

**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 №3**  
**з дисципліни «Методи оптимізації та планування експерименту»**  
**на тему: «ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З**  
**ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ»**

**Виконав:**  
**студент 2-го курсу ФІОТ**  
**групи ІО-92**  
**Гуденко Є. В.**  
**Варіант: 207**

**Перевірив:**  
**асистент**  
**Регіда П. Г.**

## Варіант:

№варіанта	X <sub>1</sub>		X <sub>2</sub>		X <sub>3</sub>	
	min	max	min	max	min	max
207	-5	15	-35	10	-35	-10

## Код програми:

```
import xlrd
import random
import numpy as np
import math
import itertools
from prettytable import PrettyTable

class Lab3:
    def __init__(self):
        self.N = 4
        self.m = 3
        self.x1_min = -5
        self.x1_max = 15
        self.x2_min = -35
        self.x2_max = 10
        self.x3_min = -35
        self.x3_max = -10
        self.x_average_min = (self.x1_min + self.x2_min + self.x3_min)/3
        self.x_average_max = (self.x1_max + self.x2_max + self.x3_max)/3
        self.y_min = 200 + self.x_average_min
        self.y_max = 200 + self.x_average_max

        self.factors_table = [[1, -1, -1, -1],
                               [1, -1, +1, +1],
                               [1, +1, -1, +1],
                               [1, +1, +1, -1]]

        self.generate_matrix()

    def generate_matrix(self):
        self.matrix = [[random.randint(self.y_min, self.y_max) for i in
range(self.m)] for j in range(4)]
        print(
            "Дані варіанту 207 : y_max = {} y_min = {} x1_min = {} x1_max =
{} x2_min = {} x2_max = {} x3_min = {} x3_max = {}".format(
                self.y_max, self.y_min, self.x1_min, self.x1_max, self.x2_min,
self.x2_max, self.x3_min, self.x3_max))
        self.naturalized_factors_table = [[self.x1_min, self.x2_min,
self.x3_min], [self.x1_min, self.x2_max, self.x3_max], [self.x1_max,
self.x2_min, self.x3_max], [self.x1_max, self.x2_max, self.x3_min]]

        table0 = PrettyTable()
        table0.field_names = (["N", "X0", "X1", "X2", "X3"] + ["Y{}".format(i+1)
for i in range(self.m)])
        for i in range(self.N):
            table0.add_row([i+1] + self.factors_table[i] + self.matrix[i])
```

```

print(table0)

table1 = PrettyTable()
table1.field_names = (["X1", "X2", "X3"] + ["Y{}".format(i + 1) for i in
range(self.m)])
for i in range(self.N):
    table1.add_row(self.naturalized_factors_table[i] + self.matrix[i])
print(table1)

self.calculate()

def calculate(self):
    self.average_Y1 = sum(self.matrix[0][j] for j in range(self.m)) / self.m
    self.average_Y2 = sum(self.matrix[1][j] for j in range(self.m)) / self.m
    self.average_Y3 = sum(self.matrix[2][j] for j in range(self.m)) / self.m
    self.average_Y4 = sum(self.matrix[3][j] for j in range(self.m)) / self.m
    self.average_Y = [self.average_Y1, self.average_Y2, self.average_Y3,
self.average_Y4]

    self.mx1 = sum(self.naturalized_factors_table[i][0] for i in
range(self.N)) / self.N
    self.mx2 = sum(self.naturalized_factors_table[i][1] for i in
range(self.N)) / self.N
    self.mx3 = sum(self.naturalized_factors_table[i][2] for i in
range(self.N)) / self.N

    self.my = sum(self.average_Y) / self.N

    self.a1 = sum(self.naturalized_factors_table[i][0] * self.average_Y[i]
for i in range(self.N)) / self.N
    self.a2 = sum(self.naturalized_factors_table[i][1] * self.average_Y[i]
for i in range(self.N)) / self.N
    self.a3 = sum(self.naturalized_factors_table[i][2] * self.average_Y[i]
for i in range(self.N)) / self.N

    self.a11 = sum((self.naturalized_factors_table[i][0]) ** 2 for i in
range(self.N)) / self.N
    self.a22 = sum((self.naturalized_factors_table[i][1]) ** 2 for i in
range(self.N)) / self.N
    self.a33 = sum((self.naturalized_factors_table[i][2]) ** 2 for i in
range(self.N)) / self.N

    self.a12 = sum(self.naturalized_factors_table[i][0] *
self.naturalized_factors_table[i][1] for i in range(self.N)) / self.N
    self.a13 = sum(self.naturalized_factors_table[i][0] *
self.naturalized_factors_table[i][2] for i in range(self.N)) / self.N
    self.a23 = sum(self.naturalized_factors_table[i][1] *
self.naturalized_factors_table[i][2] for i in range(self.N)) / self.N

    equations_sys_coefficients = [[1, self.mx1, self.mx2, self.mx3],
                                [self.mx1, self.a11, self.a12, self.a13],
                                [self.mx2, self.a12, self.a22, self.a23],
                                [self.mx3, self.a13, self.a23, self.a33]]
    equations_sys_free_members = [self.my, self.a1, self.a2, self.a3]
    self.b_coefficients = np.linalg.solve(equations_sys_coefficients,
equations_sys_free_members)
    b_normalized_coefficients = np.array([np.average(self.average_Y),
np.average(self.average_Y *
np.array([i[1] for i in self.factors_table])),
np.average(self.average_Y *
np.array([i[2] for i in self.factors_table])),

```

```

np.array([i[3] for i in self.factors_table]))))

np.average(self.average_Y *
{1:+.2f}*x1 {2:+.2f}*x2 {3:+.2f}*x3".format(*b_normalized_coefficients))
print("\nРівняння регресії для натуралізованих факторів: y = {0:.2f}
{1:+.3f}*x1 {2:+.2f}*x2 {3:+.2f}*x3".format(*self.b_coefficients))

self.cochran_criteria(self.m, self.N, self.matrix)

def cochrans_criteria(self, m, N, y_table):
    print("\nПеревірка рівномірності дисперсій за критерієм Кохрена: m = {},
N = {}".format(m, N))
    cochrans_table = xlrd.open_workbook("Cochran.xls").sheet_by_index(0)
    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
    column = f1
    row = f2-1
    gt = cochrans_table.row_values(row-1)[column-1]/math.pow(10,4)
    print("Gp = {} Gt = {} f1 = {} f2 = {} q = {:.2f}".format(gp, gt, f1,
f2, q))
    if gp < gt:
        print("Gp < Gt => дисперсії рівномірні => переходимо до наступної
статистичної перевірки")
        self.student_criteria(self.m, self.N, self.matrix,
self.factors_table)
    else:
        print("Gp > Gt => дисперсії нерівномірні => змінюємо значення m =>
m = m+1")
        self.m = self.m + 1
        self.generate_matrix()

def student_criteria(self, m, N, y_table, factors_table):
    print("\nПеревірка значимості коефіцієнтів регресії за критерієм
Ст'юдента: m = {}, N = {}".format(m, N))
    student_table = xlrd.open_workbook("Student.xls").sheet_by_index(0)

    average_variation = np.average(list(map(np.var, y_table)))
    standard_deviation_beta_s = math.sqrt(average_variation / N / m)

    y_averages = np.array(list(map(np.average, y_table)))
    x_i = np.array([[el[i] for el in factors_table] for i in
range(len(factors_table))])
    coefficients_beta_s = np.array([np.average(self.average_Y*x_i[i]) for i
in range(len(x_i))])

    print("Оцінки коефіцієнтів  $\beta_s$ : " + ",
".join(list(map(str, coefficients_beta_s))))
    t_i = np.array([abs(coefficients_beta_s[i])/standard_deviation_beta_s
for i in range(len(coefficients_beta_s))])
    print("Коефіцієнти ts: " + ", ".join(list(map(lambda i:
"{:.2f}".format(i), t_i))))
    f3 = (m-1)*N
    p = 0.95

```

```

q = 0.05
t = float(student_table.col_values(3)[f3].replace(",", "."))
self.importance = [True if el > t else False for el in list(t_i)]
# print result data
print("f3 = {} q = {} табл = {}".format(f3, q, t))
beta_i = ["β{}".format(i) for i in range(N)]
importance_to_print = ["важливий" if i else "неважливий" for i in
self.importance]
to_print = list(zip(beta_i, importance_to_print))
x_i_names = [""] + list(itertools.compress(["x{}".format(i) for i in
range(N)], self.importance))[1:]
betas_to_print = list(itertools.compress(coefficients_beta_s,
self.importance))
print("{0[0]} {0[1]} {1[0]} {1[1]} {2[0]} {2[1]} {3[0]}
{3[1]}".format(*to_print))
equation = " ".join(["".join(i) for i in zip(list(map(lambda x:
"{:.2f}".format(x), betas_to_print)), x_i_names)])
print("Рівняння регресії без незначимих членів: y = " + equation)
self.d = len(betas_to_print)
self.factors_table2 = [np.array([1] + list(i)) for i in
self.naturalized_factors_table]
self.fisher_criteria(self.m, self.N, 1, self.factors_table2,
self.matrix, self.b_coefficients, self.importance)

def calculate_theoretical_y(self, x_table, b_coefficients, importance):
    x_table = [list(itertools.compress(row, importance)) for row in x_table]
    b_coefficients = list(itertools.compress(b_coefficients, importance))
    y_vals = np.array([sum(map(lambda x, b: x * b, row, b_coefficients)) for
row in x_table])
    return y_vals

def fisher_criteria(self, m, N, d, factors_table, matrix, b_coefficients,
importance):
    print("\nПеревірка адекватності моделі за критерієм Фішера: m = {}, N =
{} для таблиці y_table".format(m, N))
    fisher_table = xlrd.open_workbook("Fisher.xls").sheet_by_index(0)

    f3 = (m - 1) * N
    f4 = N - d

    theoretical_y = self.calculate_theoretical_y(factors_table,
b_coefficients, importance)
    theoretical_values_to_print = list(
        zip(map(lambda x: "x1 = {0[1]}, x2 = {0[2]}, x3 = {0[3]}".format(x),
factors_table), theoretical_y))

    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]))
    y_averages = np.array(list(map(np.average, matrix)))
    s_ad = m / (N - d) * (sum((theoretical_y - y_averages) ** 2))
    y_variations = np.array(list(map(np.var, matrix)))
    s_v = np.average(y_variations)
    f_p = float(s_ad / s_v)
    f_t = float((fisher_table.row_values(f3) if f3 <= 30 else
fisher_table.row_values(30))[f4].replace(",", "."))
    print("Fp = {}, Ft = {}".format(f_p, f_t))
    print("Fp < Ft => модель адекватна" if f_p < f_t else "Fp > Ft => модель
неадекватна")
    print("\nВиконав: студент групи ІО-92 Гуденко Євген    Варіант 207")

```

Lab3()

## Результати виконання програми:

Дані варіанту 207 :  $y_{\max} = 205.0$   $y_{\min} = 175.0$   $x1_{\min} = -5$   $x1_{\max} = 15$   $x2_{\min} = -35$   $x2_{\max} = 10$   $x3_{\min} = -35$   $x3_{\max} = -10$

```
+-----+
| N | X0 | X1 | X2 | X3 | Y1 | Y2 | Y3 |
+-----+
| 1 | 1 | -1 | -1 | -1 | 201 | 191 | 175 |
| 2 | 1 | -1 | 1 | 1 | 184 | 178 | 199 |
| 3 | 1 | 1 | -1 | 1 | 192 | 200 | 197 |
| 4 | 1 | 1 | 1 | -1 | 195 | 191 | 195 |
+-----+
| X1 | X2 | X3 | Y1 | Y2 | Y3 |
+-----+
| -5 | -35 | -35 | 201 | 191 | 175 |
| -5 | 10 | -10 | 184 | 178 | 199 |
| 15 | -35 | -10 | 192 | 200 | 197 |
| 15 | 10 | -35 | 195 | 191 | 195 |
+-----+
```

Рівняння регресії для нормованих факторів:  $y = 191.50 + 3.50 \cdot x1 - 1.17 \cdot x2 + 0.17 \cdot x3$

Рівняння регресії для натуралізованих факторів:  $y = 189.40 + 0.350 \cdot x1 - 0.05 \cdot x2 + 0.01 \cdot x3$

Перевірка рівномірності дисперсій за критерієм Кохрена:  $m = 3$ ,  $N = 4$

$Gp = 0.5536480686695279$   $Gt = 0.7679$   $f1 = 2$   $f2 = 4$   $q = 0.05$

$Gp < Gt \Rightarrow$  дисперсії рівномірні  $\Rightarrow$  переходимо до наступної статистичної перевірки

Перевірка значимості коефіцієнтів регресії за критерієм Стюдента:  $m = 3$ ,  $N = 4$

Оцінки коефіцієнтів  $\beta_s$ : 191.5, 3.5, -1.16666666666666714, 0.16666666666666714

Коефіцієнти  $t_s$ : 92.19, 1.68, 0.56, 0.08

$f3 = 8$   $q = 0.05$   $t_{\text{табл}} = 2.306$

$\beta_0$  важливий  $\beta_1$  неважливий  $\beta_2$  неважливий  $\beta_3$  неважливий

Рівняння регресії без незначимих членів:  $y = +191.50$

Перевірка адекватності моделі за критерієм Фішера:  $m = 3$ ,  $N = 4$  для таблиці  $y_{\text{table}}$

Теоретичні значення  $y$  для різних комбінацій факторів:

$x1 = -5$ ,  $x2 = -35$ ,  $x3 = -35$ :  $y = 189.40185185185183$

$x1 = -5$ ,  $x2 = 10$ ,  $x3 = -10$ :  $y = 189.40185185185183$

$x1 = 15$ ,  $x2 = -35$ ,  $x3 = -10$ :  $y = 189.40185185185183$

$x1 = 15$ ,  $x2 = 10$ ,  $x3 = -35$ :  $y = 189.40185185185183$

$Fp = 1.3937341705081412$ ,  $Ft = 4.07$

$Fp < Ft \Rightarrow$  модель адекватна

Виконав: студент групи ІО-92 Гуденко Євген      Варіант 207

## Контрольні запитання:

**1. Що називається дробовим факторним експериментом?** Дробовим факторним експериментом називається експеримент з використанням частини повного факторного експерименту.

**2. Для чого потрібно розрахункове значення Кохрена?**

Розрахункове значення Кохрена використовують для перевірки однорідності дисперсій.

### **3. Для чого перевіряється критерій Стюдента?**

За допомогою критерію Стюдента перевіряється значущість коефіцієнтів рівняння регресії.

### **4. Чим визначається критерій Фішера і як його застосовувати?**

Критерій Фішера використовують при перевірці отриманого рівняння регресії досліджуваному об'єкту.

### **Висновок:**

В даній лабораторній роботі проведено дробовий трьохфакторний експеримент з трьома статистичними перевірками і отримано коефіцієнти рівняння регресії.