數值分析
Chapter 3 .
Interpolation and Polynomial Approximation
• Interpolation: determine the values at intermedia points.

• 任一函数 $f$ 若在 $[a, b]$ 間連續，則一定可以找到一多項式函數 $p$ 來 approximate $f$.

• Taylor polynomial. P.61 Figure 3.1

• 多項式函數之好處: 其微分與積分都還是多項式.

3.2 Lagrange Polynomials

• Given 3 pts $(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2))$

  $y = ax^2 + bx + c$

  $p(x) = f(x_0) \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1) \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f(x_2) \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$

  $\left\| \begin{array}{l}
  p(x_0) = f(x_0), p(x_1) = f(x_1), p(x_2) = f(x_2) \\
  L_{n,k}(x_k) = 1, L_{n,k}(x_i) = 0
  \end{array} \right.$

• Lagrange Polynomials

  Consider the construction of a polynomial of degree at most $n$ that passes through the $n+1$ points

  $(x_0, f(x_0)), (x_1, f(x_1)), \ldots, (x_n, f(x_n))$

  In this case, we need to construct, for each $k = 0, 1, \ldots, n$, a function $L_{n,k}(x)$ with the property that $L_{n,k}(x_i) = 0$ when $i \neq k$ and $L_{n,k}(x_k) = 1$. To satisfy $L_{n,k}(x_i) = 0$ for each $i \neq k$ requires that the numerator of $L_{n,k}(x)$ contains the terms

  $(x - x_0)(x - x_1) \ldots (x - x_{k-1})(x - x_{k+1}) \ldots (x - x_n)$

  To satisfy $L_{n,k}(x_k) = 1$, the denominator of $L_{n,k}(x)$ must be equal to this term evaluated at $x = x_k$. Thus,
\[ L_{n,k}(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)(x-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)} \]

- **nth Lagrange Interpolation Polynomial**

\[ P_n(x) = f(x_0)L_{n,0}(x) + \ldots + f(x_n)L_{n,n}(x) = \sum_{k=0}^{n} f(x_k)L_{n,k}(x) \]

where

\[ L_{n,k}(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)(x-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)} \]

for each \( k = 0, 1, \ldots, n \)

Ex 1. P.65 Using the numbers, or nodes, \( x_0 = 2, x_1 = 2.5 \) and \( x_2 = 4 \) to find the second interpolating polynomial for \( f(x) = \frac{1}{3} \) require that we first determine the coefficient polynomials \( L_0, L_1 \) and \( L_2 \). In nested form they are

\[ L_0(x) = \frac{(x-2.5)(x-4)}{(2.5-2)(2.5-4)} = (x-6.5)x + 10 \]

\[ L_1(x) = \frac{(x-2)(x-4)}{(2.5-2)(2.5-4)} = \frac{(-4x+24)x-32}{3} \]

\[ L_2(x) = \frac{(x-2)(x-2.5)}{(4-2)(4-2.5)} = \frac{(x-4.5)x+5}{3} \]

Since \( f(x_0) = f(2) = 0.5, f(x_1) = f(2.5) = 0.4, f(x_2) = f(4) = 0.25, \) we have

\[ P_2(x) = \sum_{k=0}^{2} f(x_k)L_k(x) = (0.05x - 0.425)x + 1.15 \]

An approximation to \( f(3) = \frac{1}{3} \) is

\[ f(3) \approx P(3) = 0.325 \]

- **Lagrange Polynomial Error Formula**

\[ f(x) = P_n(x) + \frac{f^{n+1}(...(\xi(x))(x - x_0)(x - x_1)\ldots(x - x_n)}{(n+1)!} \]

for some number \( \xi(x) \) between \( x_0, x_1, \ldots, x_n \) and \( x \).

P.67 Example 2, 用不同的點形成的 Lagrange 的 error 之分析.
• 必須考慮越多點（越高階）越好，可能原因是 error term 中的
ξ(\(x\)) 的可能範圍大，意味可能誤差大，尤其當加入那點的 \((x - x_k)\) 絕對值很大時（意即新加入的點，距離要求的 \(x\) 很遠時），error term 也會放大。

• Lagrange 之缺點，不知需要用多少點之資訊，即使用的越多也非最好。且新加一點後需重算，之前算的沒法 reuse.

• Recursively Generated Lagrange Polynomials
  Let \(f\) be defined at \(x_0, x_1, ..., x_k\) and \(x_i, x_j\), be two numbers in this set. 另一種 Lagrange 表示法:
  \[
P(x) = \frac{(x-x_j)P_{0,1,...,i-1,i+1,...,k}(x)-(x-x_i)P_{0,1,...,i-1,i+1,...,k}(x)}{(x_i-x_j)}
  \]
  then \(P(x)\) is the kth Lagrange Polynomial that interpolates \(f\) at the k+1 points \(x_0, x_1, ..., x_k\).
  Let \(Q \equiv P_{0,1,...,i-1,i+1,...,k}(x)\) and \(\bar{Q} \equiv P_{0,1,...,j-1,j+1,...,k}(x)\)
  \[
  \Rightarrow P(x) = \frac{(x-x_j)\bar{Q}(x)-(x-x_i)Q(x)}{(x_i-x_j)}
  \]
  \[
  \begin{array}{c|ccc}
    if \ x_r \neq x_i, x_j & x_r = x_i & x_r = x_j & \\
    Q & \bar{Q}(x_r) & f(x_i) & 0 \\
    \bar{Q} & Q(x_r) & 0 & f(x_j) \\
    P & f(x_r) & f(x_i) & f(x_j) \\
  \end{array}
  \]
  Table 3.4 Neville’s method

  \[
  \begin{align*}
  x_0 & \quad P_0 = Q_{0,0} = f(x_0) \\
  x_1 & \quad P_1 = Q_{1,0} = f(x_1) \\
  x_2 & \quad P_2 = Q_{2,0} = f(x_2) \\
  x_3 & \quad P_3 = Q_{3,0} = f(x_3) \\
  x_4 & \quad P_4 = Q_{4,0} = f(x_4) \\
  \end{align*}
  \]

  P.70 Ex 4. Suppose that we want to use Neville’s method to calculate the approximation to \(f(1.5)\). If \(x_0 = 1.0, x_1 = 1.3, x_2 = 1.6, x_3 = 1.9, \) and \(x_4 = 2.2\), then \(f(1.0) = Q_{0,0}, f(1.3) = Q_{1,0}, f(1.6) = Q_{2,0}, f(1.9) = Q_{3,0}\) and \(f(2.2) = Q_{4,0}\); so these are the five polynomials of degree
zero (constants) that approximate $f(1.5)$. Calculating the approximation $Q_{1,1}(1.5)$ gives

$$Q_{1,1}(1.5) = \frac{(1.5-1.0)Q_{1,0}-(1.5-1.3)Q_{0,0}}{(1.3-1.0)} = \frac{0.5(0.6200860)-0.2(0.7651977)}{0.3} = 0.5233449$$

Similarly,

$$Q_{2,1}(1.5) = \frac{(1.5-1.3)(0.4554022)-(1.5-1.6)(0.6200860)}{(1.6-1.3)} = 0.5102968$$

$Q_{3,1}(1.5)0.5132634$, $Q_{4,1}(1.5) = 0.5104270$

In a similarly manner, the approximation using quadratic polynomials are given by

$$Q_{2,2}(1.5) = \frac{(1.5-1.0)(0.5102968)-(1.5-1.6)(0.5233449)}{(1.6-1.0)} = 0.5124715$$

$Q_{3,2}(1.5) = 0.5112857$, $Q_{4,2}(1.5) = 0.5137361$

The higher-degree approximations are generated in a similar manner and are shown in Table 3.5

- **Divided Differences** (差分)

  The zeroth divided difference of the function $f$ with respect to $x_i$, $f[x_i]$, is simply the value of $f$ at $x_i$:

  $$f[x_i] = f(x_i)$$

  The remaining divided differences are defined inductively. The first divided differences of $f$ with respect to $x_i$ and $x_{i+1}$ is denoted $f[x_i, x_{i+1}]$ and is defined as

  $$f[x_i, x_{i+1}] = \frac{f[x_{i+1}]-f[x_i]}{x_{i+1}-x_i}$$

  After the (k-1)st divided differences, $f[x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k-1}]$ and $f[x_{i+1}, x_{i+2}, \ldots, x_{i+k-1}, x_{i+k}]$ have been determined, the kth divided differences relative to $x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k}$ is

  $$f[x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+k}]-f[x_i, x_{i+1}, x_{i+2}, \ldots, x_{i+k-1}]}{x_{i+k}-x_i}$$

  With this notation, it can be shown that

  $$P_n(x) = f[x_0] + f[x_0, x_1](x-x_0) + f[x_0, x_1, x_2](x-x_0)(x-x_1) + \ldots + f[x_0, x_1, \ldots, x_n](x-x_0)(x-x_1)\ldots(x-x_{n-1})$$
(多個點之泰勒展開式，因 \( x_{i+k} - x_i \) 越除越大，所以分母不用乘以 \( n! \))

\( (f'' \text{ vs. second divided differences}) \)

\[
\begin{align*}
&f(x_0 - h) & f(x_0) & f(x_0 + h) \\
&x_0 - h & x_0 & x_0 + h
\end{align*}
\]

\[
f'' \approx \frac{f(x_0 + h) - f(x_0) - f(x_0) - f(x_0 - h)}{h^2} = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}
\]

divided difference: \( x_0 = x_0 - h, x_1 = x_0, x_2 = x_0 + h \)

\[
f[x_0, x_1] = \frac{f(x_0) - f(x_0 - h)}{h}
\]

\[
f[x_1, x_2] = \frac{f(x_0 + h) - f(x_0)}{h}
\]

\[
\Rightarrow f[x_0, x_1, x_2] = \frac{f(x_0 + h) - f(x_0) - f(x_0 - h)}{2h}
\]

- **Newton’s Interpolatory Divided-Difference Formula**

\[
P(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \ldots, x_k](x - x_0)\ldots(x - x_{k-1})
\]

See P.75 Table 3.8, Example 1.

- Newton’s interpolatory divided-difference formula has a simpler form when \( x_0, x_1, \ldots, x_n \) are arranged consecutively with equal spacing. In this case, we introduce the notation \( h = x_{i+1} - x_i \) for each \( i = 0, 1, \ldots, n - 1 \) and let \( x = x_0 + sh \).

Then the difference \( x - x_i \) can be written as \( x - x_i = (s - i)h \), and the divided-difference formula becomes

\[
P_n(x) = P_n(x_0 + sh)
\]

\[
= f[x_0] + sh f[x_0, x_1] + s(s - 1)h^2 f[x_0, x_1, x_2] + \ldots + s(s - 1)\ldots(s - n + 1)h^n f[x_0, x_1, \ldots, x_n]
\]

\[
= f[x_0] + \sum_{k=1}^{n} s(s - 1)\ldots(s - k + 1)h^k f[x_0, x_1, \ldots, x_k]
\]
Using a generalization of the binomial-coefficient notation,
\[
\binom{s}{k} = \frac{s(s-1)...(s-k+1)}{k!}
\]

- **Newton Forward Divided-Difference Formula**

\[
P_n(x) = P_n(x_0 + sh) = f[x_0] + \sum_{k=1}^{n} \binom{s}{k} h^k f[x_0, x_1, ..., x_k]
\]
\[
= f[x_0] + \sum_{k=1}^{n} \binom{s}{k} \Delta^k f(x_0) \quad (\Delta f_i = f_{i+1} - f_i)
\]

- **backward Divided-Difference Formula.** P.78

- **P.78 Example 2.**

  - $x$ 靠近 $x_0$, 用 **forward divided difference**，$x$ 靠近 $x_n$, 用 **backward divided difference**, 若 $x$ 在中間, 兩法都會有些誤差, 此時可改用 **centered-difference formula**.

### 3.4 Hermite Interpolation

(只要有 $f'$的資訊, 即可用此法)

- **Lagrange fits n+1 nodes ⇒ degree n, 但 Hermite 除了 fits n+1 nodes, 還要 fits 所有之 $f'$（共 n+1 個）⇒ 需多加 n+1 degree ⇒ 共 2n+1 degree.**

- **Hermite Polynomial**

  Suppose that $f \in C^1[a,b]$ and that $x_0,..., x_n$ in $[a,b]$ are distinct. The unique polynomial of least degree agreeing with $f$ and $f'$ at $x_0,..., x_n$ is the polynomial of degree at most $2n+1$ given by

\[
H_{2n+1}(x) = \sum_{j=0}^{n} f(x_j) H_{n,j}(x) + \sum_{j=0}^{n} f'(x_j) \bar{H}_{n,j}(x)
\]
where
\[ H_{n,j}(x) = [1 - 2(x - x_j)L'_{n,j}(x_j)]L^2_{n,j}(x) \]
and
\[ \hat{H}_{n,j}(x) = (x - x_j)L^2_{n,j}(x) \]
Here, \( L_{n,j} \) denotes the jth Lagrange coefficient polynomial of degree n, and \( L'_{n,j}(x_j) = L'_{n,j}(x) \big|_{x=x_j} \)

\[
\begin{align*}
\begin{cases}
H_{n,j}(x_j) = 1 & \hat{H}_{n,j}(x_j) = 0 \\
H_{n,j}(x_i) = 0 & \hat{H}_{n,j}(x_i) = 0
\end{cases}
\end{align*}
\]
and
\[
\begin{align*}
\begin{cases}
H'_{n,j}(x_j) = 0 & \hat{H}'_{n,j}(x_j) = 1 \\
H'_{n,j}(x_i) = 0 & \hat{H}'_{n,j}(x_i) = 0
\end{cases}
\end{align*}
\]
given \[
\begin{align*}
\begin{cases}
L_{n,j}(x_j) = 1 \\
L_{n,j}(x_i) = 0
\end{cases}
\end{align*}
\]

• 但此法很麻煩, 除了要算 Lagrange 之外, 還要算 Lagrange 之微分, 此外, 通常很難知道每一點之微分值．

• Divided-Difference Form of the Hermite Polynomial.  
  P.84 Table 3.11.  P.85 Example 1.

• Hermite Polynomial Error Formula  
  If \( f \in C^{2n+2}[a, b] \), then  
  \[ f(x) = H_{2n+1}(x) + \frac{f^{2n+2}(\xi(x))}{(2n+2)!} (x - x_0)^2... (x - x_n)^2 \]
  for some \( \xi(x) \) with \( a < \xi(x) < b \)
3.5 Spline Interpolation

- 之前用的方法，次方太高了，有 oscillatory drawback.

- 

$[x_i, x_{i+1}]$ 若知道 $f(x_i), f(x_{i+1})$ 與 $f'(x_i), f'(x_{i+1})$ 則此段可用 cubic Hermite 來估：

$$H_3(x) = ax^3 + bx^2 + cx + d$$ (但很常不知道 $f'$)

- Cubic Spline Interpolation

Given a function $f$ defined on $[a, b]$ and a set of nodes, $a = x_0 < x_1 < ... < x_n = b$, a cubic spline interpolant, $S$, for $f$ is a function that satisfies the following conditions:

(a) For each $j = 0, 1, ..., n - 1$, $S(x)$ is a cubic polynomial, denoted by $S_j(x)$, on the subinterval $[x_j, x_{j+1})$

(b) $S_j(x_j) = f(x_j)$ for each $j = 0, 1, ..., n$

(c) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, ..., n - 2$

(d) $S'_j(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, ..., n - 2$

(其實並不需要真的知道 $f'$, $S'$ 也未不等於 $f'$)

(e) $S''_{j+1}(x_{j+1}) = S''_{j}(x_{j+1})$ for each $j = 0, 1, ..., n - 2$

(其實並不需要真的知道 $f''$, $S''$ 也未不等於 $f''$)

(f) One of the following sets of boundary conditions is satisfied:

(i) $S''(x_0) = S''(x_n) = 0$ (natural or free boundary)

(ii) $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$

(clamped boundary)
\[ y = a_1 x^3 + b_1 x^2 + c_1 x + d_1 \]
\[ y = a_0 x^3 + b_0 x^2 + c_0 x + d_0 \]
\[ y = a_2 x^3 + b_2 x^2 + c_2 x + d_2 \]

\[ f(x_0) = a_0 x_0^3 + b_0 x_0^2 + c_0 x_0 + d_0 \]
\[ f(x_1) = a_0 x_1^3 + b_0 x_1^2 + c_0 x_1 + d_0 \]
\[ f(x_2) = a_0 x_2^3 + b_0 x_2^2 + c_0 x_2 + d_0 \]
\[ f''(x_0) = 6a_0 x_0 + 2b_0 = 0 \]
\[ f''(x_1) \equiv 3a_0 x_1^2 + 2b_0 x_1 + c_0 = 3a_1 x_1^2 + 2b_1 x_1 + c_1 \]
\[ f''(x_2) \equiv 6a_0 x_2 + 2b_0 = 6a_1 x_2 + 2b_1 \]

\[ f(x_1) = a_1 x_1^3 + b_1 x_1^2 + c_1 x_1 + d_1 \]
\[ f(x_2) = a_1 x_2^3 + b_1 x_2^2 + c_1 x_2 + d_1 \]
\[ f''(x_3) = 6a_1 x_3 + 2b_1 = 6a_2 x_3 + 2b_2 \]

\[ f(x_2) = a_2 x_2^3 + b_2 x_2^2 + c_2 x_2 + d_2 \]
\[ f(x_3) = a_2 x_3^3 + b_2 x_3^2 + c_2 x_3 + d_2 \]
\[ f''(x_3) = 6a_2 x_3 + 2b_2 = 0 \]
3.6 Parametric Curves

- 因不是函數，所以將 x 與 y 分開看，x, y 分別是參數 t 之函數。

- P.100 EX1. Construct a pair of Lagrange polynomials to approximate the curve shown in Figure 3.12, using the data points shown on the curve.

There is flexibility in choosing the parameter, and we will choose the points $t_i$ equally spaced in [0,1]. In this case, we have the data in Table 3.16

<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td>0</td>
<td>0.25</td>
<td>0.5</td>
<td>0.75</td>
<td>1</td>
</tr>
<tr>
<td>$x_i$</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$y_i$</td>
<td>0</td>
<td>1</td>
<td>0.5</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

- $x(t)$ 與 $y(t)$ 可用 Lagrange polynomial 來表示.
• Piecewise cubic Hermite Polynomial

We can simplify the process to one of determining a pair of cubic Hermite polynomials in the parameter $t$, where $t_0 = 0, t_1 = 1$, given the endpoint data $(x(0), y(0))$, and $(x(1), y(1))$ and the derivatives $dy/dx$ (at $t = 0$) and $dy/dx$ (at $t = 1$).

$$\frac{dy}{dx} \ (at \ t = 0) = \frac{y'(0)}{x'(0)} \ and \ \frac{dy}{dx} \ (at \ t = 1) = \frac{y'(1)}{x'(1)}$$

因 $k_0y'(0) = \frac{y'(0)}{x'(0)}$, $k_1y'(1) = \frac{y'(1)}{x'(1)}$

$\Rightarrow$ 需假设 desired tangent line (P.102 Figure 3.14, 3.15)

<table>
<thead>
<tr>
<th>Lagrange</th>
<th>Advantages</th>
<th>Convenient form; Easy to program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disadvantage</td>
<td>Hard to calculate by hand; 二邊之波動變化大，不知需用多少點資訊，用的多也未必好 (fit 未必好, 且太複雜) ⇒ 避免</td>
<td></td>
</tr>
<tr>
<td>Newton (Divided Differences)</td>
<td>計算量 &lt; Lagrange</td>
<td>A difference or divided difference table must be prepares; 要估的點要在這些點的中間最好; 微分值會越來越小, round off 要注意</td>
</tr>
</tbody>
</table>

• Hermite

很精確, 因為考慮了 $f'$

Cubic spline

當資料點多時, 不會形成高階之函數; 需要解聯立方程組;

fit 很好;

且一階, 二階微分連續

Parametric Curves

可用於非函數之曲線

一段動, 其他段不動

或只動一點 (piecewise cubic Hermite 也可達到 類似效果)