

Міністерство освіти та науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

з дисципліни «Методи оптимізації та планування»

на тему: «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії.»

Виконав:

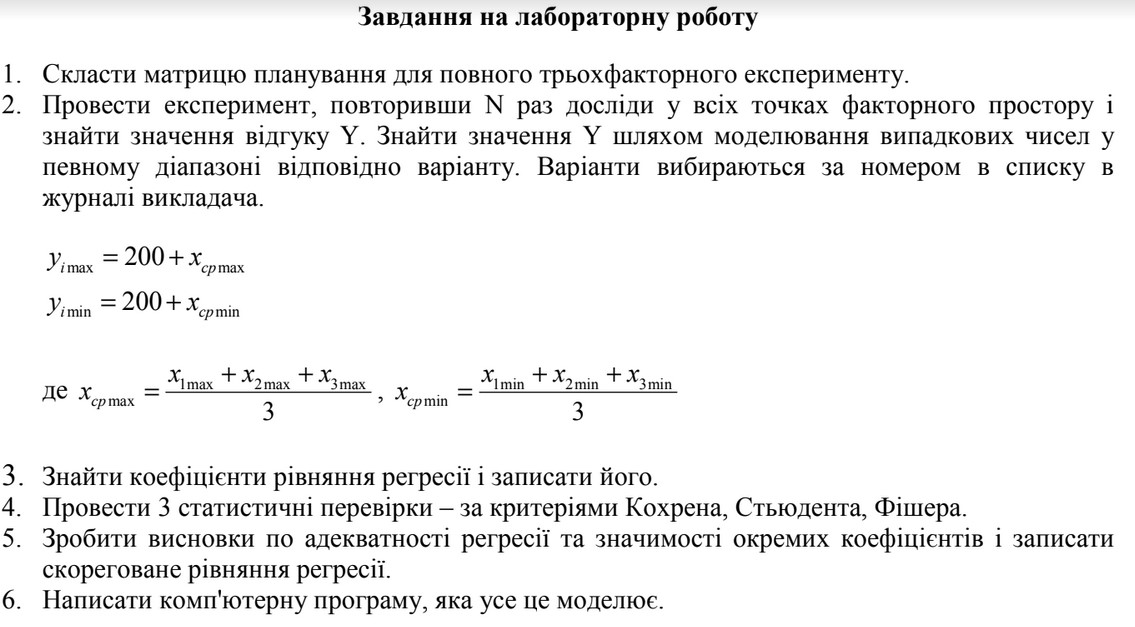
студент групи ІО-92

Костюк А.В

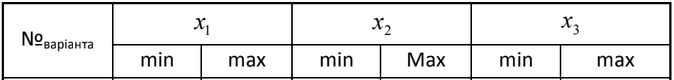
Перевірив: асистент

Регіда П. Г.

Київ – 2021



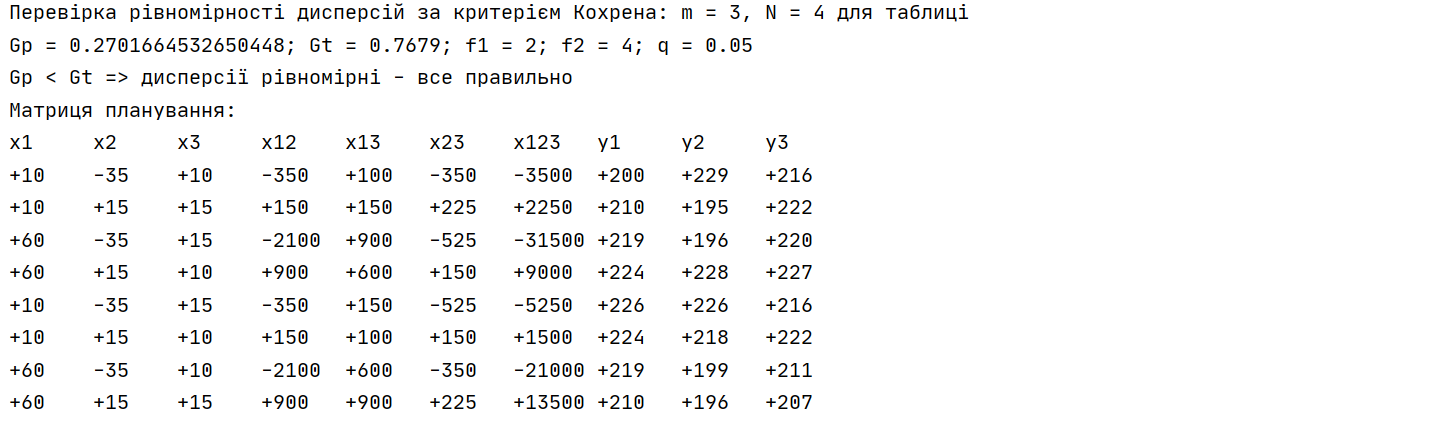
**Варіант:**

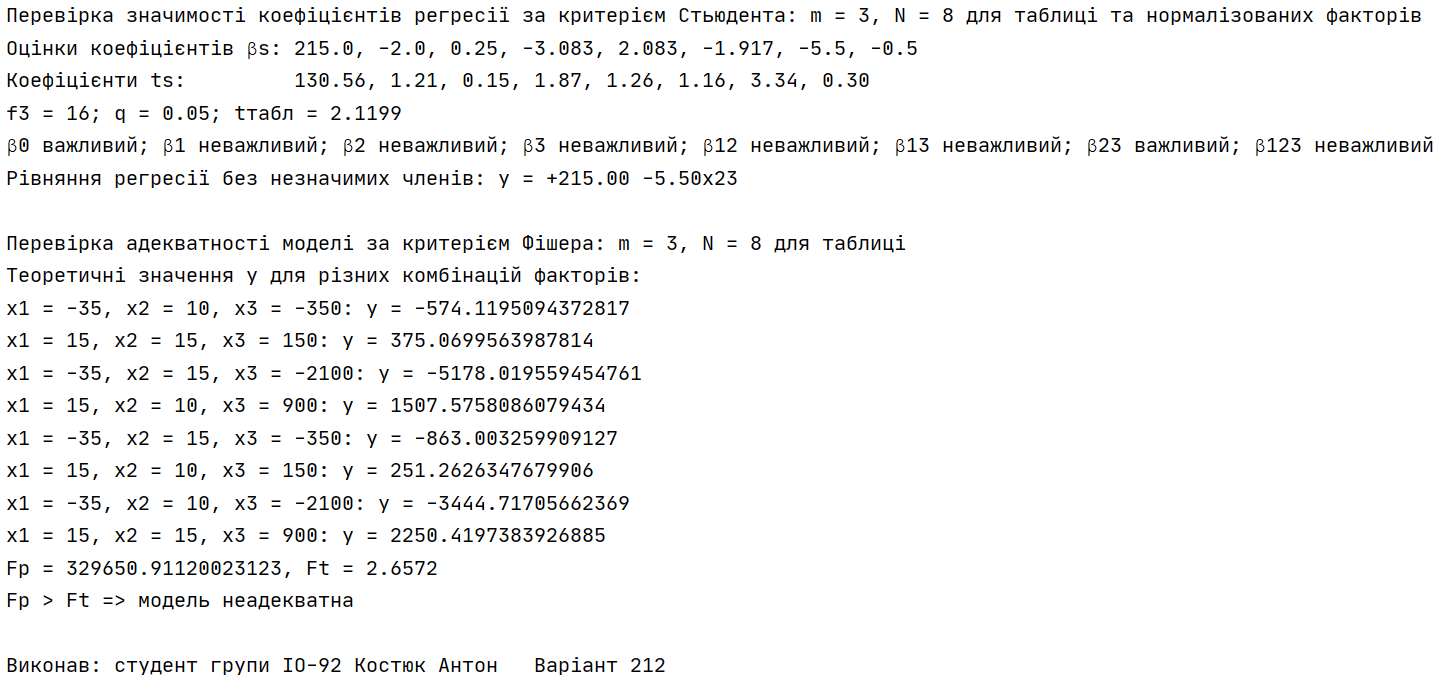


**Код програми:**

import numpy as np  
import random as r  
from functools import reduce  
from \_pydecimal import Decimal  
from itertools import compress  
import math  
from scipy.stats import f, t  
  
*# x1 x2 x3 x12 x13 x23 x123*norm\_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, -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]]  
  
zero\_factor = [+1]\*8  
  
factors\_table = [[10, -35, 10, -350, 100, -350, -3500],  
 [10, 15, 15, 150, 150, 225, 2250],  
 [60, -35, 15, -2100, 900, -525, -31500],  
 [60, 15, 10, 900, 600, 150, 9000],  
  
 [10, -35, 15, -350, 150, -525, -5250],  
 [10, 15, 10, 150, 100, 150, 1500],  
 [60, -35, 10, -2100, 600, -350, -21000],  
 [60, 15, 15, 900, 900, 225, 13500]]  
  
y\_min = 195  
y\_max = 230  
  
M = 3  
N = 8  
y\_arr = [[r.randint(y\_min, y\_max) for \_ in range(M)] for j in range(N)]  
  
x1 = np.array(list(zip(\*factors\_table))[0])  
x2 = np.array(list(zip(\*factors\_table))[1])  
x3 = np.array(list(zip(\*factors\_table))[2])  
yi = np.array([np.average(i) for i in y\_arr])  
  
def m\_ij(\*arrays):  
 return np.average(reduce(lambda accum, el: accum\*el, arrays))  
  
coeffs = [[N, m\_ij(x1), m\_ij(x2), m\_ij(x3), m\_ij(x1\*x2), m\_ij(x1\*x3), m\_ij(x2\*x3), m\_ij(x1\*x2\*x3)],  
  
 [m\_ij(x1), m\_ij(x1\*\*2), m\_ij(x1\*x2), m\_ij(x1\*x3), m\_ij(x1\*\*2\*x2), m\_ij(x1\*\*2\*x3), m\_ij(x1\*x2\*x3), m\_ij(x1\*\*2\*x2\*x3)],  
 [m\_ij(x2), m\_ij(x1\*x2), m\_ij(x2\*\*2), m\_ij(x2\*x3), m\_ij(x1\*x2\*\*2), m\_ij(x1\*x2\*x3), m\_ij(x2\*\*2\*x3), m\_ij(x1\*x2\*\*2\*x3)],  
 [m\_ij(x3), m\_ij(x1\*x3), m\_ij(x2\*x3), m\_ij(x3\*\*2), m\_ij(x1\*x2\*x3), m\_ij(x1\*x3\*\*2), m\_ij(x2\*x3\*\*2), m\_ij(x1\*x2\*x3\*\*2)],  
  
 [m\_ij(x1\*x2), m\_ij(x1\*\*2\*x2), m\_ij(x1\*x2\*\*2), m\_ij(x1\*x2\*x3), m\_ij(x1\*\*2\*x2\*\*2), m\_ij(x1\*\*2\*x2\*x3), m\_ij(x1\*x2\*\*2\*x3), m\_ij(x1\*\*2\*x2\*\*2\*x3)],  
 [m\_ij(x1\*x3), m\_ij(x1\*\*2\*x3), m\_ij(x1\*x2\*x3), m\_ij(x1\*x3\*\*2), m\_ij(x1\*\*2\*x2\*x3), m\_ij(x1\*\*2\*x3\*\*2), m\_ij(x1\*x2\*x3\*\*2), m\_ij(x1\*\*2\*x2\*x3\*\*2)],  
 [m\_ij(x2\*x3), m\_ij(x1\*x2\*x3), m\_ij(x2\*\*2\*x3), m\_ij(x2\*x3\*\*2), m\_ij(x1\*x2\*\*2\*x3), m\_ij(x1\*x2\*x3\*\*2), m\_ij(x2\*\*2\*x3\*\*2), m\_ij(x1\*x2\*\*2\*x3\*\*2)],  
  
 [m\_ij(x1\*x2\*x3), m\_ij(x1\*\*2\*x2\*x3), m\_ij(x1\*x2\*\*2\*x3), m\_ij(x1\*x2\*x3\*\*2), m\_ij(x1\*\*2\*x2\*\*2\*x3), m\_ij(x1\*\*2\*x2\*x3\*\*2), m\_ij(x1\*x2\*\*2\*x3\*\*2), m\_ij(x1\*\*2\*x2\*\*2\*x3\*\*2)]]  
  
free\_vals = [m\_ij(yi), m\_ij(yi\*x1), m\_ij(yi\*x2), m\_ij(yi\*x3), m\_ij(yi\*x1\*x2), m\_ij(yi\*x1\*x3), m\_ij(yi\*x2\*x3), m\_ij(yi\*x1\*x2\*x3)]  
  
natural\_bi = np.linalg.solve(coeffs, free\_vals)  
  
natural\_x1 = np.array(list(zip(\*norm\_factors\_table))[0])  
natural\_x2 = np.array(list(zip(\*norm\_factors\_table))[1])  
natural\_x3 = np.array(list(zip(\*norm\_factors\_table))[2])  
  
norm\_bi = [m\_ij(yi),  
 m\_ij(yi\*natural\_x1),  
 m\_ij(yi\*natural\_x2),  
 m\_ij(yi\*natural\_x3),  
 m\_ij(yi\*natural\_x1\*natural\_x2),  
 m\_ij(yi\*natural\_x1\*natural\_x3),  
 m\_ij(yi\*natural\_x2\*natural\_x3),  
 m\_ij(yi\*natural\_x1\*natural\_x2\*natural\_x3)]  
  
def cochran\_criteria(m, N, y\_table):  
 print(**"Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N = {} для таблиці"**.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, normalized\_x\_table: **"with zero factor!"**):  
 print(**"**\n**Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = {}, N = {} "  
 "для таблиці 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\_i = np.array([[el[i] for el in normalized\_x\_table] for i in range(len(normalized\_x\_table))])  
 coefficients\_beta\_s = np.array([round(np.average(y\_averages\*x\_i[i]),3) for i in range(len(x\_i))])  
 print(**"Оцінки коефіцієнтів βs: "** + **", "**.join(list(map(str,coefficients\_beta\_s))))  
 t\_i = np.array([abs(coefficients\_beta\_s[i])/standard\_deviation\_beta\_s for i in range(len(coefficients\_beta\_s))])  
 print(**"Коефіцієнти ts: "** + **", "**.join(list(map(lambda i: **"{:.2f}"**.format(i), t\_i))))  
 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(**"f3 = {}; q = {}; tтабл = {}"**.format(f3, q, t))  
 beta\_i = [**"β0"**, **"β1"**, **"β2"**, **"β3"**, **"β12"**, **"β13"**, **"β23"**, **"β123"**]  
 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"**], importance))*#'[""] + list(compress(["x{}".format(i) for i in range(N)], importance))[1:]* betas\_to\_print = list(compress(coefficients\_beta\_s, 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):  
 print(**"**\n**Перевірка адекватності моделі за критерієм Фішера: m = {}, "  
 "N = {} для таблиці"**.format(m, N))  
 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))  
 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]))  
 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(**"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\_\_()  
  
  
while not cochran\_criteria(M, 4, y\_arr):  
 M += 1  
 y\_table = [[r.randint(y\_min, y\_max) for \_ in range(M)] for j in range(N)]  
print(**"Матриця планування:"**)  
labels\_table = list(map(lambda x: x.ljust(6),  
 [**"x1"**, **"x2"**, **"x3"**, **"x12"**, **"x13"**, **"x23"**, **"x123"**] + [**"y{}"**.format(i + 1) for i in range(M)]))  
rows\_table = [list(factors\_table[i]) + list(y\_arr[i]) for i in range(N)]  
rows\_normalized\_table = [factors\_table[i] + list(y\_arr[i]) for i in range(N)]  
print((**" "**).join(labels\_table))  
print(**"**\n**"**.join([**" "**.join(map(lambda j: **"{:<+6}"**.format(j), rows\_table[i])) for i in range(len(rows\_table))]))  
print(**"**\t**"**)  
norm\_factors\_table\_zero\_factor = [[+1] + i for i in norm\_factors\_table]  
importance = student\_criteria(M, N, y\_arr, norm\_factors\_table\_zero\_factor)  
fisher\_criteria(M, N, 1, factors\_table, y\_arr, natural\_bi, importance)  
print(**"**\n**Виконав: студент групи ІО-92 Костюк Антон Варіант 212"**)

**Результати виконання:**





**Висновки:**

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії та рівняння регресії з ефектом взаємодії, складено матрицю планування експерименту, було визначено коефіцієнти рівняння регресії(натуралізовані та нормовані), виконано перевірку правильності розрахунку коефіцієнтів рівняння регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера). При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості q = 0.05.