

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

ВИКОНАВ:
Студент II курсу ФІОТ
Групи ІВ-91
Каптур М.І. - 9112

ПЕРЕВІРИВ:
асистент
Регіда П.Г.

Київ 2021 р.

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

Варіант завдання:

Варіант	X ₁		X ₂		X ₃	
	min	max	min	max	min	max
112	-40	20	-35	15	20	25

$$f(x_1, x_2, x_3) = 2,2 + 1,6 \cdot x_1 + 9,2 \cdot x_2 + 9,5 \cdot x_3 + 0,8 \cdot x_1 \cdot x_1 + 0,7 \cdot x_2 \cdot x_2 + 6,5 \cdot x_3 \cdot x_3 + 0,2 \cdot x_1 \cdot x_2 + 0,9 \cdot x_1 \cdot x_3 + 8,7 \cdot x_2 \cdot x_3 + 9,1 \cdot x_1 \cdot x_2 \cdot x_3$$

Лістинг 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

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 = [2.2, 1.6, 9.2, 9.5, 0.8, 0.7, 6.5, 0.2, 0.9, 8.7, 9.1]
    return regression_equation(x1, x2, x3, coeffs)

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 = [[-40, -35, 20],
                  [-40, -35, 25],
                  [-40, 15, 20],
                  [-40, 15, 25],
                  [20, -35, 20],
                  [20, -35, 25],
                  [20, 15, 20],
                  [20, 15, 25],
                  [-61.9, 10, 22.5],
                  [-41.9, 10, 22.5],
                  [-10, -41.9, 22.5],
                  [-10, 61.9, 22.5],
```

```

[-10, 10, -29.4],
[-10, 10, 74.4],
[-10, 10, 22.5]]

```

```

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 перцепції: 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 cochrans_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 = numpy.array([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  $\beta$ 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 = {}; табл = {}".format(f3, q, t_our))
    beta_i = [" $\beta_0$ ", " $\beta_1$ ", " $\beta_2$ ", " $\beta_3$ ", " $\beta_{12}$ ", " $\beta_{13}$ ", " $\beta_{23}$ ", " $\beta_{123}$ ", " $\beta_{11}$ ", " $\beta_{22}$ ", " $\beta_{33}$ "]
    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

m = 3
N = 15
natural_plan = generate_factors_table(natur_plan_raw)
y_arr = generate_y(m, natur_plan_raw)
while not cochrans_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)

```

Висновок:

В даній лабораторній роботі я провів трьохфакторний експеримент і отримав адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план.