

Міністерство освіти і науки України  
Національний технічний університет України  
«Київський політехнічний інститут ім. Ігоря Сікорського»  
Факультет інформатики та обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 6  
з дисципліни «Методи наукових досліджень»  
на тему «Проведення трьохфакторного експерименту  
при використанні рівняння регресії квадратичними членами.»

ВИКОНАВ:  
студент 2 курсу  
групи ІВ-91  
Щоткін М.А.  
Залікова – 9131

ПЕРЕВІРИВ:  
ас. Регіда П. Г.

**Мета:** Провести повний 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  $x_1, x_2, x_3$ . Обчислити і записати значення, відповідні кодованим значенням факторів +1; -1; +1; -1; 0 для  $x_1, x_2, x_3$ .
3. Значення функції відгуку знайти за допомогою підстановки в формулу:

$$y_i = f(x_1, x_2, x_3) + \text{random}(10) - 5,$$

де  $f(x_1, x_2, x_3)$  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.

№ варіанту	X <sub>1</sub>		X <sub>2</sub>		X <sub>3</sub>	
	min	max	min	max	min	max
129	-25	75	-20	40	-20	-15

## Програмний код

```
from math import fabs
from random import randrange
import numpy as np
from numpy.linalg import solve
from scipy.stats import f, t
from prettytable import PrettyTable

x1min = -25
x1max = 75
x2min = -20
x2max = 40
x3min = -20
x3max = -15

x01 = (x1max + x1min) / 2
x02 = (x2max + x2min) / 2
x03 = (x3max + x3min) / 2
deltax1 = x1max - x01
deltax2 = x2max - x02
deltax3 = x3max - x03

def function(X1, X2, X3):
    y = 1.4 + 4.0 * X1 + 9.3 * X2 + 7.9 * X3 + 7.7 * X1 * X1 + 1.0 * X2 * X2 + 0.7 * X3 * X3 + 2.9 * X1 * X2 + \
        0.4 * X1 * X3 + 3.9 * X2 * X3 + 1.0 * X1 * X2 * X3
    return y

def main(n,m):
    global coefs_count

    xn = [[-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, -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],
[+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, +1, +1, +1, +1, +1],
[-1.73, 0, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],
[+1.73, 0, 0, 0, 0, 0, 0, 0, 2.9929, 0, 0],
[0, -1.73, 0, 0, 0, 0, 0, 0, 0, 2.9929, 0],
[0, +1.73, 0, 0, 0, 0, 0, 0, 0, 2.9929, 0],
[0, 0, -1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],
[0, 0, +1.73, 0, 0, 0, 0, 0, 0, 2.9929, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

x1 = [x1min, x1min, x1min, x1min, x1max, x1max, x1max, x1max, -1.73 *
deltax1 + x01, 1.73 * deltax1 + x01, x01, x01,
      x01, x01, x01]
x2 = [x2min, x2min, x2max, x2max, x2min, x2min, x2max, x2max, x02, x02, -
1.73 * deltax2 + x02, 1.73 * deltax2 + x02,
      x02, x02, x02]
x3 = [x3min, x3max, x3min, x3max, x3min, x3max, x3min, x3max, x03, x03,
x03, x03, -1.73 * deltax3 + x03,
      1.73 * deltax3 + x03, x03]
x1x2 = [0] * 15
x1x3 = [0] * 15
x2x3 = [0] * 15
x1x2x3 = [0] * 15
x1kv = [0] * 15
x2kv = [0] * 15
x3kv = [0] * 15
for i in range(15):
    x1x2[i] = x1[i] * x2[i]
    x1x3[i] = x1[i] * x3[i]
    x2x3[i] = x2[i] * x3[i]
    x1x2x3[i] = x1[i] * x2[i] * x3[i]
    x1kv[i] = x1[i] ** 2
    x2kv[i] = x2[i] ** 2
    x3kv[i] = x3[i] ** 2

list_for_a = list(zip(x1, x2, x3, x1x2, x1x3, x2x3, x1x2x3, x1kv, x2kv,
x3kv))

for i in range(len(list_for_a)):
    list_for_a[i] = list(list_for_a[i])
    for j in range(len(list_for_a[i])):
        list_for_a[i][j] = round(list_for_a[i][j], 3)

planning_matrix_x = PrettyTable()
planning_matrix_x.field_names = ['X1', 'X2', 'X3', 'X1X2', 'X1X3',
'X2X3', 'X1X2X3', 'X1X1', 'X2X2', 'X3X3']
print("Матриця планування з натуралізованими коефіцієнтами X:")
planning_matrix_x.add_rows(list_for_a)
print(planning_matrix_x)

Y = [[function(list_for_a[j][0], list_for_a[j][1], list_for_a[j][2]) for
i in range(m)] for j in range(15)]

planing_matrix_y = PrettyTable()
planing_matrix_y.field_names = ['Y1', 'Y2', 'Y3']
print("Матриця планування Y:")
planing_matrix_y.add_rows(Y)
print(planing_matrix_y)

Y_average = []

```

```

for i in range(len(Y)):
    Y_average.append(np.mean(Y[i], axis=0))
print("Середні значення відгуку за рядками:")
for i in range(15):
    print("{:.3f}".format(Y_average[i]), end=" ")

dispersions = []
for i in range(len(Y)):
    a = 0
    for k in Y[i]:
        a += (k - np.mean(Y[i], axis=0)) ** 2
    dispersions.append(a / len(Y[i]))

def find_known(num):
    a = 0
    for j in range(15):
        a += Y_average[j] * list_for_a[j][num - 1] / 15
    return a

def a(first, second):
    a = 0
    for j in range(15):
        a += list_for_a[j][first - 1] * list_for_a[j][second - 1] / 15
    return a

my = sum(Y_average) / 15
mx = []
for i in range(10):
    number_lst = []
    for j in range(15):
        number_lst.append(list_for_a[j][i])
    mx.append(sum(number_lst) / len(number_lst))

det1 = [
    [1, mx[0], mx[1], mx[2], mx[3], mx[4], mx[5], mx[6], mx[7], mx[8],
mx[9]],
    [mx[0], a(1, 1), a(1, 2), a(1, 3), a(1, 4), a(1, 5), a(1, 6), a(1,
7), a(1, 8), a(1, 9), a(1, 10)],
    [mx[1], a(2, 1), a(2, 2), a(2, 3), a(2, 4), a(2, 5), a(2, 6), a(2,
7), a(2, 8), a(2, 9), a(2, 10)],
    [mx[2], a(3, 1), a(3, 2), a(3, 3), a(3, 4), a(3, 5), a(3, 6), a(3,
7), a(3, 8), a(3, 9), a(3, 10)],
    [mx[3], a(4, 1), a(4, 2), a(4, 3), a(4, 4), a(4, 5), a(4, 6), a(4,
7), a(4, 8), a(4, 9), a(4, 10)],
    [mx[4], a(5, 1), a(5, 2), a(5, 3), a(5, 4), a(5, 5), a(5, 6), a(5,
7), a(5, 8), a(5, 9), a(5, 10)],
    [mx[5], a(6, 1), a(6, 2), a(6, 3), a(6, 4), a(6, 5), a(6, 6), a(6,
7), a(6, 8), a(6, 9), a(6, 10)],
    [mx[6], a(7, 1), a(7, 2), a(7, 3), a(7, 4), a(7, 5), a(7, 6), a(7,
7), a(7, 8), a(7, 9), a(7, 10)],
    [mx[7], a(8, 1), a(8, 2), a(8, 3), a(8, 4), a(8, 5), a(8, 6), a(8,
7), a(8, 8), a(8, 9), a(8, 10)],
    [mx[8], a(9, 1), a(9, 2), a(9, 3), a(9, 4), a(9, 5), a(9, 6), a(9,
7), a(9, 8), a(9, 9), a(9, 10)],
    [mx[9], a(10, 1), a(10, 2), a(10, 3), a(10, 4), a(10, 5), a(10, 6),
a(10, 7), a(10, 8), a(10, 9), a(10, 10)]]

det2 = [my, find_known(1), find_known(2), find_known(3), find_known(4),
find_known(5), find_known(6), find_known(7),
        find_known(8), find_known(9), find_known(10)]

```

```

beta = solve(det1, det2)
print("\nОтримане рівняння регресії:")
print("{:.3f} + {:.3f} * X1 + {:.3f} * X2 + {:.3f} * X3 + {:.3f} * X1X2 + {:.3f} * X1X3 + {:.3f} * X2X3"
      + {:.3f} * X1X2X3 + {:.3f} * X11^2 + {:.3f} * X22^2 + {:.3f} * X33^2 = ŷ"
      .format(*beta))
y_i = [0] * 15
print("Експериментальні значення:")
for k in range(15):
    y_i[k] = beta[0] + beta[1] * list_for_a[k][0] + beta[2] * list_for_a[k][1] + beta[3] * list_for_a[k][2] + \
    beta[4] * list_for_a[k][3] + beta[5] * list_for_a[k][4] + beta[6] * list_for_a[k][5] + beta[7] * \
    list_for_a[k][6] + beta[8] * list_for_a[k][7] + beta[9] * list_for_a[k][8] + beta[10] * list_for_a[k][9]
    for i in range(15):
        print("{:.3f}".format(y_i[i]), end=" ")
    print("\n\n----- Перевірка за критерієм Кохрена -----")

Gp = max(dispersions) / sum(dispersions)
Gt = 0.3346
print("Gp =", Gp)
if Gp < Gt:
    print("Дисперсія однорідна")
else:
    print("Дисперсія неоднорідна")

print("\n----- Перевірка значущості коефіцієнтів за критерієм Стьюдента -----")
sb = sum(dispersions) / len(dispersions)
sbs = (sb / (15 * m)) ** 0.5

F3 = (m - 1) * n
coefs1 = []
coefs2 = []
d = 11
res = [0] * 11
for j in range(11):
    t_pract = 0
    for i in range(15):
        if j == 0:
            t_pract += Y_average[i] / 15
        else:
            t_pract += Y_average[i] * xn[i][j - 1]
        res[j] = beta[j]
    if fabs(t_pract / sbs) < t.ppf(q=0.975, df=F3):
        coefs2.append(beta[j])
        res[j] = 0
        d -= 1
    else:
        coefs1.append(beta[j])
coefs_count += len(coefs1)
print("Значущі коефіцієнти регресії:", [round(i, 3) for i in coefs1])
print("Незначущі коефіцієнти регресії:", [round(i, 3) for i in coefs2])
y_st = []
for i in range(15):
    y_st.append(res[0] + res[1] * x1[i] + res[2] * x2[i] + res[3] * x3[i] + res[4] * x1x2[i] + res[5] * x1x3[i] + res[6] * x2x3[i] + res[7] * x1x2x3[i] + res[8] * x1kv[i] + res[9] * x2kv[i] + res[10] * x3kv[i])
print("Значення з отриманими коефіцієнтами:")
for i in range(15):

```

```

        print("{:.3f}".format(y_st[i]), end=" ")

    print("\n\n----- Перевірка адекватності за критерієм
Фішера -----")
    Sad = m * sum([(y_st[i] - Y_average[i]) ** 2 for i in range(15)]) / (n -
d)
    Fp = Sad / sb
    F4 = n - d
    print("Fp =", Fp)
    if Fp < f.ppf(q=0.95, dfn=F4, dfd=F3):
        print("Рівняння регресії адекватне при рівні значимості 0.05")
    else:
        print("Рівняння регресії неадекватне при рівні значимості 0.05")

coefs_count = 0
iterations_count = 0
while coefs_count < 50:
    main(15,3)
    if coefs_count < 50:
        iterations_count += 1

print("Кількість значимих коефіцієнтів: ", coefs_count)
print("Кількість ітерацій за які досягли 50+ значимих коефіцієнтів: ",
iterations_count)

```

## Висновок:

Під час виконання даної лабораторної роботи я провів трьохфакторний експеримент під час якого потрібно отримати адекватну модель – рівняння регресії, використовуючи рототабельний композиційний план, перевірів однорідність дисперсії за критерієм Кохрена, отримав коефіцієнти рівняння регресії, оцінив значимість знайдених коефіцієнтів за критеріями Стюдента та Фішера.

Мета лабораторної роботи була досягнута.