



Міністерство освіти та науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

### Лабораторна робота №5

з дисципліни «Методи оптимізації та планування»

на тему: «Проведення трьохфакторного експерименту при використанні  
рівняння регресії з урахуванням квадратичних членів  
(центральний ортогональний композиційний план)»

Виконав:

студент 2-го курсу ФІОТ

групи ІО-92

Костюк А. В.

Перевірив:

асистент Регіда П. Г.

**Мета роботи:** Провести трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний

ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

**Варіант:**

|     |    |   |   |   |    |   |
|-----|----|---|---|---|----|---|
| 212 | -3 | 8 | 0 | 6 | -6 | 1 |
|-----|----|---|---|---|----|---|

**Код програми:**

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

raw_naturalized_factors_table = [[-3, 0, -6],
                                  [-3, 0, 1],
                                  [-3, 6, -6],
                                  [-3, 6, 1],

                                  [8, 0, -6],
                                  [8, 0, 1],
                                  [8, 6, -6],
                                  [8, 6, 1],

                                  [9.1825, 3, -2.5],
                                  [-4.1825, 3, -2.5],
                                  [2.5, 6.645, -2.5],
                                  [2.5, -0.645, -2.5],
                                  [2.5, 3, 1.7525],
                                  [2.5, 3, -6.7525],

                                  [2.5, 3, -2.5]]

raw_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],

                     [-1.215, 0, 0],
                     [+1.215, 0, 0],
                     [0, -1.215, 0],
                     [0, +1.215, 0],
                     [0, 0, -1.215],
                     [0, 0, +1.215],

                     [0, 0, 0]]
```

```

def generate_factors_table(raw_array):
    return [row + [row[0] * row[1], row[0] * row[2], row[1] * row[2], row[0] *
row[1] * row[2]]
        + list(map(lambda x: round(x ** 2, 5), row))
            for row in raw_array]

def x_i(i):
    try:
        assert i <= 10
    except:
        raise AssertionError("i must be smaller or equal 10")
    with_null_factor = list(map(lambda x: [1] + x,
generate_factors_table(raw_factors_table)))
    res = [row[i] for row in with_null_factor]
    return np.array(res)

def cochrans_criteria(m, N, y_table):
    print("Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N =
{} для таблиці y_table".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, beta_coefficients):
    print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента:
m = {}, N = {} "
        "для таблиці y_table 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_vals = [x_i(i) for i in range(11)]
    # coefficients_beta_s = np.array([round(np.average(y_averages*x_vals[i]),3)
for i in range(len(x_vals))])
    t_i = np.array([abs(beta_coefficients[i])/standard_deviation_beta_s for i in
range(len(beta_coefficients))])
    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("Оцінки коефіцієнтів  $\beta$ s: " + ", ".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))
    beta_i = [" $\beta_0$ ", " $\beta_1$ ", " $\beta_2$ ", " $\beta_3$ ", " $\beta_{12}$ ", " $\beta_{13}$ ", " $\beta_{23}$ ", " $\beta_{123}$ ", " $\beta_{11}$ ", " $\beta_{22}$ ",

```

```

"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))
    x_i_names = list(compress(["", "x1", "x2", "x3", "x12", "x13", "x23",
"x123", "x1^2", "x2^2", "x3^2"], importance))
    betas_to_print = list(compress(beta_coefficients, 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):
    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))

    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("\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 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__()

factors_table = generate_factors_table(raw_factors_table)
for row in factors_table:
    print(row)
naturalized_factors_table =
generate_factors_table(raw_naturalized_factors_table)
with_null_factor = list(map(lambda x: [1] + x, naturalized_factors_table))

m = 3
N = 15
ymin = 197
ymax = 205
y_arr = [[random.randint(ymin, ymax) for _ in range(m)] for _ in range(N)]
while not cochrans_criteria(m, N, y_arr):
    m+=1
    y_arr = [[random.randint(ymin, ymax) for _ in range(m)] for _ in range(N)]

y_i = np.array([np.average(row) for row in y_arr])

coefficients = [[m_ij(x_i(column)*x_i(row)) for column in range(11)] for row in
range(11)]

free_values = [m_ij(y_i, x_i(i)) for i in range(11)]

beta_coefficients = np.linalg.solve(coefficients, free_values)

importance = student_criteria(m, N, y_arr, beta_coefficients)
d = len(list(filter(None, importance)))
fisher_criteria(m, N, d, naturalized_factors_table, y_arr, beta_coefficients,
importance)
print("\nВиконав: студент групи ІО-92 Костюк Антон    Варіант 212")

```

## Результати виконання:

```

[-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, 1, 1, 1, 1]
[1, -1, -1, -1, -1, 1, 1, 1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[-1.215, 0, 0, -0.0, -0.0, 0, -0.0, 1.47623, 0, 0]
[1.215, 0, 0, 0.0, 0.0, 0, 0.0, 1.47623, 0, 0]
[0, -1.215, 0, -0.0, 0, -0.0, -0.0, 0, 1.47623, 0]
[0, 1.215, 0, 0.0, 0, 0.0, 0.0, 0, 1.47623, 0]
[0, 0, -1.215, 0, -0.0, -0.0, -0.0, 0, 0, 1.47623]
[0, 0, 1.215, 0, 0.0, 0.0, 0.0, 0, 0, 1.47623]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

```

Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15 для таблиці y\_table

Gr = 0.16228070175438597; Gt = 0.3346; f1 = 2; f2 = 15; q = 0.05

Gr < Gt => дисперсії рівномірні - все правильно

Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15 для таблиці y\_table та нормалізованих факторів

Оцінки коефіцієнтів  $\beta$ s: 200.988, -0.811, 0.435, -0.187, -0.542, -0.792, -0.542, 0.125, -1.309, 0.836, 0.611

Коефіцієнти ts: 733.60, 2.96, 1.59, 0.68, 1.98, 2.89, 1.98, 0.46, 4.78, 3.05, 2.23

f3 = 30; q = 0.05; табл = 2.0423

$\beta_0$  важливий;  $\beta_1$  важливий;  $\beta_2$  неважливий;  $\beta_3$  неважливий;  $\beta_{12}$  неважливий;  $\beta_{13}$  важливий;  $\beta_{23}$  неважливий;  $\beta_{123}$  неважливий;  $\beta_{11}$  важливий;  $\beta_{22}$  важливий;  $\beta_{33}$  важливий

Рівняння регресії без незначимих членів:  $y = +200.99 - 0.81x_1 - 0.79x_2 - 1.31x_1^2 + 0.84x_2^2 + 0.61x_3^2$

Перевірка адекватності моделі за критерієм Фішера:  $m = 3$ ,  $N = 15$  для таблиці  $y\_table$

Теоретичні значення  $y$  для різних комбінацій факторів:

$x_1 = 0$ ,  $x_2 = -6$ ,  $x_3 = 0$ :  $y = -572.8561118731329$

$x_1 = 0$ ,  $x_2 = 1$ ,  $x_3 = 0$ :  $y = -602.1278255552065$

$x_1 = 6$ ,  $x_2 = -6$ ,  $x_3 = -18$ :  $y = -596.3373803840623$

$x_1 = 6$ ,  $x_2 = 1$ ,  $x_3 = -18$ :  $y = -658.8590940661358$

$x_1 = 0$ ,  $x_2 = -6$ ,  $x_3 = 0$ :  $y = 1638.0124756436037$

$x_1 = 0$ ,  $x_2 = 1$ ,  $x_3 = 0$ :  $y = 1608.74076196153$

$x_1 = 6$ ,  $x_2 = -6$ ,  $x_3 = 48$ :  $y = 1614.5312071326746$

$x_1 = 6$ ,  $x_2 = 1$ ,  $x_3 = 48$ :  $y = 1552.009493450601$

$x_1 = 3$ ,  $x_2 = -2.5$ ,  $x_3 = 27.5475$ :  $y = 1842.525680538819$

$x_1 = 3$ ,  $x_2 = -2.5$ ,  $x_3 = -12.5475$ :  $y = -843.6796532940161$

$x_1 = 6.645$ ,  $x_2 = -2.5$ ,  $x_3 = 16.612499999999997$ :  $y = 457.67012199761535$

$x_1 = -0.645$ ,  $x_2 = -2.5$ ,  $x_3 = -1.6125$ :  $y = 506.3992251507025$

$x_1 = 3$ ,  $x_2 = 1.7525$ ,  $x_3 = 7.5$ :  $y = 486.66483563240854$

$x_1 = 3$ ,  $x_2 = -6.7525$ ,  $x_3 = 7.5$ :  $y = 542.429342756128$

$x_1 = 3$ ,  $x_2 = -2.5$ ,  $x_3 = 7.5$ :  $y = 499.4230136224015$

$F_p = 1452166.427905441$ ,  $F_t = 2.2107$

$F_p > F_t \Rightarrow$  модель неадекватна

Виконав: студент групи ІО-92 Костюк Антон      Варіант 212

## Висновок:

В даній лабораторній роботі проведено трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайдено рівняння регресії, яке буде адекватним для опису об'єкту.