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*»

ВИКОНАВ:
студент ІІ курсу ФІОТ
групи ІО-92
Соболь Денис

ПЕРЕВІРИВ:
Регіда П. Г.

Київ – 2021

Лабораторна робота № 6

Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами

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

Завдання до лабораторної роботи:

1. Ознайомитися з теоретичними відомостями.
2. Вибрати з таблиці варіантів і записати в протокол інтервали значень x_1, x_2, x_3. Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1;+ l; - l; 0 для $\bar{x}_1$, $\bar{x}_2$, $\bar{x}_3$.
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + \text{random}(10) - 5,$$

де $f(x_1, x_2, x_3)$ 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.

Алгоритм отримання адекватної моделі рівняння регресії

1) *Вибір рівняння регресії (лінійна форма, рівняння з урахуванням ефекту взаємодії і з урахуванням квадратичних членів);*
2) *Вибір кількості повторів кожної комбінації (m = 2);*
3) *Складення матриці планування експерименту і вибір кількості рівнів (N)*
4) *Проведення експериментів;*
5) *Перевірка однорідності дисперсії. Якщо не однорідна – повертаємося на п. 2 і збільшуємо m на 1);*
6) *Розрахунок коефіцієнтів рівняння регресії. При розрахунку використовувати* ***натуральні*** *значення x_1, x_2 и x_3.*
7) *Перевірка нуль-гіпотези. Визначення значимих коефіцієнтів;*
8) *Перевірка адекватності моделі рівняння оригіналу. При неадекватності – повертаємося на п.1, змінивши при цьому рівняння регресії;*

Мій варіант: 218

№	X_1		X_2		X_3		$F(x_1,x_2,x_3)$
	min	max	min	max	min	max	
218	-20	30	-35	15	-20	5	5,4+8,1*x1+7,5*x2+9,7*x3+5,2*x1*x1+0,6*x2*x2+4,7*x3*x3+8,1*x1*x2+0,8*x1*x3+7,4*x2*x3+0,1*x1*x2*x3

Програмний код

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

xmin = [-20, -35, -20]
xmax = [30, 15, 5]
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]]

x0 = [(xmax[_] + xmin[_])/2 for _ in range(3)]
dx = [xmax[_] - x0[_] for _ in range(3)]

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 equation_of_regression(x1, x2, x3, cef, importance=[] * 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(cef,
factors_array), importance)])


def func(x1, x2, x3):
    coeffs = [5.4, 8.1, 7.5, 9.7, 5.2, 0.6, 4.7, 8.1, 0.8, 7.4,
0.1]
    return equation_of_regression(x1, x2, x3, coeffs)


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 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 => змінюємо
значення m")
        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 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("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([equation_of_regression(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 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)
```

Виконання програми

```
"E:\Semestr 4\MOPE\Lab6\venv\Scripts\python.exe" "E:/Semestr 4/MOPE/Lab6/main.py"
Перевірка за критерієм Кохрена: m = 3, N = 15
Gp = 0.1527494908350305; 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       -20       +700      +400      +700      -14000    +400      +1225     +400      +3        +4        +3
-20       -35       +5        +700      -100      -175      +3500     +400      +1225     +25       +1        -4        +1
-20       +15       -20       -300      +400      -300      +6000     +400      +225      +400      +0        +5        +0
-20       +15       +5        -300      -100      +75       -1500     +400      +225      +25       -5        +5        +0
+30       -35       -20       -1050     -600      +700      +21000    +900      +1225     +400      -3        -2        -2
+30       -35       +5        -1050     +150      -175      -5250     +900      +1225     +25       +2        +2        -5
+30       +15       -20       +450      -600      -300      -9000     +900      +225      +400      +4        -3        +4
+30       +15       +5        +450      +150      +75       +2250     +900      +225      +25       -3        +1        +0
-38.25    -10.0     -7.5      +382.5    +286.875  +75.0     -2868.75  +1463.062 +100.0    +56.25    -1        +5        -1
+48.25    -10.0     -7.5      -482.5    -361.875  +75.0     +3618.75  +2328.062 +100.0    +56.25    +4        +5        +1
+5.0      -53.25    -7.5      -266.25   -37.5     +399.375  +1996.875 +25.0     +2835.562 +56.25    -5        +3        +2
+5.0      +33.25    -7.5      +166.25   -37.5     -249.375  -1246.875 +25.0     +1105.562 +56.25    -4        +4        +5
+5.0       -10.0    -29.125   -50.0     -145.625  +291.25   +1456.25  +25.0     +100.0    +848.266  +1        +4        -2
+5.0       -10.0    +14.125   -50.0     +70.625   -141.25   -706.25   +25.0     +100.0    +199.516  -5        +0        -2
+5.0       -10.0    -7.5      -50.0     -37.5     +75.0     +375.0    +25.0     +100.0    +56.25    -1        +3        +5

Рівняння регресії: y = +1.38 +0.01x1 -0.01x2 -0.19x3 +0.00x12 +0.00x13 -0.00x23 -0.00x123 -0.00x1^2 -0.00x2^2 -0.01x3^2

Перевірка за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 1.381, 0.014, -0.01, -0.185, 0.0, 0.001, -0.0, -0.0, -0.0, -0.001, -0.007
Коефіцієнти ts: 3.44, 0.03, 0.02, 0.46, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.02
f3 = 30; q = 0.05; tтабл = 2.0423
β0 важливий; β1 неважливий; β2 неважливий; β3 неважливий; β12 неважливий; β13 неважливий; β23 неважливий; β123 неважливий; β11 неважливий; β22 неважливий; β33 неважливий
Рівняння регресії: y = +1.38
```

```
Перевірка за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення Y для різних комбінацій факторів:
x1 = -35        x2 = -20        x3 = 700        : y = 0
x1 = -35        x2 = 5          x3 = 700        : y = 0
x1 = 15         x2 = -20        x3 = -300       : y = 0
x1 = 15         x2 = 5          x3 = -300       : y = 0
x1 = -35        x2 = -20        x3 = -1050      : y = 0
x1 = -35        x2 = 5          x3 = -1050      : y = 0
x1 = 15         x2 = -20        x3 = 450        : y = 0
x1 = 15         x2 = 5          x3 = 450        : y = 0
x1 = -10.0      x2 = -7.5       x3 = 382.5      : y = 0
x1 = -10.0      x2 = -7.5       x3 = -482.5     : y = 0
x1 = -53.25     x2 = -7.5       x3 = -266.25    : y = 0
x1 = 33.25      x2 = -7.5       x3 = 166.25     : y = 0
x1 = -10.0      x2 = -29.125    x3 = -50.0      : y = 0
x1 = -10.0      x2 = 14.125     x3 = -50.0      : y = 0
x1 = -10.0      x2 = -7.5       x3 = -50.0      : y = 0
Fp = 1.4696683153913292, Ft = 2.0374
Fp < Ft => модель адекватна
```

Висновки:

У ході лабораторної роботи було досліджено трьохфакторний експеримент з рівнянням регресії з квадратичними членами, використано критерій Кохрена для перевірки дисперсій на однорідність, критерій Стьюдента для перевірки нуль-гіпотези та критерій Фішера перевірки адекватності гіпотези. Можна зробити висновок, що квадратичні члени підвищують точність апроксимації. Розглянута модель дає результати, що практично співпадають з модельованими. При моделюванні використано рототабельний композиційний план, оскільки дробового та повного факторного плану недостатньо для пошуку всіх невідомих коефіцієнтів рівняння регресії.