



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №5
Проведення трьохфакторного експерименту при використанні
рівняння регресії з урахуванням квадратичних членів
(центральний ортогональний композиційний план)

Виконала

студентка групи ІТ-91:

Луцай Катерина

Перевірила:

Сокульський О. Є.

Київ 2022

Мета: 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.

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

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

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

4. Розрахувати коефіцієнти рівняння регресії і записати його.
5. Провести 3 статистичні перевірки.

Варіант 15

```
import numpy as np

x1_min, x1_max = -1, 2
x2_min, x2_max = -9, 6
x3_min, x3_max = -5, 8

x01 = (x1_min + x1_max)/2
x02 = (x2_min + x2_max)/2
x03 = (x3_min + x3_max)/2

y_min = 200 + (x1_min + x2_min + x3_min)/3
y_max = 200 + (x1_max + x2_max + x3_max)/3

l = 1.215

n = 15
k = 3
c = 11
m = 3

norm_factors = 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, 0, 0],
                          [1, 0, 0],
                          [0, -1, 0],
                          [0, 1, 0],
                          [0, 0, -1],
```

```
[0, 0, 1],
[0, 0, 0]])
```

```
# інтеракції між факторами
```

```
def interact_factors(factors, expanded=False):
    full_factors = np.zeros((n, 4)) if not expanded else np.zeros((n, 7))
    full_factors[:, 0] = factors[:, 0] * factors[:, 1]
    full_factors[:, 1] = factors[:, 0] * factors[:, 2]
    full_factors[:, 2] = factors[:, 1] * factors[:, 2]
    full_factors[:, 3] = factors[:, 0] * factors[:, 1] * factors[:, 2]
    if expanded:
        for i in range(3):
            full_factors[:, 4+i] = np.power(factors[:, i], 2)
    return full_factors
```

```
# отримання натуральних факторів з нормованих
```

```
def natural(norm_factors):
    natural = np.zeros_like(norm_factors)
    natural[:, 0][norm_factors[:, 0] == 1] = x1_max
    natural[:, 1][norm_factors[:, 1] == 1] = x2_max
    natural[:, 2][norm_factors[:, 2] == 1] = x3_max

    natural[:, 0][norm_factors[:, 0] == -1] = x1_min
    natural[:, 1][norm_factors[:, 1] == -1] = x2_min
    natural[:, 2][norm_factors[:, 2] == -1] = x3_min

    natural[:, 0][norm_factors[:, 0] == 0] = x01
    natural[:, 1][norm_factors[:, 1] == 0] = x02
    natural[:, 2][norm_factors[:, 2] == 0] = x03

    natural[:, 0][norm_factors[:, 0] == 1] = 1 * (x1_max - x01) + x01
    natural[:, 1][norm_factors[:, 1] == 1] = 1 * (x2_max - x02) + x02
    natural[:, 2][norm_factors[:, 2] == 1] = 1 * (x3_max - x03) + x03
    natural[:, 0][norm_factors[:, 0] == -1] = -1 * (x1_max - x01) + x01
    natural[:, 1][norm_factors[:, 1] == -1] = -1 * (x2_max - x02) + x02
    natural[:, 2][norm_factors[:, 2] == -1] = -1 * (x3_max - x03) + x03
    return natural
```

```
# регресія за факторами та коефіцієнтами
```

```
def regression(matrix, coefs, n, c):
    def func(factors, coefs, c):
        y = coefs[0]
        for i in range(c):
            y += coefs[i+1] * factors[i]
        return y
    values = np.zeros(n)
    for f in range(n):
        values[f] = func(matrix[f], coefs, c)
    return values
```

```
# вирішення натурального СЛР
```

```
def solve_coef(factors, values):
    A = np.zeros((c, c))
    x_mean = np.mean(factors, axis=0)
    for i in range(1, c):
```

```

        for j in range(1, c):
            A[j, i] = np.mean(factors[:, i-1] * factors[:, j-1])
        A[0, 0] = n
        A[0, 1:] = x_mean
        A[1:, 0] = x_mean
        B = np.mean(np.tile(values, (c, 1)).T * np.column_stack((np.ones(n),
factors)), axis=0)
        coef = np.linalg.solve(A, B)
        return coef

print(f"X1 min: {x1_min}\tX1 max: {x1_max}")
print(f"X2 min: {x2_min}\tX2 max: {x2_max}")
print(f"X3 min: {x3_min}\tX3 max: {x3_max}")
print(f"Y min: {y_min}\tY max: {y_max}")

# Перевірка однорідності дисперсії за критерієм Кохрена
# отримання оптимальної кількості випробувань для однієї комбінації факторів
Gt = 0.3346
for i in range(m, 10):
    rand_y = np.random.randint(y_min, y_max, (n, i))
    std_y = np.std(rand_y, axis=1)**2
    print(f"Max Y dispersion: {std_y.max()}")
    Gp = std_y.max()/std_y.mean()
    print(f"m = {i} Gp: {Gp}\tGt: {Gt}")
    if Gp < Gt:
        m = i
        break

print(f"Mean Y dispersion: {std_y.mean()}")
factors = natural(norm_factors)
factors = np.concatenate((factors, interact_factors(factors, True)), axis=1)

# перевірка за критерієм студента
coefs_value = np.zeros(4)
coefs_value[0] = np.mean(rand_y.mean(axis=1))
for i in range(3):
    coefs_value[i+1] = np.mean(rand_y.mean(axis=1) * factors[:, i])
stud_crit = np.abs(coefs_value) / np.sqrt(std_y.mean()/(n*m))
ts = 2.042
sig_ind = np.argwhere(stud_crit > ts)

print(f"All coefs are significant: {len(sig_ind.flatten()) ==
k}\t{sig_ind.flatten()}")

# формування планів
plan = np.concatenate((factors, rand_y), axis=1)
plan = np.concatenate((plan, rand_y.mean(axis=1).reshape(n, 1)), axis=1)
plan = np.concatenate((plan, std_y.reshape(n, 1)), axis=1)
print(f"Natural plan:\n{plan}")

coefs = solve_coef(factors, rand_y.mean(axis=1))
print("Natural coefs", coefs)
significant_coefs = np.zeros_like(coefs)
significant_coefs[sig_ind] = coefs[sig_ind]
print("Significant coefs", significant_coefs)

reg_val = regression(factors, significant_coefs, n, k)

```

```

print("Natural mean Y - regression Y")
print(np.column_stack((rand_y.mean(axis=1), reg_val)))

# перевірка критерію Фішера
disp = m/(n - len(sig_ind.flatten())) * np.sum((reg_val -
rand_y.mean(axis=1))**2)
Fp = disp / std_y.mean()
Ft = 2.16
print(f"Fisher crit: {Fp < Ft}\t{Fp}\t{Ft}")

```

Результат виконання:

X1 min: -1 X1 max: 2

X2 min: -9 X2 max: 6

X3 min: -5 X3 max: 8

Y min: 195.0 Y max: 205.33333333333334

Max Y dispersion: 14.0

m = 3 Gp: 2.5 Gt: 0.3346

Max Y dispersion: 10.6875

m = 4 Gp: 1.7750865051903115 Gt: 0.3346

Max Y dispersion: 11.359999999999998

m = 5 Gp: 1.6945107398568018 Gt: 0.3346

Max Y dispersion: 16.472222222222218

m = 6 Gp: 2.283110882956878 Gt: 0.3346

Max Y dispersion: 12.77551020408163

m = 7 Gp: 1.6347493036211698 Gt: 0.3346

Max Y dispersion: 11.484374999999998

m = 8 Gp: 1.7127543886903835 Gt: 0.3346

Max Y dispersion: 13.111111111111109

m = 9 Gp: 1.759443339960238 Gt: 0.3346

Mean Y dispersion: 7.451851851851853

All coefs are significant: False [0 1 2 3]

Natural plan:

[[2.	6.	8.	12.	16.
	48.	96.	4.	36.	64.
	203.	204.	198.	197.	195.
	195.	203.	196.	203.	199.33333333
	13.11111111]				

[2.	6.	-5.	12.	-10.
	-30.	-60.	4.	36.	25.
	204.	198.	199.	196.	200.
	197.	203.	197.	195.	198.77777778
	8.39506173]				

[2.	-9.	-5.	-18.	-10.
	45.	90.	4.	81.	25.
	202.	198.	196.	200.	197.
	202.	200.	200.	199.	199.33333333
	3.77777778]				

[-1.	-9.	-5.	9.	5.
	45.	-45.	1.	81.	25.
	199.	204.	203.	203.	200.
	201.	197.	195.	200.	200.22222222
	7.72839506]				

[-1.	6.	8.	-6.	-8.
	48.	-48.	1.	36.	64.
	196.	195.	195.	198.	197.
	199.	196.	196.	199.	196.77777778
	2.17283951]				

[2.	-9.	8.	-18.	16.
	-72.	-144.	4.	81.	64.
	202.	202.	198.	202.	204.
	200.	202.	204.	195.	201.
	7.55555556]				
[-1.	-9.	8.	9.	-8.
	-72.	72.	1.	81.	64.
	197.	200.	203.	196.	195.
	196.	202.	202.	203.	199.33333333
	9.77777778]				
[-1.	6.	-5.	-6.	5.
	-30.	30.	1.	36.	25.
	202.	197.	195.	202.	199.
	200.	195.	200.	195.	198.33333333
	7.55555556]				
[-1.3225	-1.5	1.5	1.98375	-1.98375
	-2.25	2.975625	1.74900625	2.25	2.25
	203.	203.	201.	204.	196.
	200.	201.	196.	203.	200.77777778
	7.95061728]				
[2.3225	-1.5	1.5	-3.48375	3.48375
	-2.25	-5.225625	5.39400625	2.25	2.25
	196.	195.	201.	203.	202.
	202.	199.	200.	195.	199.22222222
	8.83950617]				
[0.5	-10.6125	1.5	-5.30625	0.75

-15.91875	-7.959375	0.25	112.62515625	2.25
200.	203.	203.	200.	200.
196.	199.	195.	195.	199.
8.44444444]				
[0.5	7.6125	1.5	3.80625	0.75
11.41875	5.709375	0.25	57.95015625	2.25
202.	198.	200.	197.	195.
204.	199.	198.	204.	199.66666667
8.66666667]				
[0.5	-1.5	-6.3975	-0.75	-3.19875
9.59625	4.798125	0.25	2.25	40.92800625
199.	199.	196.	195.	197.
196.	195.	202.	203.	198.
7.77777778]				
[0.5	-1.5	9.3975	-0.75	4.69875
-14.09625	-7.048125	0.25	2.25	88.31300625
202.	200.	201.	204.	199.
195.	195.	198.	197.	199.
8.44444444]				
[0.5	-1.5	1.5	-0.75	0.75
-2.25	-1.125	0.25	2.25	2.25
199.	201.	202.	204.	202.
202.	202.	201.	201.	201.55555556
1.58024691]]				

Natural coefs [2.42232723e+00 -2.03274401e+01 2.66114382e+00 -
3.55151566e+00


```

4.22704034e-01 -4.63042561e-01 6.27101829e-02 -1.41374782e-02
3.20603135e+01 1.42193158e+00 1.84312879e+00]
Significant coefs [ 2.42232723 -20.32744012 2.66114382 -3.55151566
0.
0. 0. 0. 0.
0. ]

```

Natural mean Y - regression Y

```

[[199.33333333 -50.67781533]
[198.77777778 -4.50811179]
[199.33333333 -44.42526912]
[200.22222222 16.55705124]
[196.77777778 10.30450503]
[201. -90.59497265]
[199.33333333 -29.6126523 ]
[198.33333333 56.47420856]
[200.77777778 19.98637757]
[199.22222222 -54.10714166]
[199. -41.31005512]
[199.66666667 7.18929103]
[198. 10.98771285]
[199. -45.10847694]
[201.55555556 -17.06038204]]

```

Fisher crit: False 26473.052121608325 2.16

Висновки: було проведено трьохфакторний експеримент з урахуванням квадратичних членів ,використовуючи центральний ортогональний композиційний план, знайдено адекватне рівняння регресії.