Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Методи оптимізації та планування експерименту

Лабораторна робота №6

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

Виконала:
студентка групи ІО-92
Шолотюк Ганна Сергіївна
Номер залікової книжки № 9229
Номер у списку – 21

Перевірив:
Регіда Павло Геннадійович

Київ 2021

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

Завдання на лабораторну роботу

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; +l; -l ; 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.

Таблиця варіантів

№ варіанту	x_1		x_2		x_3		$f(x_1, x_2, x_3)$
	min	max	min	max	min	max	
221	10	40	10	60	10	15	3,6+8,8*x1+4,8*x2+5,1*x3+5,0*x1*x1+0,9*x2*x2+7,6*x3*x3+2,2*x1*x2+0,4*x1*x3+3,2*x2*x3+7,1*x1*x2*x3

Код програми

```
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 = [10, 10, 10]
xmax = [40, 60, 15]
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 = [3.6, 8.8, 4.8, 5.1, 5.0, 0.9, 7.6, 2.2, 0.4, 3.2, 7.1]
    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 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)
    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:")

coefs = find_coefficients(natural_plan, y_arr)

print_equation(coefs)

importance = student_criteria(m, N, y_arr, coefs)
d = len(list(filter(None, importance)))

fisher_criteria(m, N, d, natural_plan, y_arr, coefs, importance)

if __name__ == "__main__":
    m = 3
    N = 15
    main(m, N)
```

Результати виконання програми

```
Перевірка за критерієм Кохрена: m = 3, N = 15
Gp = 0.1336898395721925; 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
+10       +10       +10       +100      +100      +100      +1000     +100      +100      +100      -1        -3        +3
+10       +10       +15       +100      +150      +150      +1500     +100      +100      +225      -3        +0        +1
+10       +60       +10       +600      +100      +600      +6000     +100      +3600     +100      -2        +5        -4
+10       +60       +15       +600      +150      +900      +9000     +100      +3600     +225      +0        +2        -4
+40       +10       +10       +400      +400      +100      +4000     +1600     +100      +100      +5        -5        +0
+40       +10       +15       +400      +600      +150      +6000     +1600     +100      +225      +0        -4        +4
+40       +60       +10       +2400     +400      +600      +24000    +1600     +3600     +100      +4        +2        -2
+40       +60       +15       +2400     +600      +900      +36000    +1600     +3600     +225      +3        +0        -4
-0.95     +35.0     +12.5     -33.25    -11.875   +437.5    -415.625  +0.902    +1225.0   +156.25   +1        +4        +1
+50.95    +35.0     +12.5     +1783.25  +636.875  +437.5    +22290.625 +2595.903 +1225.0   +156.25   +5        -1        -4
+25.0     -8.25     +12.5     -206.25   +312.5    -103.125  -2578.125 +625.0    +68.062   +156.25   -5        -2        +0
+25.0     +78.25    +12.5     +1956.25  +312.5    +978.125  +24453.125 +625.0   +6123.062 +156.25   -1        -2        -5
+25.0     +35.0     +8.175    +875.0    +204.375  +286.125  +7153.125 +625.0    +1225.0   +66.831   +3        +2        +1
+25.0     +35.0     +16.825   +875.0    +420.625  +588.875  +14721.875 +625.0   +1225.0   +283.081  -4        +3        -5
+25.0     +35.0     +12.5     +875.0    +312.5    +437.5    +10937.5  +625.0    +1225.0   +156.25   -5        -4        +4



Рівняння регресії: y = +19.87 -0.28x1 -0.01x2 -2.77x3 +0.00x12 +0.00x13 +0.00x23 -0.00x123 +0.00x1^2 -0.00x2^2 +0.10x3^2

Перевірка за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 19.867, -0.283, -0.012, -2.775, 0.003, 0.004, 0.002, -0.0, 0.004, -0.0, 0.101
Коефіцієнти ts: 46.23, 0.66, 0.03, 6.46, 0.01, 0.01, 0.01, 0.00, 0.01, 0.00, 0.23
f3 = 30; q = 0.05; tтабл = 2.0423
β0 важливий; β1 неважливий; β2 неважливий; β3 важливий; β12 неважливий; β13 неважливий; β23 неважливий; β123 неважливий; β11 неважливий;
Рівняння регресії: y = +19.87 -2.77x3
```

```
Перевірка за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення Y для різних комбінацій факторів:
x1 = 10         x2 = 10         x3 = 100       : y = 0
x1 = 10         x2 = 15         x3 = 100       : y = 0
x1 = 60         x2 = 10         x3 = 600       : y = 0
x1 = 60         x2 = 15         x3 = 600       : y = 0
x1 = 10         x2 = 10         x3 = 400       : y = 0
x1 = 10         x2 = 15         x3 = 400       : y = 0
x1 = 60         x2 = 10         x3 = 2400      : y = 0
x1 = 60         x2 = 15         x3 = 2400      : y = 0
x1 = 35.0       x2 = 12.5       x3 = -33.25    : y = 0
x1 = 35.0       x2 = 12.5       x3 = 1783.25   : y = 0
x1 = -8.25      x2 = 12.5       x3 = -206.25   : y = 0
x1 = 78.25      x2 = 12.5       x3 = 1956.25   : y = 0
x1 = 35.0       x2 = 8.175      x3 = 875.0     : y = 0
x1 = 35.0       x2 = 16.825     x3 = 875.0     : y = 0
x1 = 35.0       x2 = 12.5       x3 = 875.0     : y = 0
Fp = 0.8422459893048129, Ft = 2.063
Fp < Ft => модель адекватна

Process finished with exit code 0
```