

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

Лабораторна робота №3

з дисципліни «**Методи оптимізації та планування**»

Виконав:

студент групи ІО-93

Сукач Артем

Залікова книжка: 9328

Перевірив:

ас. Регіда П.Г.

ПРОВЕДЕННЯ ТРЬОХФАКТОРНОГО ЕКСПЕРИМЕНТУ З ВИКОРИСТАННЯМ ЛІНІЙНОГО РІВНЯННЯ РЕГРЕСІЇ

Мета: провести дробовий трьохфакторний експеримент. Скласти матрицю планування, знайти коефіцієнти рівняння регресії, провести 3 статистичні перевірки.

Завдання на лабораторну роботу:

1. Скласти матрицю планування для дробового трьохфакторного експерименту. Провести експеримент в усіх точках факторного простору, повторивши N експериментів, де N – кількість експериментів (рядків матриці планування) в усіх точках факторного простору – знайти значення функції відгуку Y. Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі (випадковим чином).

$$y_{\max} = 200 + x_{\text{ср max}};$$

$$y_{\min} = 200 + x_{\text{ср min}}$$

$$\text{де } x_{\text{ср max}} = \frac{x_{1\max} + x_{2\max} + x_{3\max}}{3}, \quad x_{\text{ср min}} = \frac{x_{1\min} + x_{2\min} + x_{3\min}}{3}$$

2. Знайти коефіцієнти лінійного рівняння регресії. Записати лінійне рівняння регресії.
3. Провести 3 статистичні перевірки.
4. Написати комп'ютерну програму, яка усе це виконує.

Варіант завдання:

325	-20	15	25	45	10	20
-----	-----	----	----	----	----	----

Роздруківка тексту програми:

```
import random, math, numpy as np
```

```
def average(list):
```

```
    average = 0
```

```
    for element in list:
```

```
    average += element / len(list)

return average
```

```
def main():
```

```
    global m, N
```

```
    m = 3
```

```
    N = 4
```

```
def dispersion(list):
```

```
    list_average = average(list)
```

```
    dispersion = 0
```

```
    for element in list:
```

```
        dispersion += (element - list_average) ** 2 / len(list)
```

```
    return dispersion
```

```
def cochrane_criteria():
```

```
    global m, N
```

```
    dispersion_list = [dispersion(y1), dispersion(y2), dispersion(y3), dispersion(y4)]
```

```
    print(f"Дисперсії по рядках:\n{dispersion_list}")
```

```
    gp_denominator = 0
```

```
    for disp in dispersion_list:
```

```
        gp_denominator += disp
```

```
    gp = max(dispersion_list) / gp_denominator
```

```
    f1 = m - 1
```

```
    f2 = N
```

```
    gt = 0.7679
```

```
    if gp < gt:
```

```
    return True
```

```
else:
```

```
    return False
```

```
def students_criteria():
```

```
    global m, N
```

```
    sb = average(dispersion_list)
```

```
    s_beta_2 = sb / (N * m)
```

```
    s_beta = math.sqrt(s_beta_2)
```

```
    beta0 = 0
```

```
    for i in range(N):
```

```
        beta0 += average_list[i] * x0[i] / N
```

```
    beta1 = 0
```

```
    for i in range(N):
```

```
        beta1 += average_list[i] * x1[i] / N
```

```
    beta2 = 0
```

```
    for i in range(N):
```

```
        beta2 += average_list[i] * x2[i] / N
```

```
    beta3 = 0
```

```
    for i in range(N):
```

```
        beta3 += average_list[i] * x3[i] / N
```

```
    t0 = abs(beta0) / s_beta
```

```
    t1 = abs(beta1) / s_beta
```

```
    t2 = abs(beta2) / s_beta
```

```
    t3 = abs(beta3) / s_beta
```

```
f3 = (m - 1) * N
```

```
tt = 2.306
```

```
student_check = {}
```

```
if (t0 > tt):
```

```
    student_check[0] = b[0]
```

```
else:
```

```
    student_check[0] = 0
```

```
if (t1 > tt):
```

```
    student_check[1] = b[1]
```

```
else:
```

```
    student_check[1] = 0
```

```
if (t2 > tt):
```

```
    student_check[2] = b[2]
```

```
else:
```

```
    student_check[2] = 0
```

```
if (t3 > tt):
```

```
    student_check[3] = b[3]
```

```
else:
```

```
    student_check[3] = 0
```

```
return student_check
```

```
def fisher_criteria():
```

```
    global m, N
```

```
    d = 0
```

```
    for key in std_ch:
```

```
        if std_ch[key] != 0: d += 1
```

```
f1 = m - 1
```

```
f2 = N
```

```
f3 = (m - 1) * N
```

```
f4 = N - d
```

```
s2_ad = 0
```

```
for i in range(N):
```

```
    s2_ad += (y_std[i] - average_list[i]) ** 2
```

```
s2_ad *= m / f4
```

```
s2_b = average(dispersion_list)
```

```
fp = s2_ad / s2_b
```

```
ft = 4.5
```

```
if fp > ft:
```

```
    print("Fp > Ft\nОтже, рівняння регресії неадекватно оригіналу при рівні  
значимості 0.05")
```

```
else:
```

```
    print("Fp < Ft\nРівняння регресії адекватно оригіналу")
```

```
x_min = [-20, 25, 10]
```

```
x_max = [15, 45, 20]
```

```
y_min = 200 + average(x_min)
```

```
y_max = 200 + average(x_max)
```

```
x0 = [1, 1, 1, 1]
```

```
x1 = [-1, -1, 1, 1]
```

```
x2 = [-1, 1, -1, 1]
```

```
x3 = [-1, 1, 1, -1]
```

```
print(f"Мінімальне значення ф-ції відгуку: {int(y_min)}")
print(f"Максимальне значення ф-ції відгуку: {int(y_max)}")
```

```
plan_matrix = []
plan_matrix.append([x_max[0], x_min[1], x_min[2]])
plan_matrix.append([x_min[0], x_max[1], x_max[2]])
plan_matrix.append([x_max[0], x_min[1], x_max[2]])
plan_matrix.append([x_max[0], x_max[1], x_min[2]])
```

```
print(f"\nМатриця експерименту:")
for line in plan_matrix:
    print(line)
```

```
y1 = [random.randint(int(y_min), int(y_max)) for _ in range(m)]
y2 = [random.randint(int(y_min), int(y_max)) for _ in range(m)]
y3 = [random.randint(int(y_min), int(y_max)) for _ in range(m)]
y4 = [random.randint(int(y_min), int(y_max)) for _ in range(m)]
```

```
print(f"\nФункції відгуку\n{y1}\n{y2}\n{y3}\n{y4}")
```

```
dispersion_list = [dispersion(y1), dispersion(y2), dispersion(y3), dispersion(y4)]
```

```
y1_average = average(y1)
y2_average = average(y2)
y3_average = average(y3)
y4_average = average(y4)
print(
```

```
    f"\nСередні значення ф-цій відгуку\n\u2081: {y1_average}\n\u2082:
{y2_average}\n\u2083: {y3_average}\n\u2084: {y4_average}")
```

```
average_list = [y1_average, y2_average, y3_average, y4_average]
```

```
mx1 = (plan_matrix[0][0] + plan_matrix[1][0] + plan_matrix[2][0] + plan_matrix[3][0]) / 4
```

```
mx2 = (plan_matrix[0][1] + plan_matrix[1][1] + plan_matrix[2][1] + plan_matrix[3][1]) / 4
```

```
mx3 = (plan_matrix[0][2] + plan_matrix[1][2] + plan_matrix[2][2] + plan_matrix[3][2]) / 4
```

```
my = average(average_list)
```

```
a1 = (plan_matrix[0][0] * y1_average + plan_matrix[1][0] * y2_average + plan_matrix[2][0] * y3_average +
```

```
plan_matrix[3][0] * y4_average) / 4
```

```
a2 = (plan_matrix[0][1] * y1_average + plan_matrix[1][1] * y2_average + plan_matrix[2][1] * y3_average +
```

```
plan_matrix[3][1] * y4_average) / 4
```

```
a3 = (plan_matrix[0][2] * y1_average + plan_matrix[1][2] * y2_average + plan_matrix[2][2] * y3_average +
```

```
plan_matrix[3][2] * y4_average) / 4
```

```
a11 = (plan_matrix[0][0] ** 2 + plan_matrix[1][0] ** 2 + plan_matrix[2][0] ** 2 + plan_matrix[3][0] ** 2) / 4
```

```
a22 = (plan_matrix[0][1] ** 2 + plan_matrix[1][1] ** 2 + plan_matrix[2][1] ** 2 + plan_matrix[3][1] ** 2) / 4
```

```
a33 = (plan_matrix[0][2] ** 2 + plan_matrix[1][2] ** 2 + plan_matrix[2][2] ** 2 + plan_matrix[3][2] ** 2) / 4
```

```
a12 = (plan_matrix[0][0] * plan_matrix[0][1] + plan_matrix[1][0] * plan_matrix[1][1] + plan_matrix[2][0] * plan_matrix[2][1] +
```

```
plan_matrix[3][0] * plan_matrix[3][1]) / 4
```



```
a13 = (plan_matrix[0][0] * plan_matrix[0][2] + plan_matrix[1][0] * plan_matrix[1][2] +  
plan_matrix[2][0] *
```

```
plan_matrix[2][2] + plan_matrix[3][0] * plan_matrix[3][2]) / 4
```

```
a23 = (plan_matrix[0][1] * plan_matrix[0][2] + plan_matrix[1][1] * plan_matrix[1][2] +  
plan_matrix[2][1] *
```

```
plan_matrix[2][2] + plan_matrix[3][1] * plan_matrix[3][2]) / 4
```

```
a21 = a12
```

```
a31 = a13
```

```
a32 = a23
```

```
b0_numerator = np.array([[my, mx1, mx2, mx3],
```

```
    [a1, a11, a12, a13],
```

```
    [a2, a12, a22, a32],
```

```
    [a3, a13, a23, a33]])
```

```
b0_denominator = np.array([[1, mx1, mx2, mx3],
```

```
    [mx1, a11, a12, a13],
```

```
    [mx2, a12, a22, a32],
```

```
    [mx3, a13, a23, a33]])
```

```
b0 = np.linalg.det(b0_numerator) / np.linalg.det(b0_denominator)
```

```
b1_numerator = np.array([[1, my, mx2, mx3],
```

```
    [mx1, a1, a12, a13],
```

```
    [mx2, a2, a22, a32],
```

```
    [mx3, a3, a23, a33]])
```

```
b1_denominator = np.array([[1, mx1, mx2, mx3],
```

```
    [mx1, a11, a12, a13],
```

```
    [mx2, a12, a22, a32],
```

```
    [mx3, a13, a23, a33]])
```

```
b1 = np.linalg.det(b1_numerator) / np.linalg.det(b1_denominator)
```

```
b2_numerator = np.array([[1, mx1, my, mx3],  
                           [mx1, a11, a1, a13],  
                           [mx2, a12, a2, a32],  
                           [mx3, a13, a3, a33]])
```

```
b2_denominator = np.array([[1, mx1, mx2, mx3],  
                            [mx1, a11, a12, a13],  
                            [mx2, a12, a22, a32],  
                            [mx3, a13, a23, a33]])
```

```
b2 = np.linalg.det(b2_numerator) / np.linalg.det(b2_denominator)
```

```
b3_numerator = np.array([[1, mx1, mx2, my],  
                           [mx1, a11, a12, a1],  
                           [mx2, a12, a22, a2],  
                           [mx3, a13, a23, a3]])
```

```
b3_denominator = np.array([[1, mx1, mx2, mx3],  
                            [mx1, a11, a12, a13],  
                            [mx2, a12, a22, a32],  
                            [mx3, a13, a23, a33]])
```

```
b3 = np.linalg.det(b3_numerator) / np.linalg.det(b3_denominator)
```

```
b = [b0, b1, b2, b3]
```

```
print(f"\nPівняння регресії\ny = {b0} + {b1}*x1 + {b2}*x2 + {b3}*x3")
```

```
y1_reg = b0 + b1 * plan_matrix[0][0] + b2 * plan_matrix[0][1] + b3 * plan_matrix[0][2]
```

```
y2_reg = b0 + b1 * plan_matrix[1][0] + b2 * plan_matrix[1][1] + b3 * plan_matrix[1][2]
```

```
y3_reg = b0 + b1 * plan_matrix[2][0] + b2 * plan_matrix[2][1] + b3 * plan_matrix[2][2]
```

```
y4_reg = b0 + b1 * plan_matrix[3][0] + b2 * plan_matrix[3][1] + b3 * plan_matrix[3][2]
```

```

if (round(y1_reg, 0) == round(y1_average, 0)
    and round(y2_reg, 0) == round(y2_average, 0)
    and round(y3_reg, 0) == round(y3_average, 0)
    and round(y4_reg, 0) == round(y4_average, 0)):
    print(f"\nПеревірку закінчено. Коефіцієнти розраховані ПРАВИЛЬНО")
else:
    print(f"\nПеревірку закінчено. Коефіцієнти розраховані НЕПРАВИЛЬНО")

print("-" * 42 + "\nПеревірка за критерієм Кохрена\n")

if cochrane_criteria():
    print("\nЗа критерієм Кохрена дисперсія ОДНОРІДНА\n")
else:
    print(f"\nЗа критерієм Кохрена дисперсія НЕОДНОРІДНА\nЗбільшимо значення
m")
    m += 1
    main()

print("-" * 42 + "\nПеревірка значущості коефіцієнтів за критерієм Стюдента\n")

std_ch = students_criteria()
print(f"\n{std_ch}")

print(f"\nРівняння регресії\ny = {std_ch[0]} + {std_ch[1]}*x1 + {std_ch[2]}*x2 +
{std_ch[3]}*x3")

y1_std = std_ch[0] + std_ch[1] * plan_matrix[0][0] + std_ch[2] * plan_matrix[0][1] +
std_ch[3] * plan_matrix[0][2]

y2_std = std_ch[0] + std_ch[1] * plan_matrix[1][0] + std_ch[2] * plan_matrix[1][1] +
std_ch[3] * plan_matrix[1][2]

```

```
y3_std = std_ch[0] + std_ch[1] * plan_matrix[2][0] + std_ch[2] * plan_matrix[2][1] +  
std_ch[3] * plan_matrix[2][2]
```

```
y4_std = std_ch[0] + std_ch[1] * plan_matrix[3][0] + std_ch[1] * plan_matrix[3][1] +  
std_ch[3] * plan_matrix[3][2]
```

```
y_std = [y1_std, y2_std, y3_std, y4_std]
```

```
print(f"\ny\u2081 = {y1_std}\ny\u2082 = {y2_std}\ny\u2083 = {y3_std}\ny\u2084 =  
{y4_std}")
```

```
print("-" * 42 + "\nПеревірка за критерієм Фішера\n")
```

```
fisher_criteria()
```

```
main()
```

Результати роботи програми:

```
Мінімальне значення ф-ції відгуку: 205  
Максимальне значення ф-ції відгуку: 226  
  
Матриця експерименту:  
[15, 25, 10]  
[-20, 45, 20]  
[15, 25, 20]  
[15, 45, 10]  
  
Функції відгуку  
[215, 208, 211]  
[209, 212, 218]  
[226, 207, 213]  
[214, 216, 226]  
  
Середні значення ф-цій відгуку  
y1: 211.33333333333331  
y2: 213.0  
y3: 215.33333333333331  
y4: 218.66666666666663  
  
Рівняння регресії  
y = 194.02380952380858 + 0.276190476190485*x1 + 0.366666666666668146*x2 + 0.400000000000003727*x3  
  
Перевірку закінчено. Коефіцієнти розраховані ПРАВИЛЬНО  
-----  
Перевірка за критерієм Кохрена  
  
Дисперсії по рядках:  
[8.222222222222223, 14.0, 62.88888888888889, 27.555555555555557]  
  
За критерієм Кохрена дисперсія ОДНОРІДНА
```

Перевірка значущості коефіцієнтів за критерієм Стьюдента

{0: 194.02380952380858, 1: 0, 2: 0, 3: 0}

Рівняння регресії

$$y = 194.02380952380858 + 0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3$$

$$y_1 = 194.02380952380858$$

$$y_2 = 194.02380952380858$$

$$y_3 = 194.02380952380858$$

$$y_4 = 194.02380952380858$$

Перевірка за критерієм Фішера

$$F_p > F_t$$

Отже, рівняння регресії неадекватно оригіналу при рівні значимості 0.05

Process finished with exit code 0

Контрольні запитання

1) *Що називається дробовим факторним експериментом?*

У деяких випадках немає необхідності проводити повний факторний експеримент (ПФЕ). Якщо буде використовуватися лінійна регресія, то можливо зменшити кількість рядків матриці ПФЕ до кількості коефіцієнтів регресійної моделі. Кількість дослідів слід скоротити, використовуючи для планування так звані регулярні дробові репліки від повного факторного експерименту, що містять відповідну кількість дослідів і зберігають основні властивості матриці планування – це означає дробовий факторний експеримент (ДФЕ).

2) *Для чого потрібно розрахункове значення Кохрена?*

Критерій Кохрена використовують для порівняння трьох і більше виборок однакового обсягу n .

3) *Для чого перевіряється критерій Стьюдента?*

Якщо теоретичний коефіцієнт $b_i = 0$, це означає, що в апроксимуючому поліномі відповідний доданок (фактор) відсутній. Чим менше значення b_i , тим менше вплив відповідного фактора. За критерієм Стьюдента перевіряється значущість коефіцієнтів.

4) Чим визначається критерій Фішера і як його застосовувати?

Отримане рівняння регресії необхідно перевірити на адекватність досліджуваному об'єкту. Для цієї мети необхідно оцінити, наскільки відрізняються середні значення y вихідної величини, отриманої в точках факторного простору, і значення y , отриманого з рівняння регресії в тих самих точках факторного простору. Для цього використовують дисперсію адекватності. Адекватність моделі перевіряють за F-критерієм Фішера, який дорівнює відношенню дисперсії адекватності до дисперсії відтворюваності.

Висновки:

В ході виконання лабораторної роботи було проведено трьохфакторний експеримент. Складено матрицю планування, знайдено коефіцієнти рівняння регресії, проведено 3 статистичні перевірки.