Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту
Лабораторна робота №6
«Проведення трьох факторного експерименту при використанні рівняння регресії з квадратичними членами»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІО-91
Сниченков Д. А.
варіант – 21

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Текст програми

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


# Ініціалізація змінних
x_min = [10, -30, -30]
x_max = [40, 45, -10]
x0 = [(x_max[_] + x_min[_]) / 2 for _ in range(3)]
dx = [x_max[_] - 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]]

natural_plan_raw = [[x_min[0], x_min[1], x_min[2]],
                    [x_min[0], x_min[1], x_max[2]],
                    [x_min[0], x_max[1], x_min[2]],
                    [x_min[0], x_max[1], x_max[2]],
                    [x_max[0], x_min[1], x_min[2]],
                    [x_max[0], x_min[1], x_max[2]],
                    [x_max[0], x_max[1], x_min[2]],
                    [x_max[0], x_max[1], x_max[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]]]


# Основні функції
def regression_equation(x1, x2, x3, coefficients, importance=[True] * 11):
    factors_array = [1, x1, x2, x3, x1 * x1, x2 * x2, x3 * x3, x1 * x2, x1 *
x3, x2 * x3, x1 * x2 * x3]
    return sum([el[0] * el[1] for el in compress(zip(coefficients,
factors_array), importance)])


def func(x1, x2, x3):
    coefficients = [5.9, 4.0, 3.5, 8.2, 4.3, 0.7, 9.3, 6.1, 0.5, 8.6, 3.1]
    return regression_equation(x1, x2, x3, coefficients)


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]


# Вивід результатів
def print_matrix(m, n, factors, y_values, 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_values[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(coefficients, 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(coefficients, 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_values):
    x_i = set_factors_table(factors)
    coefficients = [[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_values))
    free_values = [m_ij(y_numpy, x_i(i)) for i in range(11)]
    beta_coefficients = numpy.linalg.solve(coefficients, free_values)
    return list(beta_coefficients)


# Критерії
def cochran_criteria(m, n, y_table):
    def get_cochran_value(f1, f2, q):
        part_result1 = q / f2
```

```
        params = [part_result1, 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)
    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


def main():
    m = 3
    n = 15

    natural_plan = generate_factors_table(natural_plan_raw)
    y_arr = generate_y(m, natural_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)


if __name__ == "__main__":
    main()
```

Результат виконання програми

```
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.18367346938775503; 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
+10       -30       -30       -300      -300      +900      +9000     +100      +900      +900      +42787.9  +42781.9  +42783.9
+10       -30       -10       -300      -100      +300      +3000     +100      +900      +100      +11853.9  +11846.9  +11853.9
+10       +45       -30       +450      -300      -1350     -13500    +100      +2025     +900      -40686.1  -40686.1  -40695.1
+10       +45       -10       +450      -100      -450      -4500     +100      +2025     +100      -12230.1  -12225.1  -12227.1
+40       -30       -30       -1200     -1200     +900      +36000    +1600     +900      +900      +127116.9 +127110.9 +127113.9
+40       -30       -10       -1200     -400      +300      +12000    +1600     +900      +100      +40683.9  +40676.9  +40683.9
+40       +45       -30       +1800     -1200     -1350     -54000    +1600     +2025     +900      -151890.1 -151886.1 -151883.1
+40       +45       -10       +1800     -400      -450      -18000    +1600     +2025     +100      -39417.1  -39419.1  -39417.1
-0.95     +7.5      -20.0     -7.125    +19.0     -150.0    +142.5    +0.902    +56.25    +400.0    +2742.393 +2742.393 +2750.393
+50.95    +7.5      -20.0     +382.125  -1019.0   -150.0    -7642.5   +2595.903 +56.25    +400.0    -8164.582 -8165.582 -8165.582
+25.0     -57.375   -20.0     -1434.375 -500.0    +1147.5   +28687.5  +625.0    +3291.891 +400.0    +98253.973 +98252.973 +98253.973
+25.0     +72.375   -20.0     +1809.375 -500.0    -1447.5   -36187.5  +625.0    +5238.141 +400.0    -103570.152 -103573.152 -103575.152
+25.0     +7.5      -37.3     +187.5    -932.5    -279.75   -6993.75  +625.0    +56.25    +1391.29  -7912.813 -7918.813 -7914.813
+25.0     +7.5      -2.7      +187.5    -67.5     -20.25    -506.25   +625.0    +56.25    +7.29     +2267.157 +2271.157 +2272.157
+25.0     +7.5      -20.0     +187.5    -500.0    -150.0    -3750.0   +625.0    +56.25    +400.0    -5606.225 -5609.225 -5605.225

Рівняння регресії: y = +5.75 +4.02x1 +3.37x2 +8.25x3 +6.10x12 +0.51x13 +8.60x23 +3.10x123 +4.30x1^2 +0.70x2^2 +9.30x3^2

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 5.746, 4.023, 3.374, 8.249, 6.103, 0.505, 8.595, 3.1, 4.302, 0.7, 9.303
Коефіцієнти ts: 15.08, 10.56, 8.85, 21.65, 16.02, 1.33, 22.56, 8.14, 11.29, 1.84, 24.42
f3 = 30; q = 0.05; tтабл = 2.0423
β0 важливий; β1 важливий; β2 важливий; β3 важливий; β12 важливий; β13 неважливий; β23 важливий; β123 важливий; β11 важливий; β22 неважливий; β33 важливий
Рівняння регресії: y = +5.75 +4.02x1 +3.37x2 +8.25x3 +6.10x12 +8.60x23 +3.10x123 +4.30x1^2 +9.30x3^2
```

```
Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -30        x2 = -30        x3 = -300       : y = 90638.21051139172
x1 = -30        x2 = -10        x3 = -300       : y = 28546.97712975085
x1 = 45         x2 = -30        x3 = 450        : y = -117117.34275007878
x1 = 45         x2 = -10        x3 = 450        : y = -38607.11783506824
x1 = -30        x2 = -30        x3 = -1200      : y = 344443.9525428342
x1 = -30        x2 = -10        x3 = -1200      : y = 117472.74950988323
x1 = 45         x2 = -30        x3 = 1800       : y = -484315.68163949583
x1 = 45         x2 = -10        x3 = 1800       : y = -152032.4946507779
x1 = 7.5        x2 = -20.0      x3 = -7.125     : y = 4614.532783336242
x1 = 7.5        x2 = -20.0      x3 = 382.125    : y = -55023.844276124924
x1 = -57.375    x2 = -20.0      x3 = -1434.375  : y = 269761.065498366
x1 = 72.375     x2 = -20.0      x3 = 1809.375   : y = -324136.9172428479
x1 = 7.5        x2 = -37.3      x3 = 187.5      : y = -53066.670249438685
x1 = 7.5        x2 = -2.7       x3 = 187.5      : y = -418.0233503870959
x1 = 7.5        x2 = -20.0      x3 = 187.5      : y = -29314.75543472771
Fp = 20532922372.23509, Ft = 2.4205
Fp > Ft => модель неадекватна

Process finished with exit code 0
```