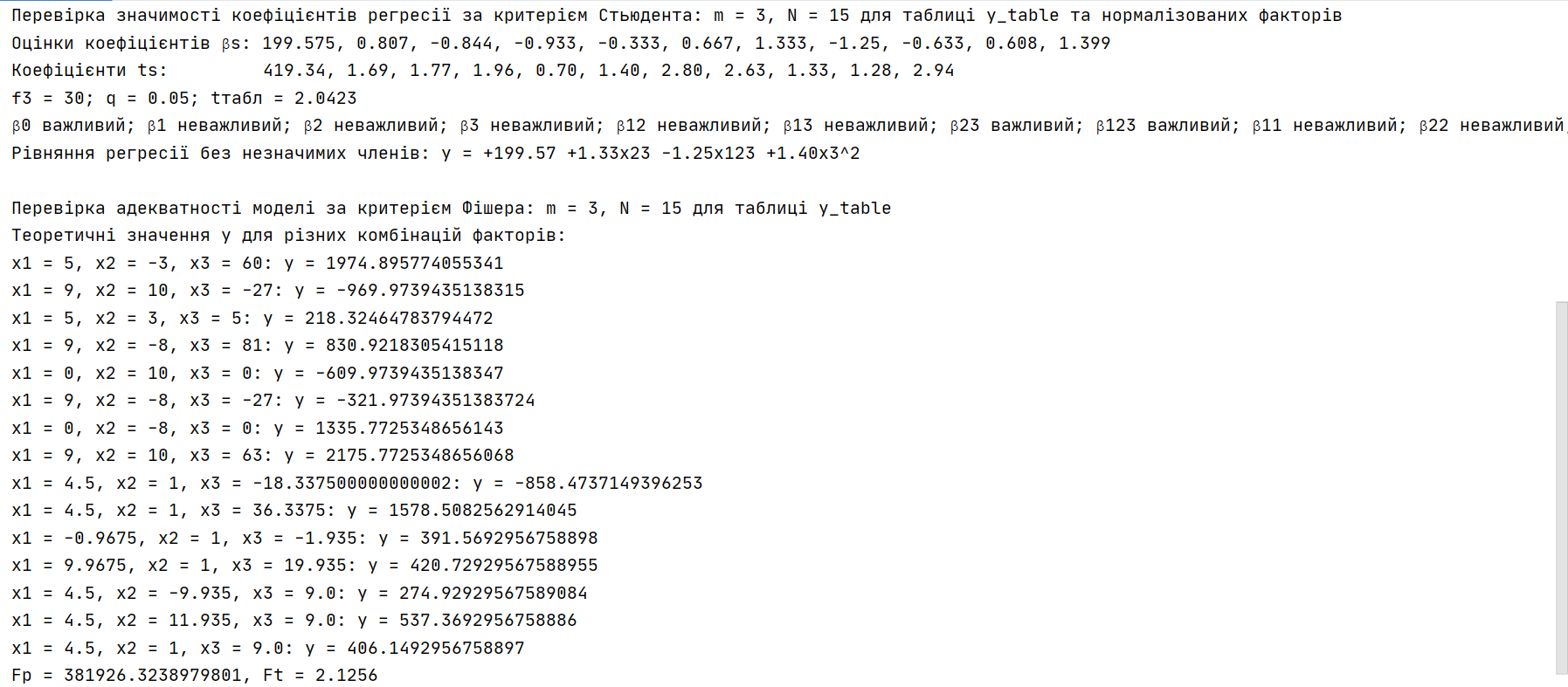
import random  
  
from Lab5\_2 import \*  
  
factors\_table = generate\_factors\_table(raw\_factors\_table)  
for row in factors\_table:  
 print(row)  
naturalized\_factors\_table = generate\_factors\_table(raw\_naturalized\_factors\_table)  
with\_null\_factor = list(map(lambda x: [1] + x, naturalized\_factors\_table))  
  
m = 3  
N = 15  
ymin = 194  
ymax = 206  
y\_arr = [[random.randint(ymin, ymax) for \_ in range(m)] for \_ in range(N)]  
while not cochran\_criteria(m, N, y\_arr):  
 m+=1  
 y\_arr = [[random.randint(ymin, ymax) for \_ in range(m)] for \_ in range(N)]  
  
y\_i = np.array([np.average(row) for row in y\_arr])  
  
coefficients = [[m\_ij(x\_i(column)\*x\_i(row)) for column in range(11)] for row in range(11)]  
  
free\_values = [m\_ij(y\_i, x\_i(i)) for i in range(11)]  
  
beta\_coefficients = np.linalg.solve(coefficients, free\_values)  
print(list(map(int,beta\_coefficients)))  
  
importance = student\_criteria(m, N, y\_arr, beta\_coefficients)  
d = len(list(filter(None, importance)))  
fisher\_criteria(m, N, d, naturalized\_factors\_table, y\_arr, beta\_coefficients, importance)

*Module Lab5\_2.py*

import math  
from \_pydecimal import Decimal  
from scipy.stats import f, t, ttest\_ind, norm  
  
from functools import reduce  
from itertools import compress  
import numpy as np  
  
  
raw\_naturalized\_factors\_table = [[12, 5, -3],  
 [-3, 9, 10],  
 [+1, 5, 3],  
 [+9, 9, -8],  
  
 [-3, 0, 10],  
 [-3, 9, -8],  
 [+7, 0, -8],  
 [+7, 9, 10],  
  
 [-4.075, +4.5, +1],  
 [+8.075, +4.5, +1],  
 [2, -0.9675, +1],  
 [2, +9.9675, +1],  
 [2, +4.5, -9.935],  
 [2, +4.5, 11.935],  
  
 [2, +4.5, +1]]  
  
raw\_factors\_table = [[-1, -1, -1],  
 [-1, +1, +1],  
 [+1, -1, +1],  
 [+1, +1, -1],  
  
 [-1, -1, +1],  
 [-1, +1, -1],  
 [+1, -1, -1],  
 [+1, +1, +1],  
  
 [-1.215, 0, 0],  
 [+1.215, 0, 0],  
 [0, -1.215, 0],  
 [0, +1.215, 0],  
 [0, 0, -1.215],  
 [0, 0, +1.215],  
  
 [0, 0, 0]]  
  
  
def generate\_factors\_table(raw\_array):  
 return [row + [row[0] \* row[1], row[0] \* row[2], row[1] \* row[2], row[0] \* row[1] \* row[2]]  
 + list(map(lambda x: round(x \*\* 2, 5), row))  
 for row in raw\_array]  
  
  
def x\_i(i):  
 try:  
 assert i <= 10  
 except:  
 raise AssertionError(**"i must be smaller or equal 10"**)  
 with\_null\_factor = list(map(lambda x: [1] + x, generate\_factors\_table(raw\_factors\_table)))  
 res = [row[i] for row in with\_null\_factor]  
 return np.array(res)  
  
  
def cochran\_criteria(m, N, y\_table):  
 print(**"**\n**Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N = {} для таблиці y\_table"**.format(m, N))  
 y\_variations = [np.var(i) for i in y\_table]  
 max\_y\_variation = max(y\_variations)  
 gp = max\_y\_variation/sum(y\_variations)  
 f1 = m - 1  
 f2 = N  
 p = 0.95  
 q = 1-p  
 gt = get\_cochran\_value(f1,f2, q)  
 print(**"Gp = {}; Gt = {}; f1 = {}; f2 = {}; q = {:.2f}"**.format(gp, gt, f1, f2, q))  
 if gp < gt:  
 print(**"Gp < Gt => дисперсії рівномірні - все правильно"**)  
 return True  
 else:  
 print(**"Gp > Gt => дисперсії нерівномірні - треба ще експериментів"**)  
 return False  
  
  
def student\_criteria(m, N, y\_table, beta\_coefficients):  
 print(**"**\n**Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = {}, N = {} "  
 "для таблиці y\_table та нормалізованих факторів"**.format(m, N))  
 average\_variation = np.average(list(map(np.var, y\_table)))  
  
 y\_averages = np.array(list(map(np.average, y\_table)))  
 variation\_beta\_s = average\_variation/N/m  
 standard\_deviation\_beta\_s = math.sqrt(variation\_beta\_s)  
 x\_vals = [x\_i(i) for i in range(11)]  
 *# coefficients\_beta\_s = np.array([round(np.average(y\_averages\*x\_vals[i]),3) for i in range(len(x\_vals))])* t\_i = np.array([abs(beta\_coefficients[i])/standard\_deviation\_beta\_s for i in range(len(beta\_coefficients))])  
 f3 = (m-1)\*N  
 q = 0.05  
  
 t = get\_student\_value(f3, q)  
 importance = [True if el > t else False for el in list(t\_i)]  
  
 *# print result data* print(**"Оцінки коефіцієнтів βs: "** + **", "**.join(list(map(lambda x: str(round(float(x), 3)), beta\_coefficients))))  
 print(**"Коефіцієнти ts: "** + **", "**.join(list(map(lambda i: **"{:.2f}"**.format(i), t\_i))))  
 print(**"f3 = {}; q = {}; tтабл = {}"**.format(f3, q, t))  
 beta\_i = [**"β0"**, **"β1"**, **"β2"**, **"β3"**, **"β12"**, **"β13"**, **"β23"**, **"β123"**, **"β11"**, **"β22"**, **"β33"**]  
 importance\_to\_print = [**"важливий"** if i else **"неважливий"** for i in importance]  
 to\_print = map(lambda x: x[0] + **" "** + x[1], zip(beta\_i, importance\_to\_print))  
 x\_i\_names = list(compress([**""**, **"x1"**, **"x2"**, **"x3"**, **"x12"**, **"x13"**, **"x23"**, **"x123"**, **"x1^2"**, **"x2^2"**, **"x3^2"**], importance))  
 betas\_to\_print = list(compress(beta\_coefficients, importance))  
 print(\*to\_print, sep=**"; "**)  
 equation = **" "**.join([**""**.join(i) for i in zip(list(map(lambda x: **"{:+.2f}"**.format(x), betas\_to\_print)),x\_i\_names)])  
 print(**"Рівняння регресії без незначимих членів: y = "** + equation)  
 return importance  
  
  
def calculate\_theoretical\_y(x\_table, b\_coefficients, importance):  
 x\_table = [list(compress(row, importance)) for row in x\_table]  
 b\_coefficients = list(compress(b\_coefficients, importance))  
 y\_vals = np.array([sum(map(lambda x, b: x\*b, row, b\_coefficients)) for row in x\_table])  
 return y\_vals  
  
  
def fisher\_criteria(m, N, d, naturalized\_x\_table, y\_table, b\_coefficients, importance):  
 f3 = (m - 1) \* N  
 f4 = N - d  
 q = 0.05  
  
 theoretical\_y = calculate\_theoretical\_y(naturalized\_x\_table, b\_coefficients, importance)  
 theoretical\_values\_to\_print = list(zip(map(lambda x: **"x1 = {0[1]}, x2 = {0[2]}, x3 = {0[3]}"**.format(x),naturalized\_x\_table),theoretical\_y))  
  
 y\_averages = np.array(list(map(np.average, y\_table)))  
 s\_ad = m/(N-d)\*(sum((theoretical\_y-y\_averages)\*\*2))  
 y\_variations = np.array(list(map(np.var, y\_table)))  
 s\_v = np.average(y\_variations)  
 f\_p = float(s\_ad/s\_v)  
 f\_t = get\_fisher\_value(f3, f4, q)  
  
 print(**"**\n**Перевірка адекватності моделі за критерієм Фішера: m = {}, "  
 "N = {} для таблиці y\_table"**.format(m, N))  
 print(**"Теоретичні значення y для різних комбінацій факторів:"**)  
 print(**"**\n**"**.join([**"{arr[0]}: y = {arr[1]}"**.format(arr=el) for el in theoretical\_values\_to\_print]))  
 print(**"Fp = {}, Ft = {}"**.format(f\_p, f\_t))  
 print(**"Fp < Ft => модель адекватна"** if f\_p < f\_t else **"Fp > Ft => модель неадекватна"**)  
 return True if f\_p < f\_t else False  
  
  
def m\_ij(\*arrays):  
 return np.average(reduce(lambda accum, el: accum\*el, arrays))  
  
  
def get\_cochran\_value(f1, f2, q):  
 partResult1 = q / f2 *# (f2 - 1)* params = [partResult1, f1, (f2 - 1) \* f1]  
 fisher = f.isf(\*params)  
 result = fisher/(fisher + (f2 - 1))  
 return Decimal(result).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
  
def get\_student\_value(f3, q):  
 return Decimal(abs(t.ppf(q/2,f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
  
def get\_fisher\_value(f3,f4, q):  
 return Decimal(abs(f.isf(q,f4,f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()

**Результати роботи програми:**

****

****

**Висновки**

У ході лабораторної роботи було досліджено трьохфакторний експеримент з рівнянням регресії з квадратичними членами, використано критерій Кохрена для перевірки дисперсій на однорідність, критерій Стьюдента для перевірки нуль-гіпотези та критерій Фішера перевірки адекватності гіпотези. Можна зробити висновок, що квадратичні члени підвищують точність апроксимації.  Розглянута модель дає результати, що практично співпадають з модельованими. При моделюванні використано центральний ортогональний композиційний план, оскільки дробового та повного факторного плану недостатньо для пошуку всіх невідомих коефіцієнтів рівняння регресії.