Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 6

з дисципліни «Методи оптимізації та планування експерименту»

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

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІО-93
Довгаль Богдан
Варіант: 308

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

```
import math
import random
from _decimal import Decimal
from itertools import compress
from scipy.stats import f, t
import numpy
from functools import reduce
import matplotlib.pyplot as plot


def regression_equation(x1, x2, x3, coeffs, importance=[True] * 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(coeffs, factors_array),
importance)])


def func(x1, x2, x3):
    coeffs = [8.0, 5.3, 0.5, 5.6, 3.2, 0.7, 4.1, 8.9, 0.5, 1.5, 1.2]
    return regression_equation(x1, x2, x3, coeffs)

xmin = [-40, -70, -20]
xmax = [20, -10, 20]
x0 = [(xmax[_] + xmin[_])/2 for _ in range(3)]
dx = [xmax[_] - x0[_] for _ in range(3)]
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]]

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 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("\nМатриця 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 рівномірності дисперсій 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 - треба ще експериментів")
        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("\nПеревірка значимості коефіцієнтів регресії 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)
    # y = []
    # x = []
    # for i in range(len(list(t_i))):
    #     x.append(i)
    #     if t_i[i] > t_our:
    #         y.append(t_i[i])
    #     else:
    #         y.append(-t_i[i])
    #
    # plot.plot(x, y)
    # plot.grid(True)
    # plot.axis([0, 11, -11, 11])
    # plot.show()
    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([regression_equation(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("\nПеревірка 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


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)
```

Результати роботи програми:

```
C:\Users\...\venv\Scripts\python.exe C:/Users/.../kim/teorver/tabz/kjnjk.py
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.13747645951035778; Gt = 0.3346; f1 = 2; f2 = 15; q = 0.05
Gp < Gt => дисперсії рівномірні - все правильно

Матриця планування для натуралізованих факторів:
x1      x2      x3      x12     x13     x23     x123    x1^2    x2^2    x3^2    y1          y2          y3
-40     -70     -20     +2800   +800    +1400   -56000  +1600   +4900   +400    -474856.0   -474862.0   -474865.0
-40     -70     +20     +2800   -800    -1400   +56000  +1600   +4900   +400    +509567.0   +509561.0   +509559.0
-40     -10     -20     +400    +800    +200    -8000   +1600   +100    +400    -67430.0    -67428.0    -67426.0
-40     -10     +20     +400    -800    -200    +8000   +1600   +100    +400    +72431.0    +72429.0    +72434.0
+20     -70     -20     -1400   -400    +1400   +28000  +400    +4900   +400    +258182.0   +258176.0   +258173.0
+20     -70     +20     -1400   +400    -1400   -28000  +400    +4900   +400    -250924.0   -250922.0   -250919.0
+20     -10     -20     -200    -400    +200    +4000   +400    +100    +400    +36327.0    +36329.0    +36322.0
+20     -10     +20     -200    +400    -200    -4000   +400    +100    +400    -35726.0    -35734.0    -35725.0
-61.9   -40.0   +0.0    +2476.0 -0.0    -0.0    +0.0    +3831.61 +1600.0 +0.0    +11898.935  +11899.935  +11899.935
+41.9   -40.0   +0.0    -1676.0 +0.0    -0.0    -0.0    +1755.61 +1600.0 +0.0    -1876.325   -1876.325   -1873.325
-10.0   -91.9   +0.0    +919.0  -0.0    -0.0    +0.0    +100.0  +8445.61 +0.0    +15565.265  +15571.265  +15567.265
-10.0   +11.9   +0.0    -119.0  -0.0    +0.0    -0.0    +100.0  +141.61 +0.0    -154.435    -160.435    -162.435
-10.0   -40.0   -34.6   +400.0  +346.0  +1384.0 -13840.0 +100.0  +1600.0 +1197.16 -112346.568 -112354.568 -112347.568
-10.0   -40.0   +34.6   +400.0  -346.0  -1384.0 +13840.0 +100.0  +1600.0 +1197.16 +122551.752 +122556.752 +122550.752
-10.0   -40.0   +0.0    +400.0  -0.0    -0.0    +0.0    +100.0  +1600.0 +0.0    +3666.0     +3662.0     +3663.0

Рівняння регресії: y = +6.98 +5.30x1 +0.51x2 +5.59x3 +3.20x12 +0.70x13 +4.10x23 +8.90x123 +0.50x1^2 +1.50x2^2 +1.20x3^2

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 6.978, 5.302, 0.513, 5.59, 3.2, 0.703, 4.101, 8.9, 0.501, 1.5, 1.201
Коефіцієнти ts: 16.69, 12.68, 1.23, 13.37, 7.65, 1.68, 9.81, 21.29, 1.20, 3.59, 2.87
f3 = 30; q = 0.05; ttабл = 2.0423
β0 важливий; β1 важливий; β2 неважливий; β3 важливий; β12 важливий; β13 неважливий; β23 важливий; β123 важливий; β11 неважливий; β22 важливий; β33 важливий
Рівняння регресії: y = +6.98 +5.30x1 +5.59x3 +3.20x12 +4.10x23 +8.90x123 +1.50x2^2 +1.20x3^2
```

```
Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -70        x2 = -20        x3 = 2800       : y = -474860.5476031228
x1 = -70        x2 = 20         x3 = 2800       : y = 509562.55627247505
x1 = -10        x2 = -20        x3 = 400        : y = -67428.34111868826
x1 = -10        x2 = 20         x3 = 400        : y = 72430.76275690901
x1 = -70        x2 = -20        x3 = -1400      : y = 258177.6797929981
x1 = -70        x2 = 20         x3 = -1400      : y = -250921.21633140408
x1 = -10        x2 = -20        x3 = -200       : y = 36325.886277433
x1 = -10        x2 = 20         x3 = -200       : y = -35728.67651363721
x1 = -40.0      x2 = 0.0        x3 = 2476.0     : y = 11899.791568577843
x1 = -40.0      x2 = 0.0        x3 = -1676.0    : y = -1875.6608694664428
x1 = -91.9      x2 = 0.0        x3 = 919.0      : y = 15566.941323853445
x1 = 11.9       x2 = 0.0        x3 = -119.0     : y = -158.2572914086779
x1 = -40.0      x2 = -34.6      x3 = 400.0      : y = -112349.90625283601
x1 = -40.0      x2 = 34.6       x3 = 400.0      : y = 122553.27761861404
x1 = -40.0      x2 = 0.0        x3 = 400.0      : y = 3663.6677030362875
Fp = 0.18991085806175823, Ft = 2.3343
Fp < Ft => модель адекватна

Process finished with exit code 0
```