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 »** на тему

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

Виконала:
студентка II курсу ФІОТ
групи ІО-93
Дяченко Віта
У списку групи №9

Перевірив:
Регіда П. Г.

Київ – 2021

Мета роботи: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.

Завдання:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x1, x2, x3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ ; - ; 0 для 1, 2, 3.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$yi = f(x1, x2, x3) + random(10)-5,$$

де f(x1, x2, x3) вибирається по номеру в списку в журналі викладача.

4. Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів.
5. Зробити висновки по виконаній роботі.

Варіант:

309	-20	15	-35	10	10	20	4,3+8,4*x1+6,4*x2+5,4*x3+4,1*x1*x1+0,2*x2*x2+7,4*x3*x3+1,0*x1*x2+0,3*x1*x3+5,6*x2*x3+2,1*x1*x2*x3

Код програми:

```
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 = [4.3, 8.4, 6.4, 5.4, 4.1, 0.2, 7.4, 1.0, 0.3, 5.6, 2.1]
    return regression_equation(x1, x2, x3, coeffs)

xmin = [-20, -35, 10]
xmax = [15, -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)
```

Результат роботи:

```
C:\Users\38068\anaconda3\envs\tabs\python.exe C:/Users/38068/PycharmProjects/tabs/tab8_mope.py
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.20192307692307693; 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
-20     -35     +10     +700    -200    -350    +7000   +400    +1225   +100    +14091.3    +14094.3    +14094.3
-20     -35     +20     +700    -400    -700    +14000  +400    +1225   +400    +19151.3    +19154.3    +19150.3
-20     -10     +10     +200    -200    -100    +2000   +400    +100    +100    +2757.3     +2758.3     +2754.3
-20     -10     +20     +200    -400    -200    +4000   +400    +100    +400    +4660.3     +4661.3     +4663.3
+15     -35     +10     -525    +150    -350    -5250   +225    +1225   +100    -2862.7     -2869.7     -2861.7
+15     -35     +20     -525    +300    -700    -10500  +225    +1225   +400    -9988.7     -9995.7     -9993.7
+15     -10     +10     -150    +150    -100    -1500   +225    +100    +100    -1870.2     -1867.2     -1865.2
+15     -10     +20     -150    +300    -200    -3000   +225    +100    +400    -3388.2     -3388.2     -3394.2
-32.775 -22.5   +15.0   +737.438 -491.625 -337.5 +11061.562 +1074.201 +506.25 +225.0 +14786.981 +14783.981 +14782.981
+27.775 -22.5   +15.0   -624.938 +416.625 -337.5 -9374.062 +771.451 +506.25 +225.0 -10640.936 -10641.936 -10637.936
-2.5    -44.125 +15.0   +110.312 -37.5   -661.875 +1654.688 +6.25  +1947.016 +225.0 +8365.156  +8364.156  +8358.156
-2.5    -0.875  +15.0   +2.188   -37.5   -13.125 +32.812  +6.25   +0.766  +225.0  +473.519   +474.519   +478.519
-2.5    -22.5   +6.35   +56.25   -15.875 -142.875 +357.188 +6.25   +506.25 +40.322 +2323.505  +2321.505  +2325.505
-2.5    -22.5   +23.65  +56.25   -59.125 -532.125 +1330.312 +6.25  +506.25 +559.322 +1590.85  +1594.85  +1587.85
-2.5    -22.5   +15.0   +56.25   -37.5   -337.5  +843.75 +6.25   +506.25 +225.0  +1804.05  +1794.05  +1796.05

Рівняння регресії: y = +9.58 +8.14x1 +6.56x2 +4.77x3 +4.09x12 +0.22x13 +7.41x23 +1.00x123 +0.30x1^2 +5.60x2^2 +2.13x3^2

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 9.582, 8.143, 6.56, 4.767, 4.086, 0.217, 7.406, 1.001, 0.299, 5.605, 2.129
Коефіцієнти ts: 25.89, 22.00, 17.73, 12.88, 11.04, 0.59, 20.01, 2.70, 0.81, 15.14, 5.75
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 неважливий; β22 важливий; β33 важливий
Рівняння регресії: y = +9.58 +8.14x1 +6.56x2 +4.77x3 +4.09x12 +7.41x23 +1.00x123 +5.60x2^2 +2.13x3^2
```

```
Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 9.582, 8.143, 6.56, 4.767, 4.086, 0.217, 7.406, 1.001, 0.299, 5.605, 2.129
Коефіцієнти ts: 25.89, 22.00, 17.73, 12.88, 11.04, 0.59, 20.01, 2.70, 0.81, 15.14, 5.75
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 неважливий; β22 важливий; β33 важливий
Рівняння регресії: y = +9.58 +8.14x1 +6.56x2 +4.77x3 +4.09x12 +7.41x23 +1.00x123 +5.60x2^2 +2.13x3^2

Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -35        x2 = 10         x3 = 700        : y = 14094.492162418715
x1 = -35        x2 = 20         x3 = 700        : y = 19152.4925573551
x1 = -10        x2 = 10         x3 = 200        : y = 2756.922076087798
x1 = -10        x2 = 20         x3 = 200        : y = 4661.255802935656
x1 = -35        x2 = 10         x3 = -525       : y = -2864.2847972144323
x1 = -35        x2 = 20         x3 = -525       : y = -9992.951070366029
x1 = -10        x2 = 10         x3 = -150       : y = -1868.0215519674264
x1 = -10        x2 = 20         x3 = -150       : y = -3391.354490365023
x1 = -22.5      x2 = 15.0       x3 = 737.438    : y = 14783.722617643614
x1 = -22.5      x2 = 15.0       x3 = -624.938   : y = -10639.394857373542
x1 = -44.125    x2 = 15.0       x3 = 110.312    : y = 8361.41918268055
x1 = -0.875     x2 = 15.0       x3 = 2.188      : y = 476.53459847578546
x1 = -22.5      x2 = 6.35       x3 = 56.25      : y = 2322.7101320837523
x1 = -22.5      x2 = 23.65      x3 = 56.25      : y = 1591.929980760412
x1 = -22.5      x2 = 15.0       x3 = 56.25      : y = 1798.0503577515049
Fp = 0.707186875504937, Ft = 2.4205
Fp < Ft => модель адекватна
```

Висновок: В лабораторній роботі було проведено трьохфакторний експеримент. Знайдено адекватну модель - рівняння регресії, використовуючи рототабельний композиційний план.