



Міністерство освіти та науки України

Національний технічний університет України “Київський політехнічний інститут”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

з дисципліни «Методи оптимізації та планування»

на тему: «Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії.»

Виконав:

студент групи ІО-92

Костюк А.В

Перевірив:

асистент

Регіда П. Г.

Завдання на лабораторну роботу

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

$$y_{i\max} = 200 + x_{cp\max}$$

$$y_{i\min} = 200 + x_{cp\min}$$

$$\text{де } x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}, \quad x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$$

3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

Варіант:

№ варіанта	x_1		x_2		x_3	
	min	max	min	Max	min	max
212	10	60	-35	15	10	15

Код програми:

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

#
norm_factors_table = [
    [x1, x2, x3, x12, x13, x23, x123]
    for x1 in [-1, +1]
    for x2 in [-1, +1]
    for x3 in [-1, +1]
    for x12 in [-1, +1]
    for x13 in [-1, +1]
    for x23 in [-1, +1]
    for x123 in [-1, +1]
]

zero_factor = [+1]*8

factors_table = [
    [10, -35, 10, -350, 100, -350, -3500],
    [10, 15, 15, 150, 150, 225, 2250],
    [60, -35, 15, -2100, 900, -525, -31500],
    [60, 15, 10, 900, 600, 150, 9000],
    [10, -35, 15, -350, 150, -525, -5250],

```

```

        [10, 15, 10, 150, 100, 150, 1500],
        [60, -35, 10, -2100, 600, -350, -21000],
        [60, 15, 15, 900, 900, 225, 13500]]

y_min = 195
y_max = 230

M = 3
N = 8
y_arr = [[r.randint(y_min, y_max) for _ in range(M)] for j in range(N)]

x1 = np.array(list(zip(*factors_table))[0])
x2 = np.array(list(zip(*factors_table))[1])
x3 = np.array(list(zip(*factors_table))[2])
yi = np.array([np.average(i) for i in y_arr])

def m_ij(*arrays):
    return np.average(reduce(lambda accum, el: accum*el, arrays))

coeffs = [[N, m_ij(x1), m_ij(x2), m_ij(x3), m_ij(x1*x2),
m_ij(x1*x3), m_ij(x2*x3), m_ij(x1*x2*x3)],

[m_ij(x1), m_ij(x1**2), m_ij(x1*x2), m_ij(x1*x3), m_ij(x1**2*x2),
m_ij(x1**2*x3), m_ij(x1*x2*x3), m_ij(x1**2*x2*x3)],

[m_ij(x2), m_ij(x1*x2), m_ij(x2**2), m_ij(x2*x3), m_ij(x1*x2**2),
m_ij(x1*x2*x3), m_ij(x2**2*x3), m_ij(x1*x2**2*x3)],

[m_ij(x3), m_ij(x1*x3), m_ij(x2*x3), m_ij(x3**2), m_ij(x1*x2*x3),
m_ij(x1*x3**2), m_ij(x2*x3**2), m_ij(x1*x2*x3**2)],

[m_ij(x1*x2), m_ij(x1**2*x2), m_ij(x1*x2**2), m_ij(x1*x2*x3),
m_ij(x1**2*x2**2), m_ij(x1**2*x2*x3), m_ij(x1*x2**2*x3), m_ij(x1**2*x2**2*x3)],

[m_ij(x1*x3), m_ij(x1**2*x3), m_ij(x1*x2*x3), m_ij(x1*x3**2),
m_ij(x1**2*x2*x3), m_ij(x1**2*x3**2), m_ij(x1*x2*x3**2), m_ij(x1**2*x2*x3**2)],

[m_ij(x2*x3), m_ij(x1*x2*x3), m_ij(x2**2*x3), m_ij(x2*x3**2),
m_ij(x1*x2**2*x3), m_ij(x1*x2*x3**2), m_ij(x2**2*x3**2), m_ij(x1*x2**2*x3**2)],

[m_ij(x1*x2*x3), m_ij(x1**2*x2*x3), m_ij(x1*x2**2*x3),
m_ij(x1*x2*x3**2), m_ij(x1**2*x2**2*x3), m_ij(x1**2*x2*x3**2),
m_ij(x1*x2**2*x3**2), m_ij(x1**2*x2**2*x3**2)]]

free_vals = [m_ij(yi), m_ij(yi*x1), m_ij(yi*x2), m_ij(yi*x3), m_ij(yi*x1*x2),
m_ij(yi*x1*x3), m_ij(yi*x2*x3), m_ij(yi*x1*x2*x3)]

natural_bi = np.linalg.solve(coeffs, free_vals)

natural_x1 = np.array(list(zip(*norm_factors_table))[0])
natural_x2 = np.array(list(zip(*norm_factors_table))[1])
natural_x3 = np.array(list(zip(*norm_factors_table))[2])

norm_bi = [m_ij(yi),
m_ij(yi*natural_x1),
m_ij(yi*natural_x2),
m_ij(yi*natural_x3),
m_ij(yi*natural_x1*natural_x2),
m_ij(yi*natural_x1*natural_x3),
m_ij(yi*natural_x2*natural_x3),
m_ij(yi*natural_x1*natural_x2*natural_x3)]

def cochrans_criteria(m, N, y_table):
    print("Перевірка рівномірності дисперсій за критерієм Кохрена: m = {}, N = {} для таблиці".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, normalized_x_table: "with zero factor!"):
    print("\nПеревірка значимості коефіцієнтів регресії за критерієм Стьюдента:
m = {}, N = {} "
        "для таблиці 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_i = np.array([el[i] for el in normalized_x_table] for i in
range(len(normalized_x_table)))
    coefficients_beta_s = np.array([round(np.average(y_averages*x_i[i]),3) for i
in range(len(x_i))])
    print("Оцінки коефіцієнтів  $\beta$ s: " + ",
".join(list(map(str,coefficients_beta_s))))
    t_i = np.array([abs(coefficients_beta_s[i])/standard_deviation_beta_s for i
in range(len(coefficients_beta_s))])
    print("Коефіцієнти ts: " + ", ".join(list(map(lambda i:
"{:.2f}".format(i), t_i))))
    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("f3 = {}; q = {}; tтабл = {}".format(f3, q, t))
    beta_i = [" $\beta_0$ ", " $\beta_1$ ", " $\beta_2$ ", " $\beta_3$ ", " $\beta_{12}$ ", " $\beta_{13}$ ", " $\beta_{23}$ ", " $\beta_{123}$ "]
    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"], importance))#[''] + list(compress(["x{}".format(i) for i in range(N)],
importance))[1:]
    betas_to_print = list(compress(coefficients_beta_s, 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):

```

```

print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, "
      "N = {} для таблиці".format(m, N))
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))
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]))
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("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__()

while not cochrans_criteria(M, 4, y_arr):
    M += 1
    y_table = [[r.randint(y_min, y_max) for _ in range(M)] for j in range(N)]
    print("Матриця планування:")
    labels_table = list(map(lambda x: x.ljust(6),
        ["x1", "x2", "x3", "x12", "x13", "x23", "x123"] +
        ["y{}".format(i + 1) for i in range(M)]))
    rows_table = [list(factors_table[i]) + list(y_arr[i]) for i in range(N)]
    rows_normalized_table = [factors_table[i] + list(y_arr[i]) for i in range(N)]
    print((" ").join(labels_table))
    print("\n".join([" ".join(map(lambda j: "{:<+6}".format(j), rows_table[i])) for
i in range(len(rows_table))]))
    print("\t")
    norm_factors_table_zero_factor = [[+1] + i for i in norm_factors_table]
    importance = student_criteria(M, N, y_arr, norm_factors_table_zero_factor)
    fisher_criteria(M, N, 1, factors_table, y_arr, natural_bi, importance)
    print("\nВиконав: студент групи ІО-92 Костюк Антон Варіант 212")

```

Результати виконання:

Перевірка рівномірності дисперсій за критерієм Кохрена: $m = 3$, $N = 4$ для таблиці

$G_p = 0.2701664532650448$; $G_t = 0.7679$; $f_1 = 2$; $f_2 = 4$; $q = 0.05$

$G_p < G_t \Rightarrow$ дисперсії рівномірні - все правильно

Матриця планування:

x1	x2	x3	x12	x13	x23	x123	y1	y2	y3
+10	-35	+10	-350	+100	-350	-3500	+200	+229	+216
+10	+15	+15	+150	+150	+225	+2250	+210	+195	+222
+60	-35	+15	-2100	+900	-525	-31500	+219	+196	+220
+60	+15	+10	+900	+600	+150	+9000	+224	+228	+227
+10	-35	+15	-350	+150	-525	-5250	+226	+226	+216
+10	+15	+10	+150	+100	+150	+1500	+224	+218	+222
+60	-35	+10	-2100	+600	-350	-21000	+219	+199	+211
+60	+15	+15	+900	+900	+225	+13500	+210	+196	+207

Перевірка значимості коефіцієнтів регресії за критерієм Стюдента: $m = 3$, $N = 8$ для таблиці та нормалізованих факторів

Оцінки коефіцієнтів β s: 215.0, -2.0, 0.25, -3.083, 2.083, -1.917, -5.5, -0.5

Коефіцієнти ts: 130.56, 1.21, 0.15, 1.87, 1.26, 1.16, 3.34, 0.30

$f_3 = 16$; $q = 0.05$; $t_{\text{табл}} = 2.1199$

β_0 важливий; β_1 неважливий; β_2 неважливий; β_3 неважливий; β_{12} неважливий; β_{13} неважливий; β_{23} важливий; β_{123} неважливий

Рівняння регресії без незначимих членів: $y = +215.00 - 5.50x_{23}$

Перевірка адекватності моделі за критерієм Фішера: $m = 3$, $N = 8$ для таблиці

Теоретичні значення y для різних комбінацій факторів:

$x_1 = -35$, $x_2 = 10$, $x_3 = -350$: $y = -574.1195094372817$

$x_1 = 15$, $x_2 = 15$, $x_3 = 150$: $y = 375.0699563987814$

$x_1 = -35$, $x_2 = 15$, $x_3 = -2100$: $y = -5178.019559454761$

$x_1 = 15$, $x_2 = 10$, $x_3 = 900$: $y = 1507.5758086079434$

$x_1 = -35$, $x_2 = 15$, $x_3 = -350$: $y = -863.003259909127$

$x_1 = 15$, $x_2 = 10$, $x_3 = 150$: $y = 251.2626347679906$

$x_1 = -35$, $x_2 = 10$, $x_3 = -2100$: $y = -3444.71705662369$

$x_1 = 15$, $x_2 = 15$, $x_3 = 900$: $y = 2250.4197383926885$

$F_p = 329650.91120023123$, $F_t = 2.6572$

$F_p > F_t \Rightarrow$ модель неадекватна

Виконав: студент групи ІО-92 Костюк Антон Варіант 212

Висновки:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії та рівняння регресії з ефектом взаємодії, складено матрицю планування експерименту, було визначено коефіцієнти рівняння регресії (натуралізовані та нормовані), виконано перевірку правильності розрахунку коефіцієнтів рівняння регресії. Також було проведено 3 статистичні перевірки (використання критеріїв Кохрена, Стюдента та Фішера). При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості $q = 0.05$.