Національний технічний університет України
«Київський політехнічний інститут ім. І. Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Методи організації та планування експериментів
Лабораторна робота №6
«Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами»

Виконав:
студент групи ІО-93
Руденко С.О.
Номер залікової книжки:
9327
Перевірив:
ст.вик.
ас. Регіда П.Г.

Київ 2021

Лабораторна робота № 6

Тема: Проведення трьохфакторного експерименту при використанні рівняння регресії з квадратичними членами.
Мета: Провести трьохфакторний експеримент і отримати адекватну модель – рівняння регресії, використовуючи рототабельний
композиційний план
Завдання: Провести експерименти і аналізуючи значення статистичних перевірок, отримати адекватну модель рівняння регресії. При розрахунках використовувати натуральні значення факторів.

Теоретичні основи:

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 x1, x2 и x3.
7) Перевірка нуль-гіпотези. Визначення значимих коефіцієнтів;
8) Перевірка адекватності моделі рівняння оригіналу. При неадекватності – повертаємося на п.1, змінивши при цьому рівняння регресії;

Варіант завдання:

324	-30	0	15	50	-30	35	3,6+8,2*x1+3,8*x2+8,7*x3+2,4*x1*x1+0,4*x2*x2+6,5*x3*x3+1,8*x1*x2+0,1*x1*x3+9,6*x2*x3+4,8*x1*x2*x3

Приклади роботи програми

```
Перевірка рівномірності дисперсій за критерієм Кохрена: m = 3, N = 15
Gp = 0.2309468822170901; 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
-40       -70       -20       +2800     +800      +1400     -56000    +1600     +4900     +400      -63112.6  -63111.6  -63107.6
-40       -70       +20       +2800     -800      -1400     +56000    +1600     +4900     +400      +158129.4 +158127.4 +158128.4
-40       -10       -20       +400      +800      +200      -8000     +1600     +100      +400      -10127.6  -10131.6  -10128.6
-40       -10       +20       +400      -800      -200      +8000     +1600     +100      +400      +21031.4  +21021.4  +21021.4
+20       -70       -20       -1400     -400      +1400     +28000    +400      +4900     +400      +97532.4  +97529.4  +97528.4
+20       -70       +20       -1400     +400      -1400     -28000    +400      +4900     +400      -16033.6  -16035.6  -16031.6
+20       -10       -20       -200      -400      +200      +4000     +400      +100      +400      +11913.4  +11910.4  +11904.4
+20       -10       +20       -200      +400      -200      -4000     +400      +100      +400      -3736.6   -3733.6   -3738.6
-61.9     -40.0     +0.0      +2476.0   -0.0      -0.0      +0.0      +3831.61  +1600.0   +0.0      +17909.474 +17908.474 +17912.474
+41.9     -40.0     +0.0      -1676.0   +0.0      -0.0      -0.0      +1755.61  +1600.0   +0.0      +10883.214 +10880.214 +10885.214
-10.0     -91.9     +0.0      +919.0    -0.0      -0.0      +0.0      +100.0    +8445.61  +0.0      +69185.041 +69186.041 +69180.041
-10.0     +11.9     +0.0      -119.0    -0.0      +0.0      -0.0      +100.0    +141.61   +0.0      +1089.241  +1092.241  +1091.241
-10.0     -40.0     -34.6     +400.0    +346.0    +1384.0   -13840.0  +100.0    +1600.0   +1197.16  -2910.848  -2905.848  -2903.848
-10.0     -40.0     +34.6     +400.0    -346.0    -1384.0   +13840.0  +100.0    +1600.0   +1197.16  +50373.232 +50375.232 +50366.232
-10.0     -40.0     +0.0      +400.0    -0.0      -0.0      +0.0      +100.0    +1600.0   +0.0      +13316.4   +13317.4   +13320.4

Рівняння регресії: y = +1.01 +0.31x1 +7.01x2 +6.85x3 +1.50x12 +0.50x13 +0.80x23 +2.00x123 +0.40x1^2 +8.10x2^2 +8.70x3^2
```

```
Перевірка значимості коефіцієнтів регресії за критерієм Стьюдента: m = 3, N = 15
Оцінки коефіцієнтів βs: 1.011, 0.31, 7.005, 6.852, 1.5, 0.5, 0.799, 2.0, 0.4, 8.1, 8.699
Коефіцієнти ts: 2.68, 0.82, 18.55, 18.15, 3.97, 1.33, 2.12, 5.30, 1.06, 21.45, 23.04
f3 = 30; q = 0.05; ttабл = 2.0423
β0 важливий; β1 неважливий; β2 важливий; β3 важливий; β12 важливий; β13 неважливий; β23 важливий; β123 важливий; β11 неважливий; β22 важливий; β33
Рівняння регресії: y = +1.01 +7.01x2 +6.85x3 +1.50x12 +0.80x23 +2.00x123 +8.10x2^2 +8.70x3^2

Перевірка адекватності моделі за критерієм Фішера: m = 3, N = 15 для таблиці y_table
Теоретичні значення y для різних комбінацій факторів:
x1 = -70        x2 = -20        x3 = 2800       : y = -63110.6222460793
x1 = -70        x2 = 20         x3 = 2800       : y = 158128.74327823825
x1 = -10        x2 = -20        x3 = 400        : y = -10128.682591810903
x1 = -10        x2 = 20         x3 = 400        : y = 21025.68293250661
x1 = -70        x2 = -20        x3 = -1400      : y = 97529.58076340175
x1 = -70        x2 = 20         x3 = -1400      : y = -16033.720378947412
x1 = -10        x2 = -20        x3 = -200       : y = 11909.520417670068
x1 = -10        x2 = 20         x3 = -200       : y = -3735.7807246789544
x1 = -40.0      x2 = 0.0        x3 = 2476.0     : y = 17909.294782848632
x1 = -40.0      x2 = 0.0        x3 = -1676.0    : y = 10883.106822584159
x1 = -91.9      x2 = 0.0        x3 = 919.0      : y = 69184.09875177428
x1 = 11.9       x2 = 0.0        x3 = -119.0     : y = 1089.8968536585783
x1 = -40.0      x2 = -34.6      x3 = 400.0      : y = -2906.7352924849456
x1 = -40.0      x2 = 34.6       x3 = 400.0      : y = 50370.83289791774
x1 = -40.0      x2 = 0.0        x3 = 400.0      : y = 13318.0710667346
Fp = 0.2908216933099792, Ft = 2.3343
```

Текст 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
import matplotlib.pyplot as plot
```

```
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 = [3.6, 8.2, 3.8, 8.7, 2.4, 0.4, 6.5, 1.8, 0.1,
9.6, 4.8]
   return regression_equation(x1, x2, x3, coeffs)

xmin = [-30, 15, -30]
xmax = [0, 50, 35]
x0 = [(xmax[_] + xmin[_])/2 for _ in range(3)]
dx = [xmax[_] - x0[_] for _ in range(3)]
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 = [[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 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 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 = [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)
    # y = []
    # x = []
    # for i in range(len(list(t_i))):
    #     x.append(i)
    #     if t_i[i] > t_our:
    #         y.append(t_i[i])
    #     else:
    #         y.append(-t_i[i])
    #
    # plot.plot(x, y)
    # plot.grid(True)
    # plot.axis([0, 11, -11, 11])
    # plot.show()
    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 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)
```

Висновок:

У ході виконання лабораторної роботи ми проведели трьохфакторний експеримент при використанні рівняння регресії з квадратичними членами. Були розроблені відповідні тестові програми, що підтверджують правильність виконання завдання. Кінцева мета роботи досягнута.