

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

**Методи планування експерименту**

**Лабораторна робота №6**

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

**Виконав:**

студент II курсу ФІОТ

групи ІВ-92

Накарловіч Р. Р.

номер у списку групи – 14

**Перевірив:**

ас. Регіда П. Г.

## Мета:

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

## Завдання:

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  $+1; -1; +l; -l; 0$  для  $\bar{x}_1, \bar{x}_2, \bar{x}_3$ .
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.

Варіанти обираються по номеру в списку в журналі викладача.

## Варіант завдання:

№ варіанту	$x_1$		$x_2$		$x_3$		$f(x_1, x_2, x_3)$
	min	max	min	max	min	max	
214	-25	75	-20	60	-25	-10	$9,1+2,2*x_1+5,9*x_2+7,9*x_3+5,6*x_1*x_1+1,0*x_2*x_2+3,0*x_3*x_3+2,2*x_1*x_2+0,8*x_1*x_3+8,8*x_2*x_3+3,6*x_1*x_2*x_3$

## Лістинг програми:

```
import math
import random

import numpy
from numpy.linalg import solve
from scipy.stats import f, t
from prettytable import PrettyTable

class LaboratoryWorkN6:
    def __init__(self, n, m):
        self.n, self.m = n, m
        self.min_x1 = -25
        self.max_x1 = 75
        self.min_x2 = -20
        self.max_x2 = 60
        self.min_x3 = -25
        self.max_x3 = -10

        self.x01 = (self.min_x1 + self.max_x1) / 2
        self.x02 = (self.min_x2 + self.max_x2) / 2
        self.x03 = (self.min_x3 + self.max_x3) / 2
        self.delta_x1 = self.max_x1 - self.x01
        self.delta_x2 = self.max_x2 - self.x02
        self.delta_x3 = self.max_x3 - self.x03

        self.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, 2.9929, 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, +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, +1.73, 0, 0, 0, 0, 0, 0, 2.9929],
                    [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

        self.x1 = [self.min_x1, self.min_x1, self.min_x1, self.min_x1,
                    self.max_x1, self.max_x1, self.max_x1,
                    self.max_x1, -1.73 * self.delta_x1 + self.x01, 1.73 *
                    self.delta_x1 + self.x01, self.x01, self.x01,
                    self.x01, self.x01, self.x01]
        self.x2 = [self.min_x2, self.min_x2, self.min_x2, self.min_x2,
                    self.max_x2, self.max_x2, self.max_x2,
                    self.max_x2, self.x02, self.x02, -1.73 * self.delta_x2 +
                    self.x02, 1.73 * self.delta_x2 + self.x02,
                    self.x02, self.x02, self.x02]
        self.x3 = [self.min_x3, self.max_x3, self.min_x3, self.max_x3,
                    self.min_x3, self.max_x3, self.min_x3,
                    self.max_x3, self.x03, self.x03, self.x03, self.x03,
                    self.x03, self.x03, -1.73 *
                    self.delta_x3 + self.x03,
                    1.73 * self.delta_x3 + self.x03, self.x03]

        self.x1x2 = [0] * 15
        self.x1x3 = [0] * 15
        self.x2x3 = [0] * 15
        self.x1x2x3 = [0] * 15
        self.x1kv = [0] * 15
        self.x2kv = [0] * 15
```

```

self.x3kv = [0] * 15

for i in range(15):
    self.x1x2[i] = self.x1[i] * self.x2[i]
    self.x1x3[i] = self.x1[i] * self.x3[i]
    self.x2x3[i] = self.x2[i] * self.x3[i]
    self.x1x2x3[i] = self.x1[i] * self.x2[i] * self.x3[i]
    self.x1kv[i] = self.x1[i] ** 2
    self.x2kv[i] = self.x2[i] ** 2
    self.x3kv[i] = self.x3[i] ** 2

    @staticmethod
    def function(x1, x2, x3):
        return 9.1 + 2.2 * x1 + 5.9 * x2 + 7.9 * x3 + 5.6 * x1 * x1 + 1 * x2 *
x2 + 3 * x3 * x3 + 2.2 * x1 * x2 + \
        0.8 * x1 * x3 + 8.8 * x2 * x3 + 3.6 * x1 * x2 * x3 +
random.randint(0, 10) - 5

    def run(self):
        def find_known(number):
            result = 0
            for j in range(15):
                result += average_y[j] * list_a[j][number - 1] / 15
            return result

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

        # Округлення до третього знаку після коми
        list_a = [list(map(lambda p: round(p, 3), nested)) for nested in
list(zip(self.x1, self.x2, self.x3, self.x1x2,
self.x1x3, self.x2x3, self.x1x2x3,
self.x1kv, self.x2kv, self.x3kv))]
        planning_matrix_x = PrettyTable()
        planning_matrix_x.title = 'Матриця планування з натуралізованими
коефіцієнтами X'
        planning_matrix_x.field_names = ['X1', 'X2', 'X3', 'X1X2', 'X1X3',
'X2X3', 'X1X2X3', 'X1X1', 'X2X2', 'X3X3']
        planning_matrix_x.add_rows(list_a)
        print(planning_matrix_x)

        # Округлення до третього знаку після коми
        y = [list(map(lambda p: round(p, 3), nested))
for nested in [[LaboratoryWorkN6.function(list_a[j][0],
list_a[j][1], list_a[j][2])
for _ in range(self.m)] for j in range(15)]]
        planning_matrix_y = PrettyTable()
        planning_matrix_y.title = 'Матриця планування Y'
        planning_matrix_y.field_names = ['Y1', 'Y2', 'Y3']
        planning_matrix_y.add_rows(y)
        print(planning_matrix_y)

        average_y = []
        for i in range(len(y)):
            average_y.append(numpy.mean(y[i], axis=0))
        print('Середні значення відгуку за рядками:')
        for i in range(15):
            print('\t{:.3f}'.format(average_y[i]), end=' ')

        dispersions = []
        for i in range(len(y)):

```

```

a = 0
for k in y[i]:
    a += (k - numpy.mean(y[i], axis=0)) ** 2
dispersions.append(a / len(y[i]))
my = sum(average_y) / 15
mx = []
for i in range(10):
    number_lst = []
    for j in range(15):
        number_lst.append(list_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], g(1, 1), g(1, 2), g(1, 3), g(1, 4), g(1, 5), g(1, 6), g(1,
7), g(1, 8), g(1, 9), g(1, 10)],
    [mx[1], g(2, 1), g(2, 2), g(2, 3), g(2, 4), g(2, 5), g(2, 6), g(2,
7), g(2, 8), g(2, 9), g(2, 10)],
    [mx[2], g(3, 1), g(3, 2), g(3, 3), g(3, 4), g(3, 5), g(3, 6), g(3,
7), g(3, 8), g(3, 9), g(3, 10)],
    [mx[3], g(4, 1), g(4, 2), g(4, 3), g(4, 4), g(4, 5), g(4, 6), g(4,
7), g(4, 8), g(4, 9), g(4, 10)],
    [mx[4], g(5, 1), g(5, 2), g(5, 3), g(5, 4), g(5, 5), g(5, 6), g(5,
7), g(5, 8), g(5, 9), g(5, 10)],
    [mx[5], g(6, 1), g(6, 2), g(6, 3), g(6, 4), g(6, 5), g(6, 6), g(6,
7), g(6, 8), g(6, 9), g(6, 10)],
    [mx[6], g(7, 1), g(7, 2), g(7, 3), g(7, 4), g(7, 5), g(7, 6), g(7,
7), g(7, 8), g(7, 9), g(7, 10)],
    [mx[7], g(8, 1), g(8, 2), g(8, 3), g(8, 4), g(8, 5), g(8, 6), g(8,
7), g(8, 8), g(8, 9), g(8, 10)],
    [mx[8], g(9, 1), g(9, 2), g(9, 3), g(9, 4), g(9, 5), g(9, 6), g(9,
7), g(9, 8), g(9, 9), g(9, 10)],
    [mx[9], g(10, 1), g(10, 2), g(10, 3), g(10, 4), g(10, 5), g(10, 6),
g(10, 7), g(10, 8), g(10, 9),
    g(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('\t{:.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[0], beta[1], beta[2], beta[3], beta[4], beta[5],
beta[6], beta[7], beta[8], beta[9],
    beta[10]))
y_i = [0] * 15
print('Експериментальні значення:')
for k in range(15):
    y_i[k] = beta[0] + beta[1] * list_a[k][0] + beta[2] * list_a[k][1] +
beta[3] * list_a[k][2] + \
        beta[4] * list_a[k][3] + beta[5] * list_a[k][4] + beta[6] *
list_a[k][5] + beta[7] * \
        list_a[k][6] + beta[8] * list_a[k][7] + beta[9] *
list_a[k][8] + beta[10] * \
        list_a[k][9]
    for i in range(15):
        print('\t{:.3f}'.format(y_i[i]), end=' ')
print('\nПеревірка за критерієм Кохрена:')
gp = max(dispersions) / sum(dispersions)
gt = 0.3346

```

```

print(f'\tGp = {gp}')
if gp < gt:
    print('\tДисперсія однорідна.')
else:
    print('\tДисперсія неоднорідна.')

print('Перевірка значущості коефіцієнтів за критерієм Стьюдента:')
sb = sum(dispersions) / len(dispersions)
sbs = (sb / (15 * self.m)) ** 0.5
f3 = (self.m - 1) * self.n
coefficients1 = []
coefficients2 = []
d = 11
res = [0] * 11
for j in range(11):
    t_practical = 0
    for i in range(15):
        if j == 0:
            t_practical += average_y[i] / 15
        else:
            t_practical += average_y[i] * self.xn[i][j - 1]
        res[j] = beta[j]
    if math.fabs(t_practical / sbs) < t.ppf(q=0.975, df=f3):
        coefficients2.append(beta[j])
        res[j] = 0
        d -= 1
    else:
        coefficients1.append(beta[j])
    print('\tЗначущі коефіцієнти регресії:', [round(i, 3) for i in
coefficients1])
    print('\tНезначущі коефіцієнти регресії:', [round(i, 3) for i in
coefficients2])

y_st = []
for i in range(15):
    y_st.append(res[0] + res[1] * self.x1[i] + res[2] * self.x2[i] +
res[3] * self.x3[i] + res[4]
                * self.x1x2[i] + res[5] * self.x1x3[i] + res[6] *
self.x2x3[i] + res[7] * self.x1x2x3[i]
                + res[8] * self.x1kv[i] + res[9] * self.x2kv[i] +
res[10] * self.x3kv[i])
print('Значення з отриманими коефіцієнтами:')
for i in range(15):
    print('\t{:.3f}'.format(y_st[i]), end=' ')

print('\nПеревірка адекватності за критерієм Фішера:')
sad = self.m * sum([(y_st[i] - average_y[i]) ** 2 for i in range(15)]) /
(self.n - d)
fp = sad / sb
f4 = self.n - d
print(f'\tFp = {fp}')
if fp < f.ppf(q=0.95, dfn=f4, dfd=f3):
    print('\tРівняння регресії адекватне.')
else:
    print('\tРівняння регресії не є адекватним.')

if __name__ == '__main__':
    worker = LaboratoryWorkN6(15, 3)
    worker.run()

```

Результати виконання роботи:

Матриця планування з натуралізованими коефіцієнтами X										
X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	X1X1	X2X2	X3X3	
-25	-20	-25	500	625	500	-12500	625	400	625	
-25	-20	-10	500	250	200	-5000	625	400	100	
-25	60	-25	-1500	625	-1500	37500	625	3600	625	
-25	60	-10	-1500	250	-600	15000	625	3600	100	
75	-20	-25	-1500	-1875	500	37500	5625	400	625	
75	-20	-10	-1500	-750	200	15000	5625	400	100	
75	60	-25	4500	-1875	-1500	-112500	5625	3600	625	
75	60	-10	4500	-750	-600	-45000	5625	3600	100	
-61.5	20.0	-17.5	-1230.0	1076.25	-350.0	21525.0	3782.25	400.0	306.25	
111.5	20.0	-17.5	2230.0	-1951.25	-350.0	-39025.0	12432.25	400.0	306.25	
25.0	-49.2	-17.5	-1230.0	-437.5	861.0	21525.0	625.0	2420.64	306.25	
25.0	89.2	-17.5	2230.0	-437.5	-1561.0	-39025.0	625.0	7956.64	306.25	
25.0	20.0	-30.475	500.0	-761.875	-609.5	-15237.5	625.0	400.0	928.726	
25.0	20.0	-4.525	500.0	-113.125	-90.5	-2262.5	625.0	400.0	20.476	
25.0	20.0	-17.5	500.0	-437.5	-350.0	-8750.0	625.0	400.0	306.25	
Матриця планування Y										
Y1	Y2	Y3								
-33585.4	-33586.4	-33590.4								
-10987.9	-10978.9	-10986.9								
128087.6	128083.6	128087.6								
53248.1	53246.1	53249.1								
168236.6	168234.6	168234.6								
84040.1	84035.1	84039.1								
-372499.4	-372496.4	-372497.4								
-122128.9	-122125.9	-122128.9								
94912.9	94915.9	94912.9								
-69053.5	-69051.5	-69049.5								
88480.76	88489.76	88481.76								
-136839.28	-136847.28	-136844.28								
-53099.576	-53103.576	-53104.576								
-3827.121	-3819.121	-3824.121								
-28970.4	-28965.4	-28972.4								

Середні значення відгуку за рядками:  
-33587.400 -10984.567 128086.267 53247.767 168235.267 84038.100 -372497.733 -122127.900 94913.900 -69051.500 88484.093 -136843.613 -53102.576 -3823.454 -28969.400

Отримане рівняння регресії:  
 $10.042 + 2.244 * X1 + 5.907 * X2 + 8.300 * X3 + 2.201 * X1X2 + 0.801 * X1X3 + 8.801 * X2X3 + 3.600 * X1X2X3 + 5.600 * X11^2 + 1.000 * X22^2 + 3.012 * X33^2 = \hat{y}$

Експериментальні значення:  
-33588.448 -10985.276 128085.656 53247.494 168234.602 84037.774 -372497.961 -122127.789 94914.969 -69051.317 88485.224 -136843.492 -53101.558 -3823.220 -28969.409

Перевірка за критерієм Кохрена:  
Gr = 0.17804878048780481  
Дисперсія однорідна.

Перевірка значущості коефіцієнтів за критерієм Стюдента:  
Значущі коефіцієнти регресії: [10.042, 2.244, 5.907, 8.3, 2.201, 0.801, 8.801, 3.6, 5.6, 1.0, 3.012]  
Незначущі коефіцієнти регресії: []

Значення з отриманими коефіцієнтами:  
-33588.448 -10985.276 128085.656 53247.494 168234.602 84037.774 -372497.961 -122127.789 94914.969 -69051.317 88485.224 -136843.492 -53101.560 -3823.221 -28969.409

Перевірка адекватності за критерієм Фішера:  
Fr = 0.7677522815822589  
Рівняння регресії адекватне.

**Висновок:**

У ході виконання лабораторної роботи проведено повний трьохфакторний експеримент. В ході дослідження було розроблено відповідну програму мовою програмування Python, яка моделює проведення трьохфакторного експерименту, використовуючи рототабельний композиційний план. Отримано адекватну модель – рівняння регресії, проведено 3 статистичні перевірки (критерії Кохрена, Стюдента та Фішера). Результати роботи, наведені у протоколі, підтверджують правильність виконання – кінцеву мету роботи було досягнуто.