

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

Лабораторна робота №4  
з дисципліни «**Методи оптимізації та планування**»  
**тема:** «Проведення трьохфакторного експерименту при використанні рівняння  
регресії з урахуванням ефекту взаємодії»

Виконала:  
студентка групи ІО-93  
номер списку - 18  
Май Тієн Ноанг

Перевірив:  
ас. Регіда П.Г.

**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.

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

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

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

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

### Варіант завдання:

318	-10	50	20	60	-10	10
-----	-----	----	----	----	-----	----

### Роздруківка тексту програми:

```
import random
import numpy as np
import sklearn.linear_model as lm
from scipy.stats import f, t
from numpy.linalg import solve

def regression(x, b):
    y = sum([x[i] * b[i] for i in range(len(x))])
    return y

def dispersion(y, y_aver, n, m):
    res = []
    for i in range(n):
        s = sum([(y_aver[i] - y[i][j]) ** 2 for j in range(m)]) / m
        res.append(round(s, 3))
    return res
```

```

def planing_matrix_interaction_effect(n, m):
    x_normalized = [[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]]
    y = np.zeros(shape=(n, m), dtype=np.int64)
    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(y_min, y_max)

    for x in x_normalized:
        x.append(x[1] * x[2])
        x.append(x[1] * x[3])
        x.append(x[2] * x[3])
        x.append(x[1] * x[2] * x[3])

    x_normalized = np.array(x_normalized[:len(y)])
    x = np.ones(shape=(len(x_normalized), len(x_normalized[0])), dtype=np.int64)

    for i in range(len(x_normalized)):
        for j in range(1, 4):
            if x_normalized[i][j] == -1:
                x[i][j] = x_range[j - 1][0]
            else:
                x[i][j] = x_range[j - 1][1]

    for i in range(len(x)):
        x[i][4] = x[i][1] * x[i][2]
        x[i][5] = x[i][1] * x[i][3]
        x[i][6] = x[i][2] * x[i][3]
        x[i][7] = x[i][1] * x[i][3] * x[i][2]

    print(f'\nМатриця планування для n = {n}, m = {m}:')
    print('\n3 кодованими значеннями факторів:')
    print('\n      X0      X1      X2      X3  X1X2  X1X3  X2X3  X1X2X3   Y1      Y2      Y3')
    print(np.concatenate((x, y), axis=1))
    print('\nНормовані значення факторів:\n')
    print(x_normalized)

    return x, y, x_normalized

def find_coef(X, Y, norm=False):
    skm = lm.LinearRegression(fit_intercept=False)
    skm.fit(X, Y)
    B = skm.coef_

```

```

if norm == 1:
    print('\nКоефіцієнти рівняння регресії з нормованими X:')
else:
    print('\nКоефіцієнти рівняння регресії:')
B = [round(i, 3) for i in B]
print(B)
return B

def bs(x, y, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]
    for i in range(7):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_aver)) / n
        res.append(b)
    return res

def kriteriy_studenta2(x, y, y_aver, n, m):
    S_kv = dispersion(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n
    s_Bs = (s_kv_aver / n / m) ** 0.5
    Bs = bs(x, y, y_aver, n)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts

def kriteriy_studenta(x, y_average, n, m, dispersion):
    dispersion_average = sum(dispersion) / n
    s_beta_s = (dispersion_average / n / m) ** 0.5

    beta = [sum(1 * y for y in y_average) / n]
    for i in range(3):
        b = sum(j[0] * j[1] for j in zip(x[:, i], y_average)) / n
        beta.append(b)

    t = [round(abs(b) / s_beta_s, 3) for b in beta]

    return t

def kriteriy_fishera(y, y_average, y_new, n, m, d, dispersion):
    S_ad = m / (n - d) * sum([(y_new[i] - y_average[i])**2 for i in range(len(y))])
    dispersion_average = sum(dispersion) / n

    return S_ad / dispersion_average

def check(X, Y, B, n, m, norm=False):

    f1 = m - 1

```

```

f2 = n
f3 = f1 * f2
q = 0.05

y_aver = [round(sum(i) / len(i), 3) for i in Y]
print('\nСереднє значення y:', y_aver)

dispersion_arr = dispersion(Y, y_aver, n, m)

qq = (1 + 0.95) / 2
student_cr_table = t.ppf(df=f3, q=qq)

ts = kriteriy_students2(X[:, 1:], Y, y_aver, n, m)

temp_cohren = f.ppf(q=(1 - q / f1), dfn=f2, dfd=(f1 - 1) * f2)
cohren_cr_table = temp_cohren / (temp_cohren + f1 - 1)
Gp = max(dispersion_arr) / sum(dispersion_arr)

print('Дисперсія y:', dispersion_arr)

print(f'Gp = {Gp}')
if Gp < cohren_cr_table:
    print(f'З ймовірністю {1-q} дисперсії однорідні.')
else:
    print("Необхідно збільшити кількість дослідів")
    m += 1
    with_interaction_effect(n, m)

print('\nКритерій Стюдента:\n', ts)
res = [t for t in ts if t > student_cr_table]
final_k = [B[i] for i in range(len(ts)) if ts[i] in res]
print('\nКоефіцієнти {} статистично незначущі, тому ми виключаємо їх з рівняння
.'.format(
    [round(i, 3) for i in B if i not in final_k]))

y_new = []
for j in range(n):
    y_new.append(regression([X[j][i] for i in range(len(ts)) if ts[i] in res],
final_k))

print(f'\nЗначення "y" з коефіцієнтами {final_k}')
print(y_new)

d = len(res)
if d >= n:
    print('\nF4 <= 0')
    print('')
    return
f4 = n - d

Fp = kriteriy_fishera(Y, y_aver, y_new, n, m, d, dispersion_arr)

```

```

Ft = f.ppf(dfn=f4, dfd=f3, q=1 - 0.05)

print('\nПеревірка адекватності за критерієм Фішера')
print('Fp =', Fp)
print('Ft =', Ft)
if Fp < Ft:
    print('Математична модель адекватна експериментальним даним')
    return True
else:
    print('Математична модель не адекватна експериментальним даним')
    return False

def with_interaction_effect(n, m):
    X, Y, X_norm = planing_matrix_interaction_effect(n, m)

    y_aver = [round(sum(i) / len(i), 3) for i in Y]

    B_norm = find_coef(X_norm, y_aver, norm=True)

    return check(X_norm, Y, B_norm, n, m, norm=True)

def planning_matrix_linear(n, m, x_range):
    x_normalized = np.array([[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]])

    y = np.zeros(shape=(n,m))
    for i in range(n):
        for j in range(m):
            y[i][j] = random.randint(y_min,y_max)

    x_normalized = x_normalized[:len(y)]

    x = np.ones(shape=(len(x_normalized), len(x_normalized[0])))
    for i in range(len(x_normalized)):
        for j in range(1, len(x_normalized[i])):
            if x_normalized[i][j] == -1:
                x[i][j] = x_range[j-1][0]
            else:
                x[i][j] = x_range[j-1][1]

    print('\nМатриця планування: ' )
    print('\n      X0  X1  X2  X3  Y1  Y2  Y3  ')
    print(np.concatenate((x, y), axis=1))

```

```

    return x, y, x_normalized

def regression_equation(x, y, n):
    y_average = [round(sum(i) / len(i), 2) for i in y]

    mx= [(sum(x[:, 1]) / n),
          (sum(x[:, 2]) / n),
          (sum(x[:, 3]) / n)]

    my = sum(y_average) / n

    a = [
        (sum([y_average[i] * x[i][1] for i in range(len(x))]) / n),
        (sum([y_average[i] * x[i][2] for i in range(len(x))]) / n),
        (sum([y_average[i] * x[i][3] for i in range(len(x))]) / n)
    ]

    a12 = sum([x[i][1] * x[i][2] for i in range(len(x))]) / n
    a13 = sum([x[i][1] * x[i][3] for i in range(len(x))]) / n
    a23 = sum([x[i][2] * x[i][3] for i in range(len(x))]) / n

    a11 = sum([i ** 2 for i in x[:, 1]]) / n
    a22 = sum([i ** 2 for i in x[:, 2]]) / n
    a33 = sum([i ** 2 for i in x[:, 3]]) / n

    X = [[1, mx[0], mx[1], mx[2]], [mx[0], a11, a12, a13], [mx[1], a12, a22, a23],
    [mx[2], a13, a23, a33]]
    Y = [my, a[0], a[1], a[2]]
    B = [round(i, 2) for i in solve(X, Y)]

    print('\nPівняння регресії:')
    print(f'y = {B[0]} + {B[1]}*x1 + {B[2]}*x2 + {B[3]}*x3')

    return y_average, B

def linear(n, m):
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

    x, y, x_norm = planning_matrix_linear(n, m, x_range)

    y_average, B = regression_equation(x, y, n)

    dispersion_arr = dispersion(y, y_average, n, m)

    temp_cohren = f.ppf(q=(1 - q / f1), dfn=f2, dfd=(f1 - 1) * f2)

```

```

cohren_cr_table = temp_cohren / (temp_cohren + f1 - 1)
Gp = max(dispersion_arr) / sum(dispersion_arr)

print('\nПеревірка за критерієм Кохрена:\n')
print(f'Розрахункове значення: Gp = {Gp}')
    f'\nТабличне значення: Gt = {cohren_cr_table}')
if Gp < cohren_cr_table:
    print(f'З ймовірністю {1-q} дисперсії однорідні.')
else:
    print("Необхідно збільшити ксть дослідів")
    m += 1
    linear(n, m)

qq = (1 + 0.95) / 2
student_cr_table = t.ppf(df=f3, q=qq)
student_t = kriteriy_studentsa(x_norm[:,1:], y_average, n, m, dispersion_arr)

print('\nТабличне значення критерій Стюдента:\n', student_cr_table)
print('Розрахункове значення критерій Стюдента:\n', student_t)
res_student_t = [temp for temp in student_t if temp > student_cr_table]
final_coefficients = [B[student_t.index(i)] for i in student_t if i in res_stud
ent_t]
print('Коефіцієнти {} статистично незначущі.'.
    format([i for i in B if i not in final_coefficients]))

y_new = []
for j in range(n):
    y_new.append(regression([x[j][student_t.index(i)] for i in student_t if i i
n res_student_t], final_coefficients))

print(f'\nОтримаємо значення рівня регресії для {m} дослідів: ')
print(y_new)

d = len(res_student_t)
f4 = n - d
Fp = kriteriy_fishera(y, y_average, y_new, n, m, d, dispersion_arr)
Ft = f.ppf(dfn=f4, dfd=f3, q=1 - 0.05)

print('\nПеревірка адекватності за критерієм Фішера:\n')
print('Розрахункове значення критерія Фішера: Fp =', Fp)
print('Табличне значення критерія Фішера: Ft =', Ft)
if Fp < Ft:
    print('Математична модель адекватна експериментальним даним')
    return True
else:
    print('Математична модель не адекватна експериментальним даним')
    return False

def main(n, m):
    main_1 = linear(n, m)
    if not main_1:

```



```

        interaction_effect = with_interaction_effect(n, m)
        if not interaction_effect:
            main(n, m)

if __name__ == '__main__':
    x_range = ((-10, 50), (20, 60), (-10, 10))

    y_max = 200 + int(sum([x[1] for x in x_range]) / 3)
    y_min = 200 + int(sum([x[0] for x in x_range]) / 3)

    main(8, 3)

```

## Результати роботи програми:

Матриця планування:

	X0	X1	X2	X3	Y1	Y2	Y3
[1]	1.	-10.	20.	-10.	237.	236.	212.
[2]	1.	-10.	60.	10.	223.	224.	211.
[3]	1.	50.	20.	10.	217.	217.	237.
[4]	1.	50.	60.	-10.	206.	230.	204.
[5]	1.	-10.	20.	10.	217.	218.	205.
[6]	1.	-10.	60.	-10.	235.	217.	220.
[7]	1.	50.	20.	-10.	238.	239.	234.
[8]	1.	50.	60.	10.	226.	222.	207.

Рівняння регресії:

$y = 228.39 + 0.03 \cdot x_1 + -0.17 \cdot x_2 + -0.35 \cdot x_3$

Перевірка за критерієм Кохрена:

Розрахункове значення:  $G_p = 0.24685540431779393$

Табличне значення:  $G_t = 0.815948432359917$

З ймовірністю 0.95 дисперсії однорідні.

Табличне значення критерій Стюдента:

2.119905299221011

Розрахункове значення критерій Стюдента:

[129.471, 0.535, 1.992, 2.04]

Коефіцієнти [0.03, -0.17, -0.35] статистично незначущі.

Отримуємо значення рівня регресії для 3 дослідів:

[228.39, 228.39, 228.39, 228.39, 228.39, 228.39, 228.39, 228.39]

Перевірка адекватності за критерієм Фішера:

Розрахункове значення критерія Фішера:  $F_p = 4.56415460365471$

Табличне значення критерія Фішера:  $F_t = 2.6571966002210865$

Розрахункове значення критерія Фішера:  $F_p = 4.56415460365471$   
Табличне значення критерія Фішера:  $F_t = 2.6571966002210865$   
Математична модель не адекватна експериментальним даним

Матриця планування для  $n = 8$ ,  $m = 3$ :

З кодованими значеннями факторів:

	X0	X1	X2	X3	X1X2	X1X3	X2X3	X1X2X3	Y1	Y2	Y3
[[	1	-10	20	-10	-200	100	-200	2000	222	234	
211]											
[	1	-10	60	10	-600	-100	600	-6000	212	215	
219]											
[	1	50	20	10	1000	500	200	10000	202	212	
200]											
[	1	50	60	-10	3000	-500	-600	-30000	235	240	
228]											
[	1	-10	20	10	-200	-100	200	-2000	222	231	
210]											
[	1	-10	60	-10	-600	100	-600	6000	207	204	
227]											
[	1	50	20	-10	1000	-500	-200	-10000	211	216	
223]											
[	1	50	60	10	3000	500	600	30000	240	220	
240]]											

Нормовані значення факторів:

```
[[ 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]]
```

Коефіцієнти рівняння регресії з нормованими X:

[220.042, 2.208, 3.875, -1.458, 7.708, -1.792, 1.875, 0.875]

Середнє значення у: [222.333, 215.333, 204.667, 234.333, 221.0, 212.667, 216.667, 233.333]

Дисперсія у: [88.222, 8.222, 27.556, 24.222, 74.0, 104.222, 24.222, 88.889]

$G_p = 0.23710798421130463$

З ймовірністю 0.95 дисперсії однорідні.

Критерій Стюдента:

[145.428, 1.46, 2.561, 0.964, 5.094, 1.184, 1.239, 0.578]

Коефіцієнти [2.208, -1.458, -1.792, 1.875, 0.875] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "у" з коефіцієнтами [220.042, 3.875, 7.708]

[223.875, 216.209, 208.459, 231.625, 223.875, 216.209, 208.459, 231.625]

Перевірка адекватності за критерієм Фішера

$F_p = 1.266272498322167$

$F_t = 2.852409165081986$

Математична модель адекватна експериментальним даним

PS C:\Users\maxma\PycharmProjects\snippetproject-basic-master> █