

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 № 5

з дисципліни **«Методи оптимізації та планування експерименту»** на тему
**«Проведення трьохфакторного експерименту при
використанні рівняння регресії з урахуванням квадратичних членів
(центральний ортогональний композиційний план)»**

Виконала:
студентка II курсу ФІОТ
групи ІО-93
Дяченко Віта
У списку групи №9

Перевірив:
Регіда П. Г.

Київ – 2021

Мета роботи: Провести трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план. Знайти рівняння регресії, яке буде адекватним для опису об'єкту.

Завдання:

1. Взяти рівняння з урахуванням квадратичних членів.
2. Скласти матрицю планування для ОЦКП
3. Провести експеримент у всіх точках факторного простору (знайти значення функції відгуку Y). Значення функції відгуку знайти у відповідності з варіантом діапазону, зазначеного далі. Варіанти вибираються по номеру в списку в журналі викладача.
4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант:

Варіант:	min	max	min	max	min	max
309	0	10	-6	6	-7	9

Код програми:

```
import random
import sklearn.linear_model as lm
from scipy.stats import f, t
from functools import partial
from pyDOE2 import *

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

x_range = ((0, 10), (-6, 6), (-7, 9))

x_aver_max = sum([x[1] for x in x_range]) / 3
x_aver_min = sum([x[0] for x in x_range]) / 3

y_max = 200 + int(x_aver_max)
y_min = 200 + int(x_aver_min)

# квадратна дисперсія
def s_kv(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 plan_matrix5(n, m):
    print(f'\nГенеруємо матрицю планування для n = {n}, m = {m}')

    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)

    if n > 14:
        no = n - 14
    else:
        no = 1
    x_norm = ccdesign(3, center=(0, no))
    x_norm = np.insert(x_norm, 0, 1, axis=1)

    for i in range(4, 11):
        x_norm = np.insert(x_norm, i, 0, axis=1)

    l = 1.215

    for i in range(len(x_norm)):
        for j in range(len(x_norm[i])):
            if x_norm[i][j] < -1 or x_norm[i][j] > 1:
                if x_norm[i][j] < 0:
                    x_norm[i][j] = -1
                else:
                    x_norm[i][j] = 1

    def add_sq_nums(x):
        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]
            x[i][8] = x[i][1] ** 2
            x[i][9] = x[i][2] ** 2
            x[i][10] = x[i][3] ** 2
        return x

    x_norm = add_sq_nums(x_norm)

    x = np.ones(shape=(len(x_norm), len(x_norm[0])), dtype=np.int64)
    for i in range(8):
        for j in range(1, 4):
            if x_norm[i][j] == -1:
                x[i][j] = x_range[j - 1][0]

```

```

        else:
            x[i][j] = x_range[j - 1][1]

    for i in range(8, len(x)):
        for j in range(1, 3):
            x[i][j] = (x_range[j - 1][0] + x_range[j - 1][1]) / 2

    dx = [x_range[i][1] - (x_range[i][0] + x_range[i][1]) / 2 for i in
range(3)]

    x[8][1] = 1 * dx[0] + x[9][1]
    x[9][1] = -1 * dx[0] + x[9][1]
    x[10][2] = 1 * dx[1] + x[9][2]
    x[11][2] = -1 * dx[1] + x[9][2]
    x[12][3] = 1 * dx[2] + x[9][3]
    x[13][3] = -1 * dx[2] + x[9][3]

    x = add_sq_nums(x)

    print('\nX:\n', x)
    print('\nX нормоване:\n')
    for i in x_norm:
        print([round(x, 2) for x in i])
    print('\nY:\n', y)

    return x, y, x_norm

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)
    print('\nРезультат рівняння зі знайденими коефіцієнтами:\n', np.dot(X, B))
    return B

def kriteriy_cochrana(y, y_aver, n, m):
    f1 = m - 1
    f2 = n
    q = 0.05
    S_kv = s_kv(y, y_aver, n, m)
    Gp = max(S_kv) / sum(S_kv)
    print('\nПеревірка за критерієм Кохрена')
    return Gp

```

```

def cohren(f1, f2, q=0.05):
    q1 = q / f1
    fisher_value = f.ppf(q=1 - q1, dfn=f2, dfd=(f1 - 1) * f2)
    return fisher_value / (fisher_value + f1 - 1)

# оцінки коефіцієнтів
def bs(x, y_aver, n):
    res = [sum(1 * y for y in y_aver) / n]

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

def kriteriy_studenta(x, y, y_aver, n, m):
    S_kv = s_kv(y, y_aver, n, m)
    s_kv_aver = sum(S_kv) / n

    # статистична оцінка дисперсії
    s_Bs = (s_kv_aver / n / m) ** 0.5 # статистична оцінка дисперсії
    Bs = bs(x, y_aver, n)
    ts = [round(abs(B) / s_Bs, 3) for B in Bs]

    return ts

def kriteriy_fishera(y, y_aver, y_new, n, m, d):
    S_ad = m / (n - d) * sum([(y_new[i] - y_aver[i]) ** 2 for i in
range(len(y))])
    S_kv = s_kv(y, y_aver, n, m)
    S_kv_aver = sum(S_kv) / n

    return S_ad / S_kv_aver

def check(X, Y, B, n, m):
    print('\n\tПеревірка рівняння:')
    f1 = m - 1
    f2 = n
    f3 = f1 * f2
    q = 0.05

    ### табличні значення
    student = partial(t.ppf, q=1 - q)
    t_student = student(df=f3)

    G_kr = cohren(f1, f2)
    ###

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

```

```

disp = s_kv(Y, y_aver, n, m)
print('Дисперсія y:', disp)

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

ts = kriteriy_students(X[:, 1:], Y, y_aver, n, m)
print('\nКритерій Стьюдента:\n', ts)
res = [t for t in ts if t > t_student]
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

F_p = kriteriy_fishera(Y, y_aver, y_new, n, m, d)

fisher = partial(f.ppf, q=0.95)
f_t = fisher(dfn=f4, dfd=f3) # табличне знач
print('\nПеревірка адекватності за критерієм Фішера')
print('Fp =', F_p)
print('F t =', f_t)
if F_p < f_t:
    print('Математична модель адекватна експериментальним даним')
else:
    print('Математична модель не адекватна експериментальним даним')

def main(n, m):
    X5, Y5, X5_norm = plan_matrix5(n, m)

    y5_aver = [round(sum(i) / len(i), 3) for i in Y5]

```

```
B5 = find_coef(X5, y5_aver)
```

```
check(X5_norm, Y5, B5, n, m)
```

```
if name == 'main':  
    main(15, 3)
```

Результат роботи:

Гереруємо матрицю планування для $n = 15$, $m = 3$

X:

```
[[ 1  0 -6 -7  0  0 42  0  0 36 49]  
[ 1 10 -6 -7 -60 -70 42 420 100 36 49]  
[ 1  0  6 -7  0  0 -42  0  0 36 49]  
[ 1 10  6 -7 60 -70 -42 -420 100 36 49]  
[ 1  0 -6  9  0  0 -54  0  0 36 81]  
[ 1 10 -6  9 -60 90 -54 -540 100 36 81]  
[ 1  0  6  9  0  0 54  0  0 36 81]  
[ 1 10  6  9 60 90 54 540 100 36 81]  
[ 1 11  0  1  0 11  0  0 121  0  1]  
[ 1 -1  0  1  0 -1  0  0  1  0  1]  
[ 1  5  7  1 35  5  7 35 25 49  1]  
[ 1  5 -7  1 -35  5 -7 -35 25 49  1]  
[ 1  5  0 10  0 50  0  0 25  0 100]  
[ 1  5  0 -8  0 -40  0  0 25  0 64]  
[ 1  5  0  1  0  5  0  0 25  0 1]]
```

X нормоване:

```
[1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, -1.0, 1.0, 1.0, 1.0]  
[1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0, 1.0]  
[1.0, -1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0, 1.0]  
[1.0, 1.0, 1.0, -1.0, 1.0, -1.0, -1.0, -1.0, 1.0, 1.0, 1.0]  
[1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0, 1.0]  
[1.0, 1.0, -1.0, 1.0, -1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1.0]  
[1.0, -1.0, 1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1.0]  
[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]  
[1.0, -1.22, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0, 1.48, 0.0, 0.0]  
[1.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0, 0.0]  
[1.0, 0.0, -1.22, 0.0, -0.0, 0.0, -0.0, -0.0, 0.0, 1.48, 0.0]  
[1.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48, 0.0]  
[1.0, 0.0, 0.0, -1.22, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 1.48]  
[1.0, 0.0, 0.0, 1.22, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.48]  
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

```
Y:
[[200. 206. 208.]
 [199. 205. 199.]
 [208. 201. 197.]
 [207. 197. 200.]
 [204. 196. 204.]
 [204. 206. 199.]
 [204. 202. 206.]
 [201. 204. 202.]
 [205. 196. 208.]
 [199. 200. 203.]
 [198. 204. 205.]
 [204. 203. 199.]
 [198. 204. 199.]
 [208. 204. 196.]
 [202. 202. 204.]]

Коефіцієнти рівняння регресії:
[202.218, -0.107, -0.02, -0.077, 0.002, 0.014, 0.028, -0.003, 0.007, 0.012, -0.001]

Результат рівняння зі знайденими коефіцієнтами:
[204.436 201.706 201.844 201.874 200.484 202.874 203.268 202.658 201.964
 202.24 202.459 202.417 201.688 201.85 201.85 ]

Перевірка рівняння:

Середнє значення y: [204.667, 201.0, 202.0, 201.333, 201.333, 203.0, 204.0, 202.333, 203.0, 200.667, 202.333, 202.0, 200.333, 202.667, 202.667]
Дисперсія y: [11.556, 8.0, 20.667, 17.556, 14.222, 8.667, 2.667, 1.556, 26.0, 2.889, 9.556, 4.667, 6.889, 24.889, 0.889]

Перевірка за критерієм Кохрена
Gr = 0.16182236883052217
З ймовірністю 0.95 дисперсії однорідні.

Критерій Стьюдента:
[414.489, 0.98, 0.101, 0.615, 0.046, 0.592, 0.592, 0.866, 302.746, 302.88, 302.612]

Коефіцієнти [-0.107, -0.02, -0.077, 0.002, 0.014, 0.028, -0.003] статистично незначущі, тому ми виключаємо їх з рівняння.

Значення "y" з коефіцієнтами [202.218, 0.007, 0.012, -0.001]
[202.236, 202.236, 202.236, 202.236, 202.236, 202.236, 202.236, 202.236, 202.236, 202.228333575, 202.228333575, 202.2357147, 202.2357147, 202.21652377499998, 202.21652377499998, 202.218]

Перевірка адекватності за критерієм Фішера
Fr = 0.5061560110279828
F_t = 2.125558760875511
Математична модель адекватна експериментальним даним
```

Висновок: в даній лабораторній роботі я провела трьохфакторний експеримент з урахуванням квадратичних членів, використовуючи центральний ортогональний композиційний план.