Лабораторна робота 12
Ідентифікація параметрів регресійних моделей методом найменших квадратів

12.1. Мета роботи

Вивчення методу найменших квадратів (МНК) для практичного розв'язання задач ідентифікації параметрів моделей; набуття навичок використання методу для розв'язання задачі ідентифікації параметрів моделей із застосуванням комп'ютера.

12.2. Методичні рекомендації щодо організації самостійної роботи

За темою лабораторної роботи студент повинен: *знати* загальне формулювання задачі ідентифікації параметрів моделі; у*міти* розв'язувати задачі ідентифікації параметрів моделі методом найменших квадратів [6]; у*міти* застосовувати процедури системи Python для розв'язання задачі ідентифікації параметрів регресійної моделі [3, 5].

Задача ідентифікації параметрів регресійної моделі в загальному вигляді формулюється таким чином.

Нехай деяка система описується вхідними *x* і вихідними *y* змінними і яким-небудь чином обрана структура моделі (тобто залежність *y* від *x*):

$$y = G(x; a) + e, \tag{12.1}$$

де $a \in R^k$, $a = (a_1, a_2, \dots a_k)^T$ – деякі параметри моделі;

e – помилка моделі (враховуючи випадкові помилки експерименту).

Необхідно на основі результатів спостереження за вхідними та вихідними змінними системи (даних експерименту) й обраного критерію знайти оцінку параметрів моделі, тобто побудувати оптимальну в деякому розумінні математичну регресійну модель.

Найбільш відомим і ефективним з методів розв'язання задачі ідентифікації параметрів моделі є **метод найменших квадратів**, сутність якого полягає в такому. Розглянемо випадок, коли вхідних змінних *x* може бути декілька ($x \in R^m$), а вихідна змінна *y* одна ($y \in R^1$).

Нехай є дані експериментів $(x^{(i)}, y_i)$, $\mathrm{i} = \overline{1,\mathrm{n}}$, причому значення y_i містять випадкову помилку. Вводиться функція (від параметрів a) виду:

$$\Phi(a) \equiv \sum_{i=1}^{n}\left[(y_i - G(x^{(i)}; a)\right]^2, \qquad (12.2)$$

яку можна розглядати як міру відхилення функції моделі $G(x; a)$ від даних експерименту $y_1, y_2, \ldots, y_n$. Тоді оцінку параметрів a моделі $G(x; a)$ можна визначити з умови найменшого відхилення значень $G(x; a)$ від даних експерименту. Тобто оцінку параметрів a знаходять як точку, в якій функція $\Phi(a)$ досягає по $a \in R^k$ мінімального значення (точка мінімуму). Нехай a^* – шукана оцінка параметрів. Тоді величини $r_i \equiv y_i - G(x^{(i)}; a^*)$, $\mathrm{i} = \overline{1,\mathrm{n}}$, називають ***залишками*** (залишковими відхиленнями), які використовуються для аналізу отриманих рішень.

Одним з критеріїв адекватності побудованої (навченою) моделі є те, що залишки r_i, $\mathrm{i} = \overline{1,\mathrm{n}}$, утворюють собою вибірку значень деякої випадкової величини, яка відповідає помилкам e з (12.1). Тому для цього будують графік залишків та обчислюють значення критерія Дарбіна-Уотсона

$$d = \frac{\sum (r_i - r_{i-1})^2}{\sum r_i^2} \qquad (12.3)$$

Для адекватності моделі має виконуватись умова $d \approx 2$, тобто кореляція залишків відсутня або дуже мала.

Для перевірки адекватності моделі також застосовують коефіцієнт кореляції між фактичними та модельними (прогнозними) даними:

$$k = \frac{\sum (y_i - \bar{y})(y_i^{(pr)} - \bar{y}_{pr})}{\sqrt{\sum (y_i - \bar{y})^2 \sum (y_i^{(pr)} - \bar{y}_{pr})^2}} \qquad (12.4)$$

Для адекватної моделі має виконуватися умова $k \approx 1$, тобто кореляція між фактичними та модельними даними велика.

Для пошуку точки мінімуму функції $\Phi(a)$ можна застосувати функцію **minimize**(fun, par) пакету **scipy.optimize** системи Python, де fun – ім'я цільової функції, par – початкове наближення точки мінімуму.

12.3. Контрольні приклади

Приклад 1. Знайдіть оцінку параметрів моделі з однією вхідною і однією вихідною змінними, обраної у вигляді полінома другого ступеня. Для оцінки параметрів моделі застосуйте метод найменших квадратів. Дані експерименту подані у табл. 12.1.

Таблиця 12.1

Дані експерименту

X	0.5	2.5	4.5	6.5	8.5
Y	4.0	205	124	384	875

Розв'язання. Виконаємо завдання з використанням процедури **minimize** пакету **scipy.optimize**.

```
from scipy.optimize import minimize

n = 5
x = [0.5, 2.5, 4.5, 6.5, 8.5]
y = [4, 205, 124, 384, 875]
# Задаємо вид функції моделі
def FunModel(x, a):
  return (a[0] + a[1]*x + a[2]*x**2)

# Визначаємо цільову функцію метода МНК
def FunMNK(a):
  S = 0
  for i in range(n):
    S = S + (y[i] - FunModel(x[i], a))**2
  return S

# Задаємо початкове значення для параметрів a функції моделі
a0 = [0, 0, 0]
# Знаходимо точку мінімуму функції FunMNK
res = minimize(FunMNK, a0, method='BFGS')
a1 = res.x
print(a1)
```

Output:

```
[ 87.64372549 -51.96785665  16.44642894]
```

Тобто, a[0] = 87.64372549; a[1] = -51.96785665; a[2] = 16.44642894. Для перевірки правильності отриманого розв'язку, а також наочної інтерпретації результатів моделювання побудуємо графіки (рис. 12.1):

```
      import numpy as np
      import matplotlib.pyplot as plt

      # Обчислюємо значення функції моделі в токах x
      ym = [0]*n
      for i in range(n):
        ym[i] = FunModel(x[i], a1)
      # Обчислюємо залишки
      r = [0]*n
      for i in range(n):
        r[i] = y[i] - ym[i]
      # Обчислюємо 100 точок на відрізку [x[0], x[-1]]
      a = x[0]; b = x[-1]; m = 100
      x1 = np.linspace(a, b, m)
      # Обчислюємо значення функції моделі  в токах x1
      y1m = FunModel(x1, a1)

      # Будуємо графіки для аналізу моделі
      fig, (ax1, ax2) = plt.subplots(1, 2)  # задаємо дві графічні
підобласті

      ax1.scatter(x, y, color = "red")  # виводимо графік даних
експерименту у вигляді точок
      ax1.plot(x1, y1m, color = "blue") # додаємо у 1-шу графічну
підобласть графік прогнозних даних по моделі
      ax1.set(xlabel='x', ylabel='y')

      ax2.scatter(x, r, color = "green")  # виводимо графік залишків
вигляді точок
      ax2.plot(x, r)                      # додаємо у 2-гу графічну
підобласть ще лінію
      ax2.set(xlabel='x', ylabel='r')
      ax2.axhline(0, color='k')           #x-axis line
```

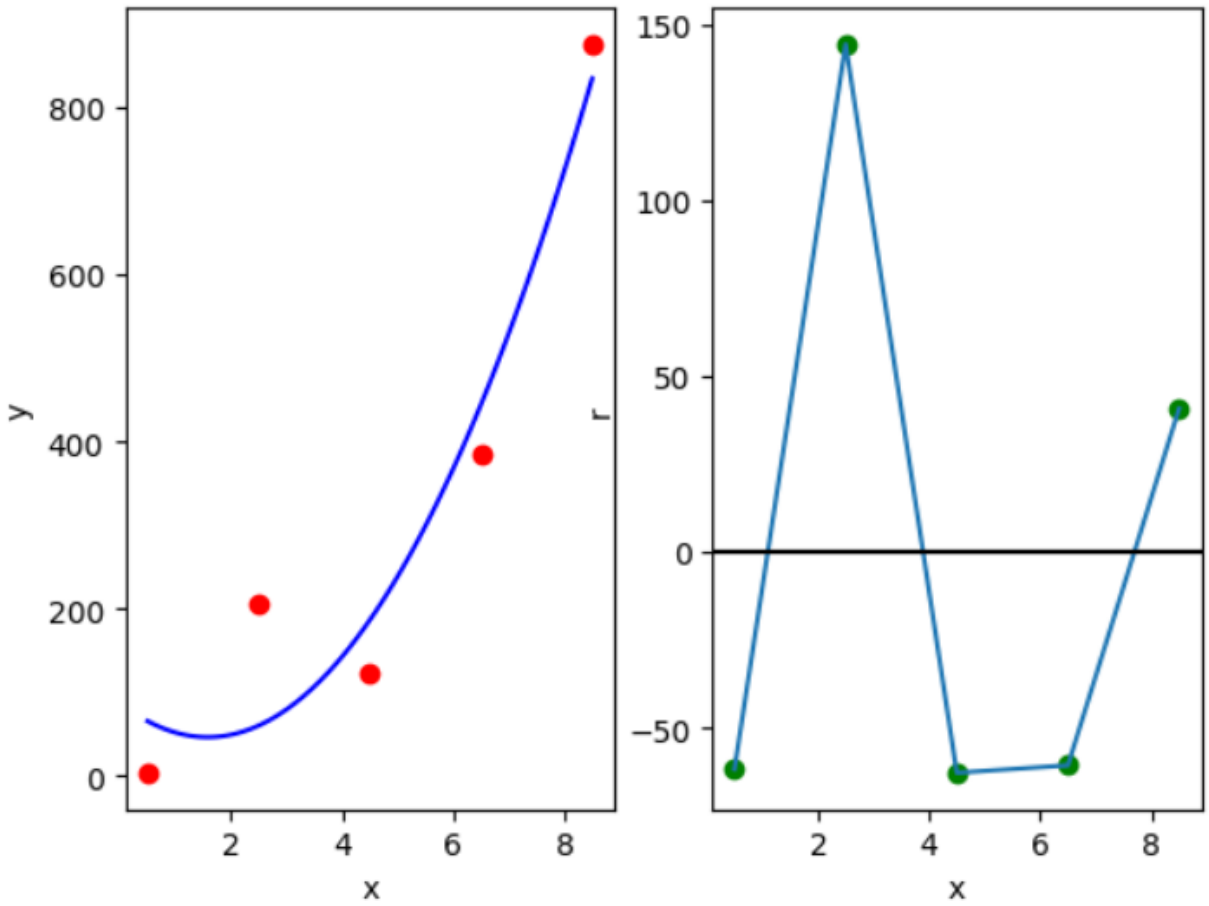

Рис. 12.1. Графік результатів оцінки параметрів моделі**.**

Якщо структура моделі (12.1) була обрана правильно, то графік залишків повинен мати хаотичний вигляд, тобто бути схожим на графік поведінки випадкової величини. У нашому випадку видно, що графік залишків містить у собі деяку закономірність (можливо, кубічну залежність). Тобто, структуру моделі (12.1), можливо, треба змінити.

Приклад 2 (модель роботи насосу). Залежність (HQ-характеристика насосу) створюваного насосом напору (H) від витрати води (Q) описується функцією $\boldsymbol{H = H_F - S_F Q^{\beta}}$. У результаті проведення натурних експериментів отримані дані, подані в табл. 12.2.

Таблиця 12.2

Дані натурних експериментів

Q, м^3/год.	200	500	900	1300	2000	2600
H, м вод. ст.	82	81	79.5	77	69	61.5

Необхідно знайти параметри (H_F, S_F, β) HQ-характеристики насоса і побудувати її графік.

Розв'язання. Виконаємо завдання з використанням процедур пакета **scipy**.

Оскільки розглядається залежність напору (H) від витрати води (Q), то Q – вхідна, а H – вихідна змінні. Тут функція $H = H_F - S_F Q^{\beta}$ – це структура моделі, а (H_F, S_F, β) – параметри моделі.

```
# Вводимо початкові данні
n = 6
x = [200., 500, 900, 1300, 2000, 2600]
y = [82, 81, 79.5, 77, 69, 61.5]
# Задаємо вид функції моделі
def FunModel(x, a):
  return (a[0] - a[1]*x**a[2])

# Визначаємо цільову функцію метода МНК
def FunMNK(a):
  S = 0
  for i in range(n):
    S = S + (y[i] - FunModel(x[i], a))**2
  return S

# Задаємо початкове значення для параметрів a функції моделі
a0 = [82, 0, 2]
# Знаходимо точку мінімуму функції FunMNK
res = minimize(FunMNK, a0, method='BFGS')
a1 = res.x
print("Параметри моделі =", a1)
```

Output:

```
Параметри моделі = [8.19337798e+01 3.05786878e-06 2.00047472e+00]
```

Для перевірки правильності отриманого розв'язку, а також наочної інтерпретації результатів моделювання побудуємо графіки (рис.12.2):

```
import numpy as np
import matplotlib.pyplot as plt

# Обчислюємо значення функції моделі в токах x
ym = [0]*n
for i in range(n):
  ym[i] = FunModel(x[i], a1)
# Обчислюємо залишки
r = [0]*n
for i in range(n):
  r[i] = y[i] - ym[i]
# Обчислюємо 100 точок на відрізку [x[0], x[-1]]
a = x[0]; b = x[-1]; m = 100
```

```
        x1 = np.linspace(a, b, m)
        # Обчислюємо значення функції моделі  в токах x1
        y1m = FunModel(x1, a1)

        # Будуємо графіки для аналізу моделі
        fig, (ax1, ax2) = plt.subplots(1, 2)  # задаємо дві графічні
підобласті

        ax1.scatter(x, y, color = "red")  # виводимо графік даних
експерименту у вигляді точок
        ax1.plot(x1, y1m, color = "blue") # додаємо у 1-шу графічну
підобласть графік прогнозних даних по моделі
        ax1.set(xlabel='x', ylabel='y')
```

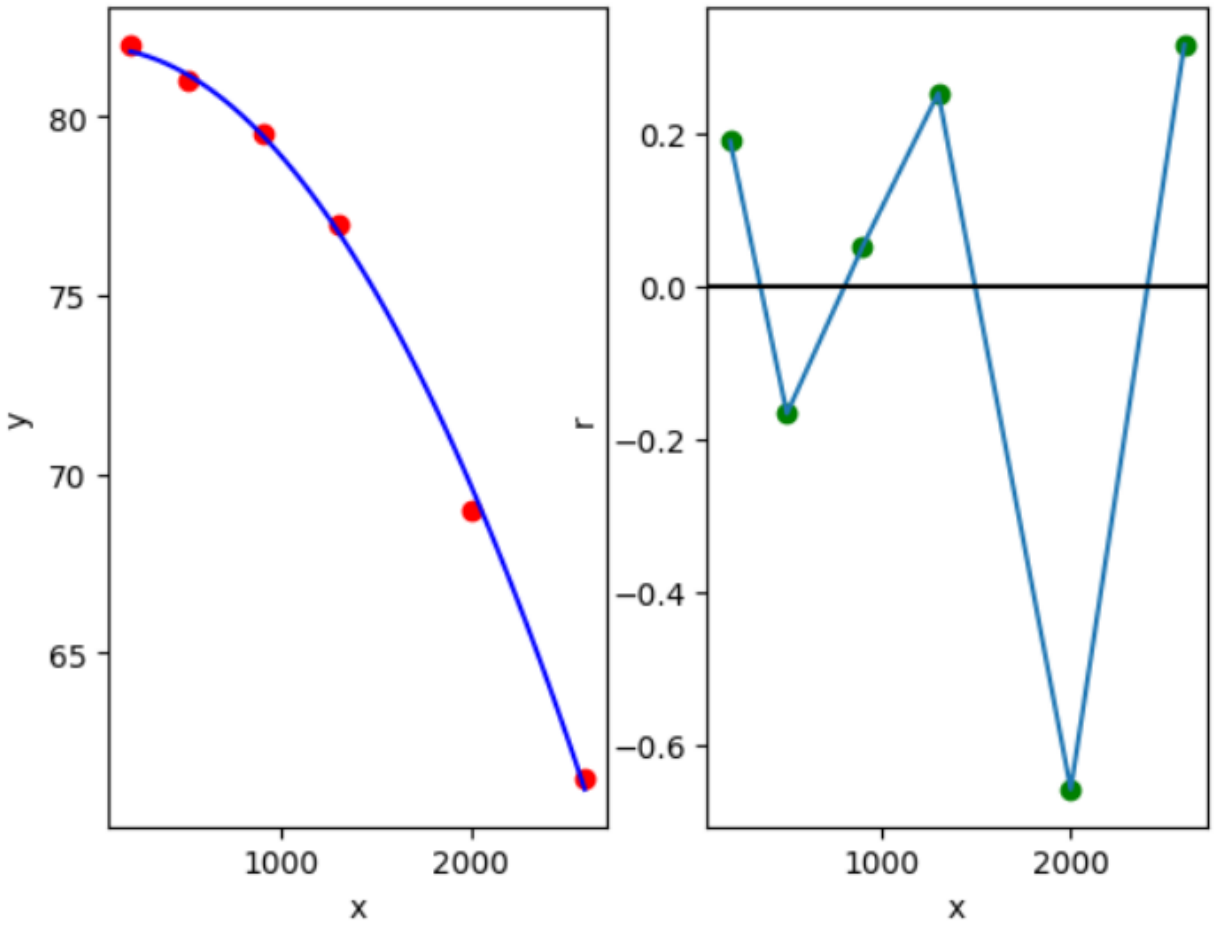

Рис. 12.2. Графік результатів оцінки параметрів моделі роботи насосу.

З графіків видно, що модель добре лягає на дані експерименту. Обчислимо коефіцієнт кореляції між фактичними та модельними (прогнозними) даними:

```
    import math
```

```
    # Коефіцієнт кореляції між фактичними та модельними даними (для
адекватної моделі має виконуватися умова k » 1)
    def CoefCorr(Y, Y_pred):
      n = len(Y)
      Y_mid = 0
      Y_pred_mid = 0
      for i in range(n):
        Y_mid = Y_mid + Y[i]
        Y_pred_mid = Y_pred_mid + Y_pred[i]
      Y_mid = Y_mid/n
      Y_pred_mid = Y_pred_mid/n

      S1 = 0
      S2 = 0
      S3 = 0
      for i in range(n):
        S1 = S1 + (Y[i] - Y_mid)*(Y_pred[i] - Y_pred_mid)
        S2 = S2 + (Y[i] - Y_mid)*(Y[i] - Y_mid)
        S3 = S3 + (Y_pred[i] - Y_pred_mid)*(Y_pred[i] - Y_pred_mid)

      kcor = S1/math.sqrt(S2*S3)
      return kcor

    k = CoefCorr(y, ym)
    print("Коефіцієнт кореляції =", k)
```

Output:

```
    Коефіцієнт кореляції = 0.9989894909033621
```

Як і очікувалося, кореляція доволі велика.

З рис. 12.2 також видно, що графік залишків має хаотичний вигляд. Перевіримо це по критерію Дарбіна-Уотсона:

```
    # Критерій Дарбіна-Уотсона (для адекватної моделі має виконуватися
умова d » 2)
    def DurbinWatson(r):
      n = len(r)
      S1 = 0
      S2 = 0
      for i in range(n):
        S1 = S1 + r[i]*r[i]
        if i > 0:
          t = r[i] - r[i-1]
          S2 = S2 + t*t
      du = S2/S1
      return du

    d = DurbinWatson(r)
    print("Критерій Дарбіна-Уотсона =", d)
```

Output:

```
Критерій Дарбіна-Уотсона = 3.005342012956457
```

Значення критерію Дарбіна-Уотсона доволі сильно відрізняється від 2, але це скоріше тому, що данних мало (всього 6).

Також є сенс оцінити дисперсію та середнє квадратичне відхилення залишків:

```
S = 0
for i in range(n):
  S = S + r[i]**2
Vdisp = S/(n-1)
print("Дисперсія залишків =", Vdisp)
Vsd = math.sqrt(Vdisp)
print("Стандартне відхилення залишків =", Vsd)
```

Output:

```
Дисперсія залишків = 0.1323267136149962
Стандартне відхилення залишків = 0.3637673894331324
Середнє значення y = 75.0
```

Як видно, оцінка середнього квадратичного відхилення залишків досить мала відносно значень вихідної змінної H (середнє = 75.0), тому можна вважати, що модель (HQ-характеристика насосу) побудована добре.

12.4. Порядок виконання роботи і варіанти завдань

12.4.1. Зміст звіту

У практичній частині роботи необхідно:
виконати приклади, що наведені в підрозділі 12.3;
виконати завдання 1 і 2, наведені в 12.4.2;
надати тексти складених програм та результати їх виконання.

12.4.2. Варіанти індивідуальних завдань

Завдання 1. Створіть синтетичні дані експерименту, використовуючи гіпотетичну залежність $y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \varepsilon$ між вхідною змінною x і вихідною змінною y, де випадкові помилки ε розподілені за рівномірним законом розподілу. Значення змінної x у кількості n візьміть на відрізку від x_1 до x_n з постійним кроком. Дані для значень n, x_1 , x_n і параметрів залежності b_j візьміть з відповідного варіанта завдань. Вважайте, що випадкова помилка ε розподілена на

інтервалі (-a, a), де a дорівнює $0{,}05y_{сер}$, $y_{\text{сер}}$ – середнє значення змінної y в даних експерименту.

Завдання 2. Використовуючи інструменти системи Python, знайдіть параметри моделі (12.1) за даними експерименту, що були створені в першому завданні. В якості структури моделі візьміть поліноми першого, другого та третього степеня. Побудуйте графіки функції моделі та відповідних залишків. Зробіть висновки відносно обраної структури моделі.

Таблиця 12.3

Варіанти завдань

Варіант	**n**	**x_1**	**x_n**	**Вектор b**
1	20	-3	3	(-4; -0,8; 1,6; 2,3)
2	20	0,5	4	(0,4; 0,3; 1,0; 1,7)
3	25	4,5	8	(7,7; 9,4; 11,4; 13,6)
4	25	0,4		(0,43; 0,94; 1,91; 3.6)
5	15	4	6,8	(13,88; 16,93; 20,47; 24.15)
6	15	1	6	(2,11; 2,45; 2,61; 2.73)
7	30	0,3	1,8	(4,39; 3,75; 4,98; 5.11)
8	30	1	6	(0,1; 0,21; 0,43; 0.5)
9	10	1	4	(4,11; 4,16; 3,23; 3.29)
10	15	0.5	4	(2,47; 2,86; 3,01; 2.91)
11	22	0,68	0,9	(0,8; 0,89; 1,02; 0.06)
12	23	0,43	0,8	(1,63; 1,73; 1,87; 0.7)
13	21	0,02	0,3	(1,02; 1,09; 1,14; 1.21)
14	19	0,11	0,5	(9,05; 6,61; 4,69; 1.6)
15	24	0,35	0,7	(2,47; 2,91; 2,01; 2.1)

12.5. Контрольні запитання

1. У чому полягає сутність метода найменших квадратів для ідентифікації параметрів моделі?
2. Що є вихідними даними при застосуванні метода найменших квадратів для ідентифікації параметрів моделі?
3. Для чого будують графік моделі?
4. Для чого будують графік залишків?
5. Який характер поведінки графіка залишків відповідає правильно обраній структурі моделі?