Mục lục

Chương 3

Nội suy và xấp xỉ bằng đa thức

3.1 Nội suy tổng quát

Cho không gian véctơ V. Ánh xạ tuyến tính $L : V \to \mathbb{R}$ gọi là một phiếm hàm tuyến tính.

Ví dụ 3.1. Trên không gian hàm V thường xét một số phiếm hàm:

$$L(f) = f(x_0)$$
$$L(f) = f^{(k)}(x_0)$$
$$L(f) = \int_a^b f(x) f_0(x)\, dx.$$

Cho hệ n hàm độc lập tuyến tính $\{f_1, f_2, \dots, f_n\}$ trong V và n phiếm hàm độc lập tuyến tính $L_1, L_2, \dots, L_n$. Khi đó tồn tại duy nhất hàm P có dạng

$$P = \sum_{i=1}^{n} c_i f_i \tag{3.1}$$

sao cho

$$L_i(P) = b_i, \ \forall i = \overline{1,n} \tag{3.2}$$

Thật vậy,

$$(3.2) \Leftrightarrow L_i\left(\sum_{j=1}^{n} c_j f_j\right) = b_i, \ \forall i = \overline{1,n}$$
$$\Leftrightarrow \sum_{j=1}^{n} L_i\left(f_j\right) c_j = b_i, \ \forall i = \overline{1,n}$$
$$\Leftrightarrow \sum_{j=1}^{n} a_{ij} c_j = b_i, \ \forall i = \overline{1,n}$$

hay

$$Ac = b \tag{3.3}$$

trong đó $A = \left(a_{ij}\right)_n$, $b = (b_1, b_2, \dots, b_n)$, $c = (c_1, c_2, \dots, c_n)$. Với các giả thiết độc lập tuyến tính của hệ hàm và hệ phiếm hàm, ta có $|A| \neq 0$. Do đó, bài toán có nghiệm duy nhất.

3.2 Đa thức nội suy

Tìm

$$P(x) = \sum_{i=0}^{n} a_i x^i$$

sao cho, tại mốc nội suy x_i:

$$P(x_i) = y_i, \ \forall i = \overline{0, n}. \tag{3.4}$$

Đây là trường hợp đặc biệt của bài toán nội suy tổng quát với hệ hàm

$$f_0 = 1, \ f_1 = x, \ f_2 = x^2, \dots, \ f_n = x^n$$

và phiếm hàm $L_i(f) = f(x_i)$. Ta có $L_i\left(x^j\right) = x_i^j$ với $i, j = \overline{0, n}$, trong đó quy ước $0^0 = 1$. Hoặc, ta biến đổi trực tiếp

$$(3.4) \Leftrightarrow \sum_{j=0}^{n} a_j x_i^j = b_i, \ i = \overline{0, n}$$

$$\Leftrightarrow \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ & \cdots\cdots\cdots & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \cdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix}.$$

Định thức của hệ là định thức Vandermonde: $|A| = \prod_{i<j} \left(x_j - x_i\right) \neq 0$.

Ví dụ 3.2. Tìm đa thức nội suy của hàm số có giá trị cho trong bảng

x	-1	0	1	2
y	4	3	2	7

Mô tả các điểm và đa thức bằng đồ thị.

Giải. Đa thức nội suy có dạng $P(x) = a + bx + cx^2 + dx^3$.

Cách 1: Ta có

$$P(-1) = a - b + c - d = 4, \quad P(0) = a = 3,$$
$$P(1) = a + b + c + d = 2, \quad P(2) = a + 2b + 4c + 8d = 7$$

Giải hệ được $a = 3, b = -2, c = 0, d = 1$, và kết luận $P(x) = 3 - 2x + x^3$.

```
from sympy import *
a, b, c, d = symbols('a b c d')
P = lambda x: a + b*x + c * x**2 + d * x**3
P(-1)  # a - b + c - d

solve( [P(-1) - 4, P(0) - 3, P(1) - 2, P(2) -7] , [a,
    b, c, d] )  # {a: 3, b: -2, c: 0, d:
    1}
```

hoặc tổng quát hơn

```
X = [-1, 0, 1, 2]
Y = [4, 3, 2, 7]
list( zip(X, Y) )  # [(-1, 4), (0, 3), (1, 2), (2, 7)]

from sympy import *
a = MatrixSymbol('a', 4, 1)
a[0]  # a_0

def P(x):
  tong = 0
  for i in range(4):
    tong += a[i] * x**i
  return tong

P(2)  # a_0 + 2a_1 + 4a_2 + 8a_3

solve( [P(x) - y for (x, y) in zip(X, Y)] , a )  # {a0:
    3, a1: -2, a2: 0, a3: 1}
```

Cách 2: Giải hệ

$$\begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 2 \\ 7 \end{bmatrix}.$$

```
X = [-1, 0, 1, 2]
Y = [4, 3, 2, 7]
```

```
A = [ [X[i] ** j for j in range(4)] for i in range(4)
    ]  # Python quy ước
```

$0^0 = 1$

```
import numpy as np
np.linalg.solve(A, Y)  # array([ 3., -2., -0., 1.])
```

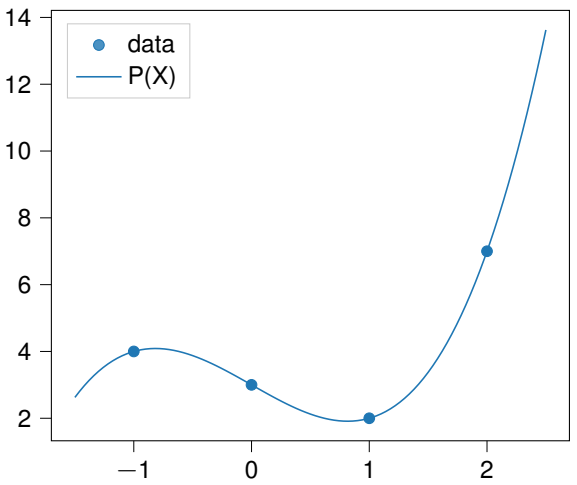
Cách 3: dùng lệnh có sẵn của gói sympy

```
from sympy import *
x = symbols('x')

interpolate([(-1, 4), (0, 3), (1, 2), (2, 7)], x)  #
```

$x^3 - 2x + 3$

Để vẽ các điểm nội suy và đa thức nội suy tìm được, ta dùng lệnh sau

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

X = [-1, 0, 1, 2]
Y = [4, 3, 2, 7]
plt.scatter(X, Y, label='data')

P = lambda x: x**3 - 2*x + 3
X = np.linspace(-1.5, 2.5, 101)
Y = [P(x) for x in X]
plt.plot(X, Y, label='P(x)')

plt.legend();
```



□

3.2.1 Đa thức Lagrange

Tìm

$$P(x) = \sum_{i=0}^{n} y_i L_i(x) \tag{3.5}$$

trong đó $L_i(x_j) = \delta_{ij} = \begin{cases} 0 & \text{nếu } i \neq j \\ 1 & \text{nếu } i = j \end{cases}$.

Ta có $L_i(x) = C\prod_{j\neq i}(x - x_j)$. Mặt khác $L_i(x_i) = 1$ nên $C\prod_{j\neq i}(x_i - x_j) = 1$. Vậy

$$L_i(x) = \prod_{j\neq i} \frac{x - x_j}{x_i - x_j} \tag{3.6}$$

Ví dụ 3.3. Trong Ví dụ 3.2, tìm đa thức nội suy bằng đa thức Lagrange.

Giải. Các đa thức Lagrange

$$\begin{aligned}
L_0(x) &= \frac{(x-0)(x-1)(x-2)}{(-1-0)(-1-1)(-1-2)} &&= -\frac{x^3}{6} + \frac{x^2}{2} - \frac{x}{3} \\
L_1(x) &= \frac{(x+1)(x-1)(x-2)}{(0+1)(0-1)(0-2)} &&= \frac{x^3}{2} - x^2 - \frac{x}{2} + 1 \\
L_2(x) &= \frac{(x+1)(x-0)(x-2)}{(1+1)(1-0)(1-2)} &&= -\frac{x^3}{2} + \frac{x^2}{2} + x \\
L_3(x) &= \frac{(x+1)(x-0)(x-1)}{(2+1)(2-0)(2-1)} &&= \frac{x^3}{6} - \frac{x}{6}.
\end{aligned}$$

Vậy

$$P(x) = 4L_0(x) + 3L_1(x) + 2L_2(x) + 7L_3(x) = x^3 - 2x + 3.$$

□

Cách 1: lập trình

```
X = [-1, 0, 1, 2]
Y = [4, 3, 2, 7]

def L(i, x):
    prod = 1
    for j in range(4):
        if j != i:
            prod *= (x - X[j]) / (X[i] - X[j])
```

```
    return prod

from sympy import *
x = symbols('x')
L(0, x)
L(0, x).expand()

P = 0
for i in range(4):
    P += Y[i] * L(i, x)

P
P.expand()
```

Cách 2: dùng lệnh của gói sympy

```
from sympy import *
x = symbols('x')

P = interpolating_poly(4, x, [-1, 0, 1, 2], [4, 3, 2, 7])  # 4 là số mốc nội suy
P.expand()
```

Cách 3: gói scipy

```
from scipy.interpolate import lagrange
lagrange([-1, 0, 1, 2], [4, 3, 2, 7])  # → poly1d([ 1., 0., -2., 3.])
```

Định lý 3.1. *Cho hàm số $f \in C^{n+1}[a,b]$. Giả sử $P(x)$ là đa thức nội suy hàm $f(x)$ tại các mốc nội suy $x_i \in [a,b]$, $i = \overline{0,n}$. Khi đó với mỗi $x \in [a,b]$, tồn tại $\xi(x)$ ở giữa $\min\limits_{0\le i\le n} x_i$ và $\max\limits_{0\le i\le n} x_i$, sao cho*

$$f(x) = P(x) + \frac{f^{(n+1)}\left(\xi(x)\right)}{(n+1)!}\prod_{i=0}^{n}(x - x_i).$$

3.3 Xấp xỉ số liệu và phương pháp Neville

3.4 Sai phân chia

3.5 Nội suy Hermite

3.6 Nội suy Newton

Định nghĩa 3.1 (Sai phân). *Cho dãy y_i, $i = \overline{0,n}$. Sai phân cấp k của dãy tại phần tử y_i, $i = \overline{0,n-k}$ được định nghĩa đệ quy:*

$$\begin{aligned} &\Delta^0 y_i = y_i, \ \forall i \\ &\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i, \ k = \overline{1,n}. \end{aligned} \tag{3.7}$$

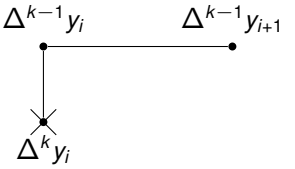
Ba sai phân cấp đầu tiên:

$$\begin{aligned} \Delta y_i &= \Delta^1 y_i = \Delta^0 y_{i+1} - \Delta^0 y_i = y_{i+1} - y_i \\ \Delta^2 y_i &= \Delta^1 y_{i+1} - \Delta^1 y_i = (y_{i+2} - y_{i+1}) - (y_{i+1} - y_i) = y_{i+2} - 2y_{i+1} + y_i \\ \Delta^3 y_i &= \Delta^2 y_{i+1} - \Delta^2 y_i = (y_{i+3} - 2y_{i+2} + y_{i+1}) - (y_{i+2} - 2y_{i+1} + y_i) \\ &= y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i. \end{aligned}$$

Tổng quát:

$$\Delta^k y_i = \sum_{j=0}^{k} (-1)^{k-j} C_k^j y_{i+j}.$$

Ta dùng lược đồ sai phân



để xây dựng bảng sai phân $\Delta^k y_i$ với $k = \overline{0,n}$, và $i = \overline{0,n-k}$.

k \ i	0	1	2	3	...
0	y_0	y_1	y_2	y_3	...
1	Δy_0	Δy_1	Δy_2		
2	$\Delta^2 y_0$	$\Delta^2 y_1$	...		
3	$\Delta^3 y_0$	...			
⋮	...				

Giả sử các mốc nội suy cách đều

$$x_i - x_{i-1} = h, \ \forall i = \overline{1,n}.$$

Công thức Newton tiến: tính $t = \dfrac{x - x_0}{h}$ và

$$\begin{aligned} P(x) &= \sum_{k=0}^{n} \frac{\Delta^k y_0}{k!} \prod_{i=0}^{k-1} (t - i) \\ &= y_0 + \Delta y_0 t + \frac{\Delta^2 y_0}{2!} t(t-1) + \cdots + \frac{\Delta^n y_0}{n!} t(t-1)\cdots(t-n+1) \end{aligned} \tag{3.8}$$

Công thức Newton lùi: tính $t = \dfrac{x - x_n}{h}$ và

$$\begin{aligned} P(x) &= \sum_{k=0}^{n} \frac{\Delta^k y_{n-k}}{k!} \prod_{i=0}^{k-1} (t + i) \\ &= y_n + \Delta y_{n-1} t + \frac{\Delta^2 y_{n-2}}{2!} t(t+1) + \cdots + \frac{\Delta^n y_0}{n!} t(t+1)\cdots(t+n-1) \end{aligned} \tag{3.9}$$

Ví dụ 3.4. Trong Ví dụ 3.2, tìm đa thức nội suy bằng đa thức nội suy Newton.

Giải. Các mốc nội suy cách đều với $h = 1$. Ta có bảng sai phân

k \ i	0	1	2	3
0	4	3	2	7
1	−1	−1	5	
2	0	6		
3	6			

```
X = [-1, 0, 1, 2]
Y = [4, 3, 2, 7]

d = lambda k, i: Y[i] if k == 0 else d(k-1, i+1) - d(k-1, i)

[ [d(k, i) for i in range(4-k)] for k in range(4) ]
```

Công thức Newton tiến: $t = \dfrac{x - (-1)}{1} = x + 1$ và

$$\begin{aligned} P(x) &= 4 + (-1)\, t + \frac{0}{2!} t(t-1) + \frac{6}{3!} t(t-1)(t-2) \\ &= 4 - (x+1) + (x+1)\, x\, (x-1) = x^3 - 2x + 3. \end{aligned}$$

Công thức Newton lùi: $t = \dfrac{x - 2}{1}$ và

$$P(x) = 7 + 5t + \frac{6}{2!} t(t+1) + \frac{6}{3!} t(t+1)(t+2)$$

$$= 7 + 5(x-2) + 3(x-2)(x-1) + (x-2)(x-1)x$$
$$= x^3 - 2x + 3.$$

Dưới đây là mã Python của công thức Newton tiến. Đối với công thức Newton lùi, ta chỉ sửa một chút tại các dòng có chú thích.

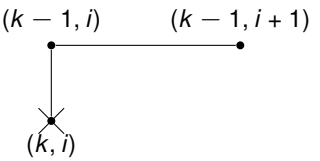
```
1  from sympy import *
2  x, t = symbols('x t')

3  P = Y[0]                              # Y[3]
4  for k in range(1, 4):
5      prod = d(k, 0) / factorial(k)     # d(k, 3-k)
6      for i in range(k):
7          prod *= t - i                 # t + i
8      P += prod

9  P
10 P.subs(t, (x - X[0]) / 1)             # X[3]
11 _.expand()
```

□

Ta cũng có thể định nghĩa cho sai phân bằng từ khóa `def`

```
def d(k, i):
    if k == 0:
        return Y[i]
    return d(k-1, i+1) - d(k-1, i)
```

hoặc dùng quy hoạch động theo lược đồ




```
Y = [4, 3, 2, 7]
for k in range(1, 4):
    for i in range(4-k):
        Y[i] = Y[i+1] - Y[i]   # Δ^k y_i
    print(Y[:4-k])
```

Khi đó để tính $P(x)$, ta thêm vào sau dòng 1 mã

```
P = Y[0]   # Y[3]
prod = 1
```

và sau dòng 5 mã đồng cấp (trong vòng lặp tại dòng 2)

```
    prod *= ( t - (k-1) ) / k   # t + (k-1), lưu 1/k! ∏_{i=0}^{k-1} (t ∓ i)
    P += Y[0] * prod            # Y[3-k]
```

Còn với công thức Newton lùi, ta cũng thay mã tương ứng ở phần chú thích tại vị trí phù hợp.

3.7 Nội suy spline bậc ba

3.8 Đường cong tham số

Tài liệu tham khảo

[1] Phạm Kỳ Anh. *Giải tích số*. Đại học Quốc gia Hà Nội, 2002. 284 trang.

[2] Richard L. Burden, Douglas J. Faires **and** Annette M. Burden. *Numerical Analysis*. phiên bản 10. Cengage Learning, 2016. 918 trang.

[3] NumPy community. *NumPy User Guide*. phiên bản 1.22.0. 531 trang. URL: https://numpy.org/doc/stable.

[4] SciPy community. *SciPy Reference Guide*. phiên bản 1.8.1. 3584 trang. URL: https://docs.scipy.org/doc.

[5] Phan Văn Hạp **and** Lê Đình Thịnh. *Phương pháp tính và các thuật toán*. Nhà xuất bản Giáo dục, 2000. 400 trang.

[6] Doãn Tam Hòe. *Toán học tính toán*. Đại học Quốc gia Hà Nội, 2009. 240 trang.

[7] Matplotlib development team. *Matplotlib documentation*. phiên bản 3.5.1. URL: https://matplotlib.org/3.5.1/tutorials/index.html.

[8] SymPy Development Team. *SymPy Documentation*. phiên bản 1.8. 2750 trang. URL: https://github.com/sympy/sympy/releases.