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 № 4

з дисципліни «МНД» на тему
«Проведення трьохфакторного експерименту при використанні рівняння регресії з урахуванням ефекту взаємодії»

ВИКОНАЛА:
студентка ІІ курсу ФІОТ
групи ІВ-91
Сайко С. А.
Залікова – 9126
ПЕРЕВІРИВ:
ас. Регіда П. Г.

Київ – 2021

Мета: Провести повний трьохфакторний експеримент. Знайти рівняння регресії адекватне об'єкту.

Завдання на лабораторну роботу

1. Скласти матрицю планування для повного трьохфакторного експерименту.
2. Провести експеримент, повторивши N раз досліди у всіх точках факторного простору і знайти значення відгуку Y. Знайти значення Y шляхом моделювання випадкових чисел у певному діапазоні відповідно варіанту. Варіанти вибираються за номером в списку в журналі викладача.

$$y_{i\max} = 200 + x_{cp\max}$$
$$y_{i\min} = 200 + x_{cp\min}$$

де $x_{cp\max} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}$, $x_{cp\min} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$

3. Знайти коефіцієнти рівняння регресії і записати його.
4. Провести 3 статистичні перевірки – за критеріями Кохрена, Стьюдента, Фішера.
5. Зробити висновки по адекватності регресії та значимості окремих коефіцієнтів і записати скореговане рівняння регресії.
6. Написати комп'ютерну програму, яка усе це моделює.

Варіант: 119

№ варіанта	X1		X2		X3	
	min	max	min	max	min	max
124	-30	0	-25	10	-25	-5

Лістинг програми:

```
import math
import numpy as np
from numpy import average, transpose
from numpy.linalg import solve
from prettytable import PrettyTable
from scipy.stats import f
from scipy.stats import t as t_criterium
from functools import partial
from random import randint

m, N, d = 3, 8, 8
gener_x = []
x1, x2, x3 = (-30, 0), (-25, 10), (-25, -5)
x_name = [x1, x2, x3]
for i in range(N):
    gener_x.append([])
    gener_x[i].append(randint(x1[0], x1[1]))
    gener_x[i].append(randint(x2[0], x2[1]))
    gener_x[i].append(randint(x3[0], x3[1]))

plan_matrix = []
for str_ind in range(N):
```

```
    plan_matrix.append([])
    plan_matrix[str_ind].append(1)
    for xi in gener_x[str_ind]:
        index = gener_x[str_ind].index(xi)
        x = x_name[index]
        if average([x[0], x[1]]) < xi:
            plan_matrix[str_ind].append(1)
        else:
            plan_matrix[str_ind].append(-1)
c = np.array(plan_matrix)
Tplan_matrix = c.T
# x0_factor = Tplan_matrix[0]
# x1_factor = Tplan_matrix[1]
# x2_factor = Tplan_matrix[2]
# x3_factor = Tplan_matrix[3]

x0_factor = [1, 1, 1, 1, 1, 1, 1, 1]
x1_factor = [-1, -1, 1, 1, -1, -1, 1, 1]
x2_factor = [-1, 1, -1, 1, -1, 1, -1, 1]
x3_factor = [-1, 1, 1, -1, 1, -1, -1, 1]

x1x2_factor = [a * b for a, b in zip(x1_factor, x2_factor)]
x1x3_factor = [a * b for a, b in zip(x1_factor, x3_factor)]
x2x3_factor = [a * b for a, b in zip(x2_factor, x3_factor)]
x1x2x3_factor = [a * b * c for a, b, c in zip(x1_factor, x2_factor, x3_factor)]

x1_list = []
x2_list = []
x3_list = []
x1x2_list = []
x1x3_list = []
x2x3_list = []
x1x2x3_list = []
x_main_list = [x0_factor, x1_list, x2_list, x3_list, x1x2_list, x1x3_list,
x2x3_list, x1x2x3_list]
x_factor_list = [x0_factor, x1_factor, x2_factor, x3_factor, x1x2_factor,
x1x3_factor, x2x3_factor, x1x2x3_factor]

list_bi = []

F1 = m - 1
F2 = N
F3 = F1 * F2
F4 = N - d

x1, x2, x3 = (-30, 0), (-25, 10), (-25, -5)
x_tuple = (x1, x2, x3)
x_max_average = average([i[1] for i in x_tuple])
x_min_average = average([i[0] for i in x_tuple])
y_max = int(200 + x_max_average)
y_min = int(200 + x_min_average)
y_min_max = [y_min, y_max]
mat_Y = [[randint(y_min_max[0], y_min_max[1]) for _ in range(m)] for _ in range(N)]

def get_average_y():
    return [round(sum(mat_Y[k1]) / m, 3) for k1 in range(N)]

def get_dispersion():
    return [round(sum([((k1 - get_average_y()[j]) ** 2) for k1 in mat_Y[j]]) / m, 3)
for j in
            range(N)]

def fill_x_matrix():
    [x1_list.append(x1[0 if i == -1 else 1]) for i in x1_factor]
```

```
    [x2_list.append(x2[0 if i == -1 else 1]) for i in x2_factor]
    [x3_list.append(x3[0 if i == -1 else 1]) for i in x3_factor]
    [x1x2_list.append(a * b) for a, b in zip(x1_list, x2_list)]
    [x1x3_list.append(a * b) for a, b in zip(x1_list, x3_list)]
    [x2x3_list.append(a * b) for a, b in zip(x2_list, x3_list)]
    [x1x2x3_list.append(a * b * c) for a, b, c in zip(x1_list, x2_list, x3_list)]

def cohren():
    q = 0.05
    Gp = max(get_dispersion()) / sum(get_dispersion())
    q1 = q / F1
    fisher_value = f.ppf(q=1 - q1, dfn=F2, dfd=(F1 - 1) * F2)
    Gt = fisher_value / (fisher_value + F1 - 1)
    return Gp < Gt

def fisher():
    fisher_teor = partial(f.ppf, q=0.05)
    Ft = fisher_teor(dfn=F4, dfd=F3)
    return Ft


fill_x_matrix()
dispersion = get_dispersion()
sum_dispersion = sum(dispersion)
y_average = get_average_y()

column_names1 = ["X0", "X1", "X2", "X3", "X1X2", "X1X3", "X2X3", "X1X2X3", "Y1",
"Y2", "Y3", "Y", "S^2"]
trans_y_mat = transpose(mat_Y).tolist()

list_for_solve_a = list(zip(*x_main_list))
list_for_solve_b = x_factor_list

for k in range(N):
    S = 0
    for i in range(N):
        S += (list_for_solve_b[k][i] * y_average[i]) / N
    list_bi.append(round(S, 5))

pt = PrettyTable()
cols = x_factor_list
[cols.extend(ls) for ls in [trans_y_mat, [y_average], [dispersion]]]
[pt.add_column(column_names1[coll_id], cols[coll_id]) for coll_id in range(13)]
print("Матриця планування")
print(pt, "\n")


pt = PrettyTable()
cols = x_main_list
[cols.extend(ls) for ls in [trans_y_mat, [y_average], [dispersion]]]
[pt.add_column(column_names1[coll_id], cols[coll_id]) for coll_id in range(13)]
print("Нормована матриця")
print(pt, "\n")
list_ai = [round(i, 5) for i in solve(list_for_solve_a, y_average)]
print("Критерій Кохрена")
if cohren():
    print("Дисперсія однорідна!\n")
    Dispersion_B = sum_dispersion / N
    Dispersion_beta = Dispersion_B / (m * N)
    S_beta = math.sqrt(abs(Dispersion_beta))
    beta_list = np.zeros(8).tolist()
    for i in range(N):
        beta_list[0] += (y_average[i] * x0_factor[i]) / N
        beta_list[1] += (y_average[i] * x1_factor[i]) / N
```

```
        beta_list[2] += (y_average[i] * x2_factor[i]) / N
        beta_list[3] += (y_average[i] * x3_factor[i]) / N
        beta_list[4] += (y_average[i] * x1x2_factor[i]) / N
        beta_list[5] += (y_average[i] * x1x3_factor[i]) / N
        beta_list[6] += (y_average[i] * x2x3_factor[i]) / N
        beta_list[7] += (y_average[i] * x1x2x3_factor[i]) / N
    t_list = [abs(beta_list[i]) / S_beta for i in range(0, N)]
    print("Критерій Стьюдента")
    for i, j in enumerate(t_list):
        print(f't{i}={beta_list[i]}')
        if j < t_criterium.ppf(q=0.975, df=F3):
            beta_list[i] = 0
            d -= 1
    print()
    print('Рівняння регресії з коефіцієнтами від нормованих значень факторів')
    print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1x2 + {}*x1x3 + {}*x2x3 +
{}*x1x2x3".format(*list_bi))
    print('Рівняння регресії з коефіцієнтами від натуральних значень факторів')
    print("y = {} + {}*x1 + {}*x2 + {}*x3 + {}*x1x2 + {}*x1x3 + {}*x2x3 +
{}*x1x2x3".format(*list_ai))
    print()
    Y_counted = [sum([beta_list[0], *[beta_list[i] * x_main_list[1:][j][i] for i in
range(N)]])
                 for j in range(N)]
    Dispersion_ad = 0
    for i in range(len(Y_counted)):
        Dispersion_ad += (( y_average[i] -Y_counted[i]) ** 2) * m / (N - d)
    Fp = Dispersion_ad / Dispersion_beta
    Ft = fisher()
    print("Критерій Фішера")
    if 4.5 > Fp:
        print("Рівняння регресії адекватне!")
    else:
        print("Рівняння регресії неадекватне.")
else:
    print("Дисперсія неоднорідна")
```

Результат виконання роботи:

```
Матриця планування
+----+----+----+----+------+------+------+--------+-----+-----+-----+---------+--------+
| X0 | X1 | X2 | X3 | X1X2 | X1X3 | X2X3 | X1X2X3 |  Y1 |  Y2 |  Y3 |    Y    |  S^2   |
+----+----+----+----+------+------+------+--------+-----+-----+-----+---------+--------+
| 1  | -1 | -1 | -1 |  1   |  1   |  1   |   -1   | 189 | 187 | 200 |  192.0  | 32.667 |
| 1  | -1 | 1  | 1  |  -1  |  -1  |  1   |   -1   | 177 | 191 | 183 | 183.667 | 32.889 |
| 1  | 1  | -1 | 1  |  -1  |  1   |  -1  |   -1   | 197 | 195 | 200 | 197.333 | 4.222  |
| 1  | 1  | 1  | -1 |  1   |  -1  |  -1  |   -1   | 183 | 182 | 185 | 183.333 | 1.556  |
| 1  | -1 | -1 | 1  |  1   |  -1  |  -1  |   1    | 200 | 177 | 192 | 189.667 | 90.889 |
| 1  | -1 | 1  | -1 |  -1  |  1   |  -1  |   1    | 175 | 185 | 174 |  178.0  | 24.667 |
| 1  | 1  | -1 | -1 |  -1  |  -1  |  1   |   1    | 183 | 178 | 173 |  178.0  | 16.667 |
| 1  | 1  | 1  | 1  |  1   |  1   |  1   |   1    | 190 | 177 | 179 |  182.0  | 32.667 |
+----+----+----+----+------+------+------+--------+-----+-----+-----+---------+--------+

Нормована матриця
+----+-----+-----+-----+------+------+------+--------+-----+-----+-----+---------+--------+
| X0 |  X1 |  X2 |  X3 | X1X2 | X1X3 | X2X3 | X1X2X3 |  Y1 |  Y2 |  Y3 |    Y    |  S^2   |
+----+-----+-----+-----+------+------+------+--------+-----+-----+-----+---------+--------+
| 1  | -30 | -25 | -25 | 750  | 750  | 625  | -18750 | 189 | 187 | 200 |  192.0  | 32.667 |
| 1  | -30 |  10 |  -5 | -300 | 150  | -50  |  1500  | 177 | 191 | 183 | 183.667 | 32.889 |
| 1  |  0  | -25 |  -5 |  0   |  0   | 125  |   0    | 197 | 195 | 200 | 197.333 | 4.222  |
| 1  |  0  |  10 | -25 |  0   |  0   | -250 |   0    | 183 | 182 | 185 | 183.333 | 1.556  |
| 1  | -30 | -25 |  -5 | 750  | 150  | 125  | -3750  | 200 | 177 | 192 | 189.667 | 90.889 |
| 1  | -30 |  10 | -25 | -300 | 750  | -250 |  7500  | 175 | 185 | 174 |  178.0  | 24.667 |
| 1  |  0  | -25 | -25 |  0   |  0   | 625  |   0    | 183 | 178 | 173 |  178.0  | 16.667 |
| 1  |  0  |  10 |  -5 |  0   |  0   | -50  |   0    | 190 | 177 | 179 |  182.0  | 32.667 |
+----+-----+-----+-----+------+------+------+--------+-----+-----+-----+---------+--------+
Критерій Кохрена
Дисперсія однорідна!

Критерій Стьюдента
t0=186.083375
t1=2.583375
t2=-1.3333750000000002
t3=-0.0833750000000002
t4=-0.8333750000000002
t5=-0.5833750000000002
t6=2.166875000000001
t7=-1.0001250000000148

Рівняння регресії з коефіцієнтами від нормованих значень факторів
y = 186.08337 + 2.58338*x1 + -1.33338*x2 + -0.08338*x3 + -0.83338*x1x2 + -0.58338*x1x3 + 2.16688*x2x3 + -1.00013*x1x2x3
Рівняння регресії з коефіцієнтами від натуральних значень факторів
y = 187.48807 + 0.04721*x1 + -0.02381*x2 + -0.01667*x3 + -0.00889*x1x2 + -0.00675*x1x3 + 0.00667*x2x3 + -0.00038*x1x2x3

Критерій Фішера
Рівняння регресії неадекватне.
```

Висновок:

Під час виконання лабораторної роботи було змодельовано трьохфакторний експеримент при використанні лінійного рівняння регресії та рівняння регресії з ефектом взаємодії, складено матрицю планування експерименту, було визначено коефіцієнти рівняння регресії(натуралізовані та

нормовані), виконано перевірку правильності розрахунку коефіцієнтів рівняння регресії. Також було проведено 3 статистичні перевірки(використання критеріїв Кохрена, Стьюдента та Фішера). При виявленні неадекватності лінійного рівняння регресії оригіналу було застосовано ефект взаємодії факторів. Довірча ймовірність в даній роботі дорівнює 0.95, відповідно рівень значимості q = 0.05.