МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №6
з дисципліни «Методи оптимізації планування експерименту»

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

Виконала:
студентка групи ІО-91
Кійченко А. К.

Перевірив:
Регіда П. Г.

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 x1, x2, x3.

3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + random(10)-5,$$

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.

Завдання за варіантом:

111	10	60	-70	-10	60	70	6,7+7,1*x1+7,8*x2+7,4*x3+0,1*x1*x1+0,7*x2*x2+7,1*x3*x3+8,2*x1*x2+0,2*x1*x3+9,5*x2*x3+6,6*x1*x2*x3

Лістинг програми

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


def f(x1, x2, x3):
    coef = [6.7, 7.1, 7.8, 7.4, 8.2, 0.2, 9.5, 6.6, 0.1, 0.7, 7.1]
    return regression_equation(x1, x2, x3, coef)


def add_sq_nums(x):
    for i in range(len(x)):
        x[i][3] = x[i][0] * x[i][1]
        x[i][4] = x[i][0] * x[i][2]
        x[i][5] = x[i][1] * x[i][2]
        x[i][6] = x[i][0] * x[i][1] * x[i][2]
        x[i][7] = x[i][0] ** 2
        x[i][8] = x[i][1] ** 2
        x[i][9] = x[i][2] ** 2
    return x


def plan_matrix5(x_norm):
    l = 1.73
    x_norm = np.array(x_norm)
    x_norm = np.transpose(x_norm)
    x = np.ones(shape=(len(x_norm), len(x_norm[0])))
    for i in range(8):
        for j in range(3):
            if x_norm[i][j] == -1:
                x[i][j] = x_range[j][0]
            else:
                x[i][j] = x_range[j][1]
    for i in range(8, len(x)):
        for j in range(3):
            x[i][j] = float((x_range[j][0] + x_range[j][1]) / 2)
    dx = [x_range[i][1] - (x_range[i][0] + x_range[i][1]) / 2 for i in range(3)]
    x[8][0] = (-l * dx[0]) + x[9][0]
    x[9][0] = (l * dx[0]) + x[9][0]
    x[10][1] = (-l * dx[1]) + x[9][1]
    x[11][1] = (l * dx[1]) + x[9][1]
    x[12][2] = (-l * dx[2]) + x[9][2]
    x[13][2] = (l * dx[2]) + x[9][2]
    x = add_sq_nums(x)
    for i in range(len(x)):
```

```
        for j in range(len(x[i])):
            x[i][j] = round(x[i][j], 3)
    return x.tolist()


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(f(row[0], row[1], row[2]) + random.randint(-5, 5), 3) for _ in
range(m)] for row in factors_table]


def cochran_criteria(m, N, y_table):
    def get_cochran_value(f1, f2, q):
        partResult1 = q / f2
        params = [partResult1, f1, (f2 - 1) * f1]
        fisher = scipy.stats.f.isf(*params)
        result = fisher / (fisher + (f2 - 1))
        return Decimal(result).quantize(Decimal('.0001')).__float__()
    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 = {}".format(gp, gt))
    if gp < gt:
        print("Дисперсія однорідна")
        return True
    else:
        print("Дисперсія неоднорідна")
        return False


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 np.array(res)
    return x_i


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


def find_coefficients(factors, y_vals):
    x_i = set_factors_table(factors)
    coefficients = [[m_ij(x_i(column), x_i(row)) for column in range(11)] for row in
```

```
range(11)]
    y_numpy = list(map(lambda row: np.average(row), y_vals))
    free_values = [m_ij(y_numpy, x_i(i)) for i in range(11)]
    beta_coefficients = np.linalg.solve(coefficients, free_values)
    return list(beta_coefficients)


def print_equation(coefficients, 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(coefficients, 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 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__()
    average_variation = np.average(list(map(np.var, y_table)))
    x_i = set_factors_table(natural_plan)
    variation_beta_s = average_variation/N/m
    standard_deviation_beta_s = math.sqrt(variation_beta_s)
    t_i = np.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 = [1 if el > t_our else 0 for el in list(t_i)]
    d = sum(importance)
    # print result data
    print("βs: " + ", ".join(list(map(lambda x: str(round(float(x), 3)),
beta_coefficients))))
    print("ts: " + ", ".join(list(map(lambda i: "{:.2f}".format(i), t_i))))
    print("f3 = {}; q = {}; tтабл = {}".format(f3, q, t_our))
    print("d =", d)
    print_equation(beta_coefficients, importance)
    return importance


def regression_equation(x1, x2, x3, coef, 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(coef, factors_array), importance)])


def fisher_criteria(m, N, d, x_table, y_table, b_coefficients, importance):
    def get_fisher_value(f3, f4, q):
        return Decimal(abs(scipy.stats.f.isf(q, f4,
f3))).quantize(Decimal('.0001')).__float__()
    f3 = (m - 1) * N
    f4 = N - d
    q = 0.05
    theoretical_y = np.array([regression_equation(row[0], row[1], row[2],
b_coefficients) for row in x_table])
    # print(theoretical_y)
    average_y = np.array(list(map(lambda el: np.average(el), y_table)))
    s_ad = m/(N-d) * sum((theoretical_y - average_y)**2)
```

```
    # print(s_ad)
    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)
    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("Fp = {}, Ft = {}".format(f_p, f_t))
    print("Рівняння регресії адекватно оригіналу при рівні значимості 0.05" if f_p < f_t
else "Рівняння регресії неадекватно оригіналу при рівні значимості 0.05")
    return True if f_p < f_t else False


l = 1.73
x1min = 10
x1max = 60
x2min = -70
x2max = -10
x3min = 60
x3max = 70
x_norm = [[-1, -1, -1, -1, 1, 1, 1, 1, -1.73, 1.73, 0, 0, 0, 0],
          [-1, -1, 1, 1, -1, -1, 1, 1, 0, 0, -1.73, 1.73, 0, 0],
          [-1, 1, -1, 1, -1, 1, -1, 1, 0, 0, 0, 0, -1.73, 1.73],
          [1, 1, -1, -1, -1, -1, 1, 1, 0, 0, 0, 0, 0, 0],
          [1, -1, 1, -1, -1, 1, -1, 1, 0, 0, 0, 0, 0, 0],
          [1, -1, -1, 1, 1, -1, -1, 1, 0, 0, 0, 0, 0, 0],
          [-1, 1, 1, -1, 1, -1, -1, 1, 0, 0, 0, 0, 0, 0],
          [1, 1, 1, 1, 1, 1, 1, 1, 2.9929, 2.9929, 0, 0, 0, 0],
          [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 2.9929, 2.9929, 0, 0],
          [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 2.9929, 2.9929]]

x_range = [[x1min, x1max], [x2min, x2max], [x3min, x3max]]
x_nat = plan_matrix5(x_norm)
m = 3
N = 14
x_norm = np.transpose(np.array(x_norm))
natural_plan = generate_factors_table(x_nat)
y_arr = generate_y(m, x_nat)
print("Матриця планування:\n  X1  |  X2  |  X3  |X1*X2|X1*X3|X2*X3|X1*X2*X3|  X1\u00b2 |
X2\u00b2 |  X3\u00b2")
for i in range(14):

print(f"{x_norm[i][0]:^6}|{x_norm[i][1]:^6}|{x_norm[i][2]:^6}|{x_norm[i][3]:^5}|{x_norm[
i][4]:^5}|{x_norm[i][5]:^5}|{x_norm[i][6]:^8}|{x_norm[i][7]:^6}|{x_norm[i][8]:^6}|{x_nor
m[i][9]:^6}")

print("Матриця планування з натуралізованими значеннями факторів:"
      "\n  X1  |  X2  |  X3  | X1*X2 | X1*X3 | X2*X3 | X1*X2*X3 |   X1\u00b2  |
X2\u00b2  |  X3\u00b2")
for i in range(14):

print(f"{x_nat[i][0]:^6}|{x_nat[i][1]:^6}|{x_nat[i][2]:^6}|{x_nat[i][3]:^7}|{x_nat[i][4]
:^7}|{x_nat[i][5]:^7}|{x_nat[i][6]:^10}|{x_nat[i][7]:^8}|{x_nat[i][8]:^8}|{x_nat[i][9]:^
6}")

print("     Y1     |     Y2     |     Y3")
for i in range(14):
```

```
    print("{:^12}|{:^12}|{:^12}".format(y_arr[i][0], y_arr[i][1], y_arr[i][2]))
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)
```

Результат виконання роботи

Запишемо лінійне рівняння регресії:

$$\hat{y} = \varphi(x_0, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1x_2x_3, x_1^2, x_2^2, x_3^2, b_0, b_1, b_2, b_3, b_{12}, b_{13}, b_{23}, b_{123}, b_{11}, b_{22}, b_{33})$$

$$\hat{y} = b_0x_0 + b_1x_i + b_2x_2 + b_3x_3 + b_{12}x_1x_2 + b_{13}x_1x_3 + b_{23}x_2x_3 + b_{123}x_1x_2x_3 + b_{11}x_1^2 + b_{22}x_2^2 + b_{33}x_3^2$$

$$\varphi_i = b_0x_{0i} + b_1x_{1i} + b_2x_{2i} + b_3x_{3i} + b_{12}x_{1i}x_{2i} + b_{13}x_{1i}x_{3i} + b_{23}x_{2i}x_{3i} + b_{123}x_{1i}x_{2i}x_{3i} + b_{11}x_{1i}^2 + b_{22}x_{2i}^2 + b_{33}x_{3i}^2$$

Матриця планування:

X1	X2	X3	X1*X2	X1*X3	X2*X3	X1*X2*X3	X1²	X2²	X3²
-1.0	-1.0	-1.0	1.0	1.0	1.0	-1.0	1.0	1.0	1.0
-1.0	-1.0	1.0	1.0	-1.0	-1.0	1.0	1.0	1.0	1.0
-1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	1.0	1.0
-1.0	1.0	1.0	-1.0	-1.0	1.0	-1.0	1.0	1.0	1.0
1.0	-1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0	1.0	1.0
1.0	-1.0	1.0	-1.0	1.0	-1.0	-1.0	1.0	1.0	1.0
1.0	1.0	-1.0	1.0	-1.0	-1.0	-1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
-1.73	0.0	0.0	0.0	0.0	0.0	0.0	2.9929	0.0	0.0
1.73	0.0	0.0	0.0	0.0	0.0	0.0	2.9929	0.0	0.0
0.0	-1.73	0.0	0.0	0.0	0.0	0.0	0.0	2.9929	0.0
0.0	1.73	0.0	0.0	0.0	0.0	0.0	0.0	2.9929	0.0
0.0	0.0	-1.73	0.0	0.0	0.0	0.0	0.0	0.0	2.9929
0.0	0.0	1.73	0.0	0.0	0.0	0.0	0.0	0.0	2.9929

Матриця планування з натуралізованими значеннями факторів:

X1	X2	X3	X1*X2	X1*X3	X2*X3	X1*X2*X3	$X1^2$	$X2^2$	$X3^2$
10.0	-70.0	60.0	-700.0	600.0	-4200.0	-42000.0	100.0	4900.0	3600.0
10.0	-70.0	70.0	-700.0	700.0	-4900.0	-49000.0	100.0	4900.0	4900.0
10.0	-10.0	60.0	-100.0	600.0	-600.0	-6000.0	100.0	100.0	3600.0
10.0	-10.0	70.0	-100.0	700.0	-700.0	-7000.0	100.0	100.0	4900.0
60.0	-70.0	60.0	-4200.0	3600.0	-4200.0	-252000.0	3600.0	4900.0	3600.0
60.0	-70.0	70.0	-4200.0	4200.0	-4900.0	-294000.0	3600.0	4900.0	4900.0
60.0	-10.0	60.0	-600.0	3600.0	-600.0	-36000.0	3600.0	100.0	3600.0
60.0	-10.0	70.0	-600.0	4200.0	-700.0	-42000.0	3600.0	100.0	4900.0
-8.25	-40.0	65.0	330.0	-536.25	-2600.0	21450.0	68.062	1600.0	4225.0
78.25	-40.0	65.0	-3130.0	5086.25	-2600.0	-203450.0	6123.062	1600.0	4225.0
35.0	-91.9	65.0	-3216.5	2275.0	-5973.5	-209072.5	1225.0	8445.61	4225.0
35.0	11.9	65.0	416.5	2275.0	773.5	27072.5	1225.0	141.61	4225.0
35.0	-40.0	56.35	-1400.0	1972.25	-2254.0	-78890.0	1225.0	1600.0	3175.322
35.0	-40.0	73.65	-1400.0	2577.75	-2946.0	-103110.0	1225.0	1600.0	5424.323

Y1	Y2	Y3
-293748.3	-293743.3	-293743.3
-337275.3	-337275.3	-337271.3
-19917.3	-19919.3	-19919.3
-18145.3	-18146.3	-18147.3
-1707140.3	-1707141.3	-1707140.3
-1981567.3	-1981566.3	-1981562.3
-220711.3	-220713.3	-220713.3
-251836.3	-251833.3	-251835.3
150715.181	150708.181	150705.181
-1359656.669	-1359654.669	-1359659.669
-1426500.743	-1426497.743	-1426500.743
220948.197	220949.197	220946.197

```
 -529028.07 | -529030.07 | -529024.07

 -679234.05 | -679230.05 | -679233.05

Gp = 0.33474576271186446, Gt = 0.3517

Дисперсія однорідна

Рівняння регресії: y = -11007.29 +14.41*x1 +2.41*x2 +344.87*x3 +8.20*x12
+0.20*x13 +9.50*x23 +6.60*x123 -0.00*x1^2 +0.63*x2^2 +4.50*x3^2

βs: -11007.286, 14.412, 2.405, 344.874, 8.195, 0.198, 9.495, 6.6, -0.002, 0.629,
4.504

ts: 35597.12, 46.61, 7.78, 1115.31, 26.50, 0.64, 30.71, 21.34, 0.01, 2.03, 14.57

f3 = 28; q = 0.05; tтабл = 2.0484

d = 8

Рівняння регресії: y = -11007.29 +14.41*x1 +2.41*x2 +344.87*x3 +8.20*x12
+9.50*x23 +6.60*x123 +4.50*x3^2

Fp = 1.3739142945217766, Ft = 2.4453

Рівняння регресії адекватно оригіналу при рівні значимості 0.05
```

Висновки:

В результаті виконання роботи було досягнуто поставленої мети: проведено трьохфакторний експеримент і отримано адекватну модель –рівняння регресії, використовуючи рототабельний композиційний план.