

Міністерство освіти та науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №6

з дисципліни «Методи оптимізації та планування»

на тему: «Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

Виконав:

студент 2-го курсу ФІОТ групи ІО-92

Костюк А.В.

Перевірив:

асистент Регіда П. Г.

Київ – 2021

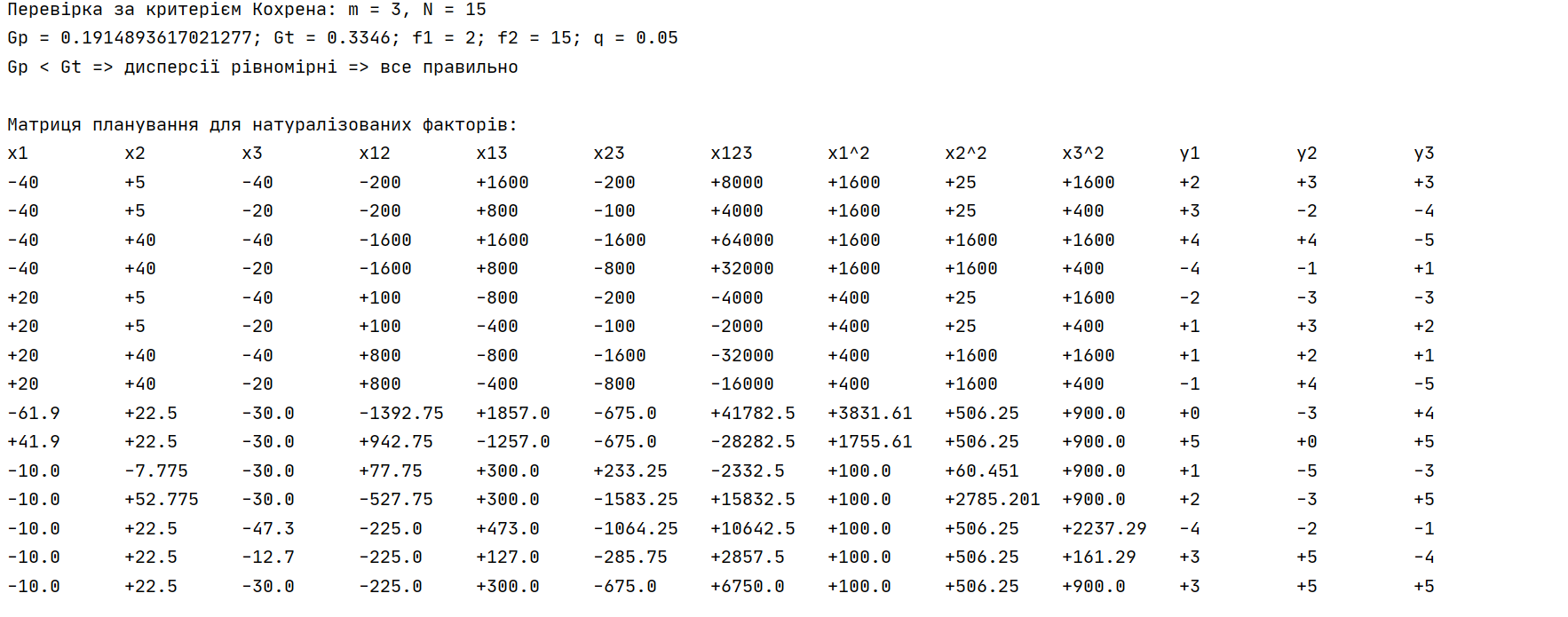
**Варіант:**

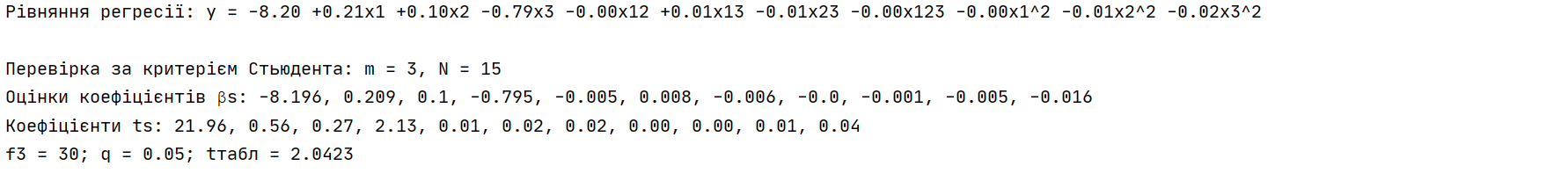
Варіант №212

**Код програми:**

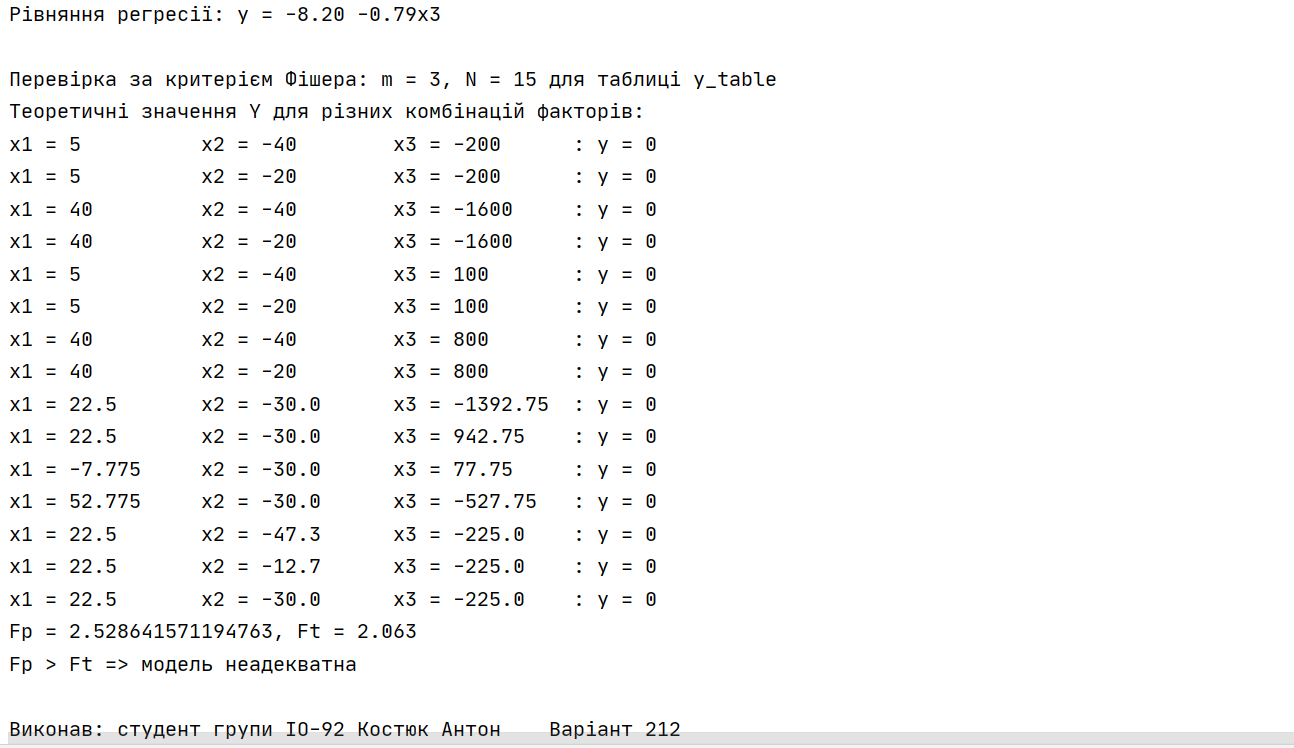
import math  
import random  
from \_decimal import Decimal  
from scipy.stats import f, t  
import numpy  
from itertools import compress  
from functools import reduce  
import timeit  
  
xmin = [-40, 5, -40]  
xmax = [20, 40, -20]  
norm\_plan\_raw = [[-1, -1, -1],  
 [-1, +1, +1],  
 [+1, -1, +1],  
 [+1, +1, -1],  
 [-1, -1, +1],  
 [-1, +1, -1],  
 [+1, -1, -1],  
 [+1, +1, +1],  
 [-1.73, 0, 0],  
 [+1.73, 0, 0],  
 [0, -1.73, 0],  
 [0, +1.73, 0],  
 [0, 0, -1.73],  
 [0, 0, +1.73]]  
  
x0 = [(xmax[\_] + xmin[\_])/2 for \_ in range(3)]  
dx = [xmax[\_] - x0[\_] for \_ in range(3)]  
  
natur\_plan\_raw = [[xmin[0], xmin[1], xmin[2]],  
 [xmin[0], xmin[1], xmax[2]],  
 [xmin[0], xmax[1], xmin[2]],  
 [xmin[0], xmax[1], xmax[2]],  
 [xmax[0], xmin[1], xmin[2]],  
 [xmax[0], xmin[1], xmax[2]],  
 [xmax[0], xmax[1], xmin[2]],  
 [xmax[0], xmax[1], xmax[2]],  
 [-1.73\*dx[0]+x0[0], x0[1], x0[2]],  
 [1.73\*dx[0]+x0[0], x0[1], x0[2]],  
 [x0[0], -1.73\*dx[1]+x0[1], x0[2]],  
 [x0[0], 1.73\*dx[1]+x0[1], x0[2]],  
 [x0[0], x0[1], -1.73\*dx[2]+x0[2]],  
 [x0[0], x0[1], 1.73\*dx[2]+x0[2]],  
 [x0[0], x0[1], x0[2]]]  
  
  
def equation\_of\_regression(x1, x2, x3, cef, importance=[] \* 11):  
 factors\_array = [1, x1, x2, x3, x1 \* x2, x1 \* x3, x2 \* x3, x1 \* x2 \* x3, x1 \*\* 2, x2 \*\* 2, x3 \*\* 2]  
 return sum([el[0] \* el[1] for el in compress(zip(cef, factors\_array), importance)])  
  
  
def func(x1, x2, x3):  
 coeffs = [5.4, 2.4, 7.3, 9.6, 2.5, 0.2, 8.2, 1.7, 0.7, 0.6, 9.3]  
 return equation\_of\_regression(x1, x2, x3, coeffs)  
  
  
def generate\_factors\_table(raw\_array):  
 raw\_list = [row + [row[0] \* row[1], row[0] \* row[2], row[1] \* row[2], row[0] \* row[1] \* row[2]] + list(  
 map(lambda x: x \*\* 2, row)) for row in raw\_array]  
 return list(map(lambda row: list(map(lambda el: round(el, 3), row)), raw\_list))  
  
  
def generate\_y(m, factors\_table):  
 return [[round(func(row[0], row[1], row[2]) + random.randint(-5, 5), 3) for \_ in range(m)] for row in factors\_table]  
  
  
def print\_matrix(m, N, factors, y\_vals, additional\_text=**":"**):  
 labels\_table = list(map(lambda x: x.ljust(10),  
 [**"x1"**, **"x2"**, **"x3"**, **"x12"**, **"x13"**, **"x23"**, **"x123"**, **"x1^2"**, **"x2^2"**, **"x3^2"**] + [  
 **"y{}"**.format(i + 1) for i in range(m)]))  
 rows\_table = [list(factors[i]) + list(y\_vals[i]) for i in range(N)]  
 print(**"**\n**Матриця планування"** + additional\_text)  
 print(**" "**.join(labels\_table))  
 print(**"**\n**"**.join([**" "**.join(map(lambda j: **"{:<+10}"**.format(j), rows\_table[i])) for i in range(len(rows\_table))]))  
 print(**"**\t**"**)  
  
  
def print\_equation(coeffs, importance=[True] \* 11):  
 x\_i\_names = list(compress([**""**, **"x1"**, **"x2"**, **"x3"**, **"x12"**, **"x13"**, **"x23"**, **"x123"**, **"x1^2"**, **"x2^2"**, **"x3^2"**], importance))  
 coefficients\_to\_print = list(compress(coeffs, importance))  
 equation = **" "**.join(  
 [**""**.join(i) for i in zip(list(map(lambda x: **"{:+.2f}"**.format(x), coefficients\_to\_print)), x\_i\_names)])  
 print(**"Рівняння регресії: y = "** + equation)  
  
  
def set\_factors\_table(factors\_table):  
 def x\_i(i):  
 with\_null\_factor = list(map(lambda x: [1] + x, generate\_factors\_table(factors\_table)))  
 res = [row[i] for row in with\_null\_factor]  
 return numpy.array(res)  
  
 return x\_i  
  
  
def m\_ij(\*arrays):  
 return numpy.average(reduce(lambda accum, el: accum \* el, list(map(lambda el: numpy.array(el), arrays))))  
  
  
def find\_coefficients(factors, y\_vals):  
 x\_i = set\_factors\_table(factors)  
 coeffs = [[m\_ij(x\_i(column), x\_i(row)) for column in range(11)] for row in range(11)]  
 y\_numpy = list(map(lambda row: numpy.average(row), y\_vals))  
 free\_values = [m\_ij(y\_numpy, x\_i(i)) for i in range(11)]  
 beta\_coefficients = numpy.linalg.solve(coeffs, free\_values)  
 return list(beta\_coefficients)  
  
  
def cochran\_criteria(m, N, y\_table):  
 def get\_cochran\_value(f1, f2, q):  
 partResult1 = q / f2  
 params = [partResult1, f1, (f2 - 1) \* f1]  
 fisher = f.isf(\*params)  
 result = fisher / (fisher + (f2 - 1))  
 return Decimal(result).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
 print(**"Перевірка за критерієм Кохрена: m = {}, N = {}"**.format(m, N))  
 y\_variations = [numpy.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 => змінюємо значення m"**)  
 return False  
  
  
def student\_criteria(m, N, y\_table, beta\_coefficients):  
 def get\_student\_value(f3, q):  
 return Decimal(abs(t.ppf(q / 2, f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
 print(**"**\n**Перевірка за критерієм Стьюдента: m = {}, N = {} "**.format(m, N))  
 average\_variation = numpy.average(list(map(numpy.var, y\_table)))  
 variation\_beta\_s = average\_variation / N / m  
 standard\_deviation\_beta\_s = math.sqrt(variation\_beta\_s)  
 t\_i = [abs(beta\_coefficients[i]) / standard\_deviation\_beta\_s for i in range(len(beta\_coefficients))]  
 f3 = (m - 1) \* N  
 q = 0.05  
 t\_our = get\_student\_value(f3, q)  
 importance = [True if el > t\_our 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\_our))  
 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))  
 print(\*to\_print, sep=**"; "**)  
 print\_equation(beta\_coefficients, importance)  
 return importance  
  
  
def fisher\_criteria(m, N, d, x\_table, y\_table, b\_coefficients, importance):  
 def get\_fisher\_value(f3, f4, q):  
 return Decimal(abs(f.isf(q, f4, f3))).quantize(Decimal(**'.0001'**)).\_\_float\_\_()  
  
 f3 = (m - 1) \* N  
 f4 = N - d  
 q = 0.05  
 theoretical\_y = numpy.array([equation\_of\_regression(row[0], row[1], row[2], b\_coefficients) for row in x\_table])  
 average\_y = numpy.array(list(map(lambda el: numpy.average(el), y\_table)))  
 s\_ad = m / (N - d) \* sum((theoretical\_y - average\_y) \*\* 2)  
 y\_variations = numpy.array(list(map(numpy.var, y\_table)))  
 s\_v = numpy.average(y\_variations)  
 f\_p = float(s\_ad / s\_v)  
 f\_t = get\_fisher\_value(f3, f4, q)  
 theoretical\_values\_to\_print = list(  
 zip(map(lambda x: **"x1 = {0[1]:<10} x2 = {0[2]:<10} x3 = {0[3]:<10}"**.format(x), x\_table), theoretical\_y))  
 print(**"**\n**Перевірка 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  
  
m = 3  
N = 15  
natural\_plan = generate\_factors\_table(natur\_plan\_raw)  
y\_arr = generate\_y(m, natur\_plan\_raw)  
  
while not cochran\_criteria(m, N, y\_arr):  
 m += 1  
 y\_arr = generate\_y(m, natural\_plan)  
  
print\_matrix(m, N, natural\_plan, y\_arr, **" для натуралізованих факторів:"**)  
coefficients = find\_coefficients(natural\_plan, y\_arr)  
print\_equation(coefficients)  
  
importance = student\_criteria(m, N, y\_arr, coefficients)  
  
d = len(list(filter(None, importance)))  
fisher\_criteria(m, N, d, natural\_plan, y\_arr, coefficients, importance)  
print(**"**\n**Виконав: студент групи ІО-92 Костюк Антон Варіант 212"**)

**Результат виконання програми:**





**β0 важливий; β1 неважливий; β2 неважливий; β3 важливий; β12 неважливий; β13 неважливий; β23 неважливий; β123 неважливий; β11 неважливий; β22 неважливий; β33 неважливий**



**Висновки:**

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії, рівняння регресії з ефектом взаємодії та рівняння регресії з квадратичними членами, складено матрицю планування експерименту, було визначено коефіцієнти рівнянь регресії (натуралізовані та нормовані), для форми з квадратичними членами - натуралізовані, виконано перевірку правильності розрахунку коефіцієнтів рівнянь регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера) для кожної форми рівняння регресії . При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів, при неадекватності і такого рівняння регресії було затосовано рівняння регресії з квадратичними членами.